

ulm university universität
uulm

Alexander Andreas Nassal aus Heidenheim an der Brenz

Simulation von Softwareprojekten

Ein allgemeines Simulationsmodell für Planspiele zu Softwaretechnik und Projektmanagement

Dissertation

zur Erlangung des Doktorgrades Dr. rer. nat. der Fakultät für Ingenieurwissenschaften, Informatik und Psychologie der Universität Ulm

Universität Ulm, 2017
Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Softwaretechnik und Programmiersprachen
Institutsleiter: Prof. Dr. Matthias Tichy

Amtierender Dekan: Prof. Dr. Frank Kargl

Erstgutachter: Prof. Dr. Matthias Tichy, Universität Ulm
Zweitgutachter: Prof. Dr. Kurt Schneider, Leibniz Universität Hannover

Tag der Promotion: 28. Juli 2017

Zusammenfassung

Im Lehrbetrieb im Bereich der Softwaretechnik stößt man mitunter auf Situationen, in denen eine praxisnahe Schulung schwierig, teuer, riskant oder zeitaufwändig ist. Planspiele sind ein etabliertes Werkzeug, um in diesen Fällen nicht ausschließlich auf eine reine Wissensvermittlung zurückgreifen zu müssen. Dazu müssen entsprechende Szenarien, Storylines und Spielmechaniken geschaffen werden, die dem Spieler einen realitätsnahen Eindruck des Stoffs vermitteln können, und ihm erlauben in einer sicheren Spielumgebung zu experimentieren.

Um die im Spiel behandelten Aspekte zu simulieren, wird ein entsprechendes Modell benötigt. Bisherige Planspiele verwenden entweder Modelle mit einem geringen Detailgrad oder speziell für ihren Zweck entwickelte Lösungen. Eine Wiederverwendung bestehender Modelle bei der Entwicklung neuer Planspiele ist wünschenswert, scheitert jedoch meist daran, dass die existierenden Modelle nicht auf eine solche Wiederverwendung ausgelegt sind. Das Erstellen eines neuen Modells ist ebenfalls mit einigen Schwierigkeiten verbunden. Insbesondere der Einarbeitungsaufwand in die zu simulierenden Aspekte, der Umfang und die Komplexität des Modells, sowie dessen Validierung, stellen einen Entwickler vor große Herausforderungen. Ein Ansatz, um die Entwicklung neuer Planspiele zu vereinfachen, und Entwicklungszeit sowie -aufwand dabei zu reduzieren, ist die Bereitstellung eines allgemeinen Modells, das als Simulationskern neuer Planspiele eingesetzt werden kann. Ein solches Modell existiert derzeit jedoch nicht.

Diese Arbeit stellt die Konzeption und Realisierung eines solchen Simulationsmodells und einer dazu passenden Ausführungsumgebung vor. Bei der Entwicklung des Modells wurde ein neuer Ansatz getestet, der nicht die direkt beobachtbaren Effekte rekonstruiert, sondern diese durch die Kombination elementarer Teilmodelle erzeugt. Dazu werden aus der Literatur bekannte Modelle der Softwareentwicklung, sowie weitere Modelle aus angrenzenden Bereichen wie Psychologie und Ökonomie kombiniert und ausführbar gemacht. Diese Arbeit zeigt, dass mit einem auf diese Weise entstandenen Modell die elementaren Prozesse der Softwareentwicklung aus Projektmanagementsicht simuliert werden können, und stellt die Vorteile dieses Ansatzes heraus.

Das Modell berücksichtigt dabei auch die menschlichen Aspekte, die in der Projektarbeit eine wesentliche Rolle spielen, jedoch in bestehenden Planspielen nicht ausreichend behandelt werden. Das Simulationsmodell enthält dazu ein eigenes Teilmodell, das, basierend auf Modellen der Verhaltenspsychologie, die Handlungsplanung der simulierten Mitarbeiter realisiert. Damit ist es möglich, eine realitätsnahe Simulation der Projektarbeit anzubieten.

Durch eine entsprechende Architektur von Modell und Ausführungsumgebung kann das Modell durch Konfiguration und Parametrisierung für unterschiedliche Planspiele wiederverwendet werden. Dazu wird das zu simulierende Softwareprojekt auf ein abstraktes allgemeines Datenmodell abgebildet, auf dem die Simulation ausgeführt wird.

Das so konzipierte Modell wurde vollständige implementiert und bei der Entwicklung eines ersten Planspiels eingesetzt.

Diese Arbeit beschreibt auch die Evaluation des neuen Modells, die anhand verschiedener Szenarien und Testfällen durchgeführt wurde. Dadurch wurde vor allem sichergestellt, dass das Modell die Aspekte, welche in Softwaretechnik und Projektmanagement vermittelt werden sollen, adäquat abbildet, und sich das Modell passend zur Realität verhält. Dazu wurden Effekten, die bei der Durchführung von realen Softwareprojekten beobachtet werden können, am Simulationsmodell gezeigt.

Das Simulationsmodell ist in ein Framework eingebettet, das den praktischen Einsatz des Modells unterstützt. Es enthält neben Modell und Ausführungsumgebung auch einige unterstützende Werkzeuge, wie beispielsweise eine grafische Darstellung des Simulationsverlaufs. Wie Modell und Framework zur Entwicklung neuer Planspiele eingesetzt werden können, wird in dieser Arbeit anhand mehrerer Beispiele erläutert.

Abstract

In the field of software engineering education, one sometimes encounters situations in which practical training is difficult, expensive, risky or time-consuming. In such situations, planning games are an established allowing one tool not to be restricted to mere knowledge transfer. For this, appropriate scenarios, storylines and game mechanics have to be created, which can give the player a realistic impression of the learning material and allow him to experiment in a safe game environment.

In order to simulate the aspects treated in the game, a suitable model is needed. Existing planning games either use models with a low level of detail, or solutions specifically designed for their purpose. Reuse of existing models in the development of new planning games is desirable but usually fails because the existing models are not designed for such a reuse. Creating a new model also involves some difficulties. In particular, the complexity of the model, as well as its validation, is a great challenge to a developer. The provision of a general model that can be used as the simulation core of new planning games is one approach to simplify the development of such games, and to reduce development time and effort. However, such a model does not currently exist.

This work presents the conception and realization of such a simulation model and a suitable execution environment. For the development of the model, a new approach was tested which does not reconstruct the directly observable effects, but generates them by combination of small elementary models. For this purpose, well-known models of software development, as well as other models from related fields, such as psychology and economics, are combined and made executable. This work shows that a model developed in this way can be used to simulate the basic processes of software development from a project management view, and highlights the advantages of this approach.

The model also considers the human aspects which play an essential role in project work, but are not treated adequately in existing simulation games. The simulation model contains a separate sub model, which is based on models of behavioral psychology and realizes the action planning of the simulated employees. This makes it possible to offer a realistic simulation of project work.

An appropriate architecture of the model and the execution environment allows the developer to re-use the model for different planning games via configuration and parameterization. For this purpose, the software projects that are to be simulated are mapped to an abstract general data model, on which the simulation is executed.

The model was implemented completely and used in the development of a first simulation game.

This work also describes the evaluation of the new model, which was based on various scenarios and test cases. This ensures that the aspects which are important in software engineering and project management are adequately implemented, and that the model is consistent with reality. For this purpose, effects that can be observed during the execution of real software projects have been shown with the simulation model.

The simulation model is embedded in a framework that supports the practical use of the model. In addition to model and execution environment, it also contains some supporting tools such as a graphical representation of the simulation process. The usage of model and framework for developing new planning games is explained in this thesis by several examples.

Danksagung

An dieser Stelle möchte ich meinen ganz besonderen Dank den nachstehenden Personen entgegen bringen, ohne deren Mithilfe diese Arbeit niemals zustande gekommen wäre:

Bedanken möchte ich mich bei Herrn Prof. Dr. Helmuth Partsch, der es mir ermöglicht hat, dieses spannende Promotionsthema zu wählen, und mir bei meinen ersten Schritten im Reich der Wissenschaft zur Seite gestanden hat.

Ich bedanke mich ganz besonders bei Herrn Prof. Dr. Matthias Tichy, der sich ohne zögern bereit erklärt hat, nach dem Ausscheiden von Prof. Partsch die Betreuung meiner Promotion zu übernehmen, und mich vor allem bei der Erstellung dieser Dissertation mit großem persönlichem Engagement und Zeiteinsatz unterstützt hat.

Prof. Dr. Kurt Schneider danke ich für die unkomplizierte Bereitschaft zur Übernahme des Zweitgutachtens meiner Arbeit.

Mein besonderer Dank gilt auch Herrn Dr. Klaus Murmann für die großzügigen Freiräume und das damit verbundene Vertrauen, was eine Promotion in diesem Rahmen überhaupt erst möglich gemacht hat.

Ganz herzlich bedanken möchte ich mich bei meinen Kollegen des Instituts für Softwaretechnik und Programmiersprachen und der Servicegruppe Informatik für die stets freundschaftliche Zusammenarbeit, die uneingeschränkte Unterstützung bei Problemen, die vielen spannenden und inspirierenden Diskussionen und vor allem für die spaßige gemeinsame Zeit.

Nicht zuletzt danke ich meiner Familie, die mich während der ganzen Zeit immer unterstützt und ermutigt hat.

Alexander Nassal, Juli 2017

Anmerkung

Teile dieser Dissertation wurden bereits in folgenden Fachartikeln veröffentlicht:

Alexander Nassal. *A General Framework for Software Project Management Simulation Games*. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*, Seite 321–325. ISSN 2166-0727.

Alexander Nassal und Marcel Dausend. *A Board Game for Educational Purposes in Software Engineering*. In Georg Hagel und Jürgen Mottok (Herausgeber), *European Conference Software Engineering Education (ECSEE) 2014*, Seite 235–239. Shaker, November 2014. ISBN 978-3844030679.

Alexander Nassal. *Projektmanagement spielend lernen*. In Axel Schmolitzky und Anna S. Hauptmann (Herausgeber), *Tagungsband des 14. Workshops "Software Engineering im Unterricht der Hochschulen" (SEUH) 2015, Dresden, Deutschland, 26.–27. Februar 2015*, Band 1332 der *CEUR Workshop Proceedings*, Seite 53–64. CEUR-WS.org, 2015. ISSN 1613-0073.

Alexander Nassal und Matthias Tichy. *Modeling Human Behavior for Software Engineering Simulation Games*. In *Proceedings of the 5th International Workshop on Games and Software Engineering*, GAS '16, Seite 8–14. ACM, 2016. ISBN 978-1450341608.

Alexander Nassal und Matthias Tichy. *Ein Framework zur Erstellung von Planspielen zur Softwaretechnik*. In Bernd Bruegge und Stephan Krusche (Herausgeber), *Tagungsband des 15. Workshops "Software Engineering im Unterricht der Hochschulen" (SEUH) 2017, Hannover, Deutschland, 22.–23. Februar 2017*, Band 1790 der *CEUR Workshop Proceedings*, Seite 51–62. CEUR-WS.org, 2017. ISSN 1613-0073.

Inhaltsverzeichnis

1 Einführung **5**
- 1.1 Planspiele . . . 6
- 1.2 Existierende Planspiele zur Softwaretechniklehre . . . 8
  - 1.2.1 Beispiele für rechnergestützte Planspiele . . . 8
  - 1.2.2 Diskussion . . . 11
- 1.3 Motivation der Arbeit . . . 12
- 1.4 Ziele, Abgrenzung und Ansatz der Arbeit . . . 15
- 1.5 Ergebnisse . . . 17
- 1.6 Aufbau der Arbeit . . . 18

2 Ein Simulationsmodell für Planspiele **19**
- 2.1 Simulation und Modell . . . 20
  - 2.1.1 Die drei Hauptmerkmale von Modellen . . . 20
  - 2.1.2 Modellgrenze und Schnittstelle zur Umgebung . . . 21
  - 2.1.3 Simulationskonzept . . . 22
- 2.2 Framework . . . 23
- 2.3 Datenmodell . . . 26
  - 2.3.1 Einheiten und spezielle Datentypen . . . 26
  - 2.3.2 Statische Struktur . . . 28
  - 2.3.3 Notation und Beispiele . . . 34
- 2.4 Elementarmodelle . . . 37
  - 2.4.1 Physis- und Leistungsmodelle . . . 39
  - 2.4.2 Kognitive Modelle . . . 48
  - 2.4.3 Kommunikationsmodelle . . . 56
  - 2.4.4 Modelle der Artefaktbearbeitung . . . 59
  - 2.4.5 Motivationsmodell . . . 67
  - 2.4.6 Modell zum Aufgabenwechsel . . . 77
- 2.5 Tätigkeitsmodelle . . . 79
  - 2.5.1 Zeitliche Struktur . . . 79
  - 2.5.2 Erholung . . . 80
  - 2.5.3 Aufgabentätigkeiten . . . 80
- 2.6 Verhaltensmodelle . . . 87
  - 2.6.1 Die subjektive Wahrnehmung der Agenten . . . 88
  - 2.6.2 Handlungsplanung . . . 92
  - 2.6.3 VIE-Planung . . . 96
  - 2.6.4 Ergebnisse und Handlungs-Ergebnis-Kontingenz im VIE-Modell . . . 100
  - 2.6.5 Folgen und Ergebnis-Folgen-Kontingenz im VIE-Modell . . . 106
  - 2.6.6 Bedürfnisse und Folgen-Bedürfnis-Kontingenz im VIE-Modell . . . 108
  - 2.6.7 Beispielkonfigurationen des VIE-Modells . . . 110
  - 2.6.8 Zusammenhang zwischen Bedürfnisvektor und Motivationsmodell . . . 115
- 2.7 Realisierung des Simulationsmodells . . . 115

3 Evaluation des Simulationsmodells **119**
- 3.1 Auswahl und Konkretisierung der zu zeigenden Effekte . . . 119

3.2 Validierung von Systemmodellen . . . 123
3.3 Vorgehensweise zur Evaluation des Simulationsmodells . . . 127
3.4 Durchgeführte Experimente und Ergebnisse . . . 131
3.4.1 Prüfung der aufgestellten Hypothesen . . . 131
3.4.2 Weitere Analyse von Parameterabhängigkeiten . . . 160
3.5 Betrachtung der Validitätsgüte . . . 165
3.6 Zusammenfassung der Evaluationsergebnisse . . . 168

4 Anwendungen des Modells 169
4.1 Werkzeuge . . . 169
4.1.1 Dateiformat für Szenarien . . . 169
4.1.2 Grafische Oberfläche für Szenarien und Simulation . . . 170
4.1.3 Evaluationsframework . . . 173
4.2 Konzeption und Entwicklung von Planspielen . . . 175
4.3 Fallstudie: Ein Planspiel zum Softwaregrundprojekt . . . 180
4.4 Weitere Anwendungsbeispiele . . . 187
4.4.1 Schnittstellenentwurf und Integration . . . 187
4.4.2 Referenzdokumente für Inspektionen . . . 190
4.4.3 Einbeziehung des Kunden bei der Inspektion nicht-technischer Dokumente . . . 191
4.4.4 Testgetriebene Entwicklung . . . 192
4.4.5 Anforderungsänderungen während der Entwicklung . . . 193
4.4.6 Agiles Vorgehen mit Scrum . . . 194

5 Bewertung und Diskussion 197
5.1 Zusammenfassung und Bewertung . . . 197
5.2 Vergleich mit anderen Modellierungsansätzen . . . 202
5.2.1 Abgrenzung zur Prozessmodellierung . . . 202
5.2.2 Modelle zur Dynamik von Softwareentwicklungsprozessen . . . 203
5.3 Vergleich mit Baukästen zur Planspielentwicklung . . . 205
5.3.1 SESAM . . . 205
5.3.2 SimSE . . . 206
5.4 Weiterführende Schritte . . . 207
5.4.1 Verbesserungen an Framework und Simulationsmodell . . . 208
5.4.2 Erweiterung der Evaluation . . . 211
5.4.3 Weitere Einsatzfelder für das Simulationsmodell . . . 212

A Anhang 215
A.1 Übersicht über die Attribute der Szenarioelemente . . . 216
A.2 Übersicht über die verwendeten mathematischen Symbole . . . 221
A.3 Definition der Zustände des Motivationsmodells . . . 223
A.4 Liste der bei der Evaluation verwendeten Parameter . . . 224
A.5 Navarros 86 fundamentale Regeln des Software Engineering . . . 232

Abbildungsverzeichnis 251

Literaturverzeichnis 255

Index 261

1 Einführung

Die Entwicklung qualitativ hochwertiger Software stellt alle daran Beteiligten auch heute noch vor eine große Herausforderung. Obwohl in der Vergangenheit schon viele Richtlinien, Prozesse und Methoden zur Softwareentwicklung definiert und etabliert wurden, fehlen noch an vielen Stellen klare Vorgaben, wie gute Software zu entwickeln ist. Es existieren derzeit keine universellen Methoden, die gleichermaßen für alle Systeme angewendet werden können [Sommerville, 2007]. Vielmehr sind die Methoden von den jeweils zu lösenden Problemen, des Projektrahmens und des zu entwickelnden Systems abhängig.

Informatik und Softwaretechnik sind vergleichsweise junge Disziplinen, die einer schnellen Entwicklung unterliegen. Das betrifft nicht nur die technischen Aspekte, sondern zeigt sich auch in der Häufigkeit, mit der sich Vorgehen und Methodik bei der Softwareentwicklung ändern. Während in anderen Ingenieursdisziplinen, wie Maschinenbau oder Architektur, Wissen und Erfahrung schon über einen langen Zeitraum gesammelt werden konnte, und sich daraus klare Vorgehensweise ergeben haben, fehlt es der Softwaretechnik teilweise noch an konkretem Knowhow in vielen Bereichen.

»Im Software Engineering geht es jedoch hauptsächlich darum, die am besten geeignete Methode für die jeweiligen Umstände zu wählen, [...]« [Sommerville, 2007, S. 33]. Welche das ist, lässt sich oft nicht anhand formaler Kriterien entscheiden. Die Wahl muss deshalb oft auf Basis der subjektiven Einschätzung der Beteiligten getroffen werden. Wie gut diese Einschätzung zutrifft wird wiederum maßgeblich durch deren Erfahrung bestimmt. Eine wesentliche Rolle fällt dabei dem Projektleiter zu, der durch die Auswahl und Überwachung des Entwicklungsprozesses und der dazu passenden Führung seiner Mitarbeiter erheblich am Projekterfolg beteiligt ist. Dazu muss er neben dem Wissen über das Repertoire an Methoden, auch über die Fähigkeit verfügen, die Umstände richtig zu bewerten.

Ziel einer guten Lehre im Bereich der Softwaretechnik sollte es daher sein, auch Erfahrung zu vermitteln. Das stellt die Lehrenden oft vor eine große Herausforderung, denn Erfahrung lässt sich nur schwer theoretisch im Form von Frontalunterricht, wie er beispielsweise in Vorlesungen stattfindet, gewinnen. Viel wichtiger ist es, dass die Studierenden sich Erfahrungen in Softwaretechnik und Projektmanagement selbst aneignen, indem sie mit den Problemen direkt konfrontiert werden. Das findet im Idealfall in möglichst realistischen Szenarien, wie beispielsweise Projekten und Praktika, sowie speziellen Übungsaufgaben statt, und setzt die persönliche Aktivität der Lernenden voraus. Besonders im Softwaretechnikbereich trägt dies zu einem tieferen Verständnis bei, da sich so die Tragweite der anfänglich fälschlicherweise oft als trivial erscheinenden Probleme besser begreifen lässt. Dieser Ansatz wird *Erfahrungsbasiertes Lernen* genannt. Dabei wird neues Wissen durch die Transformation von Erfahrung geschaffen [Kolb, 1984]. Einige Studien haben gezeigt, dass das Lernen durch eigene Erfahrungen von Erwachsenen häufig bevorzugt wird, und dieses Wissen besser auf deren aktuelle Probleme angewendet werden kann [Knowles, 1984].

Vor allem im Bereich der Prozesse und des Projektmanagements ist es aufwändig und schwierig, praktische Beispiele und Übungen zu finden, die den Studierenden einen adäquaten Eindruck von Problemen und Lösungen auf diesem Gebiet aufzeigen. Das Problem hierbei entsteht mitunter dadurch, dass sich die zu übenden Aspekte nur im Kontext eines vollständigen Projekts betrachten lassen. Einige Aspekte, wie beispielsweise Aufwandsschätzung oder Terminplanung, lassen sich zwar Anhand von Ausschnitten aus fiktiven oder realen Projekten vermitteln, die in Form von Übungen von den Studierenden bearbeitet werden, sobald sich jedoch ein Aspekt nicht mehr auf einzelne gut abgrenzbare Teile, sondern auf das ganze Projekt bezieht, wie beispielsweise die Einstellung oder Entlassung von Mitarbeitern, stoßen solche Übungen an ihre Grenzen. Besonders im Bereich Mitarbeitereinsatz und -führung sind ausschnitthafte Übungen, die über die Abfrage theoretischen Wissens hinausgehen, nur schwer zu konzipieren, da hier weiche Faktoren wichtig sind, und sich für diese nur schwer klare Regeln und Handlungsvorgaben definieren lassen.

Ein wichtiges Ziel der Übungen ist es, den Studierenden die komplexen Zusammenhänge in

Softwareprojekten aufzuzeigen, und ihnen ein Gefühl für die Probleme und Schwierigkeiten zu vermitteln, die für solche Projekte typisch sind. Dabei sollen die Studierenden das zuvor theoretisch erworbene Wissen selbst erfahren und anwenden lernen. Das funktioniert nach unserer Erfahrung am besten in realen Projekten. Da solche Projekte einen erheblichen organisatorischen und zeitlichen Aufwand, sowohl für die Lehrenden, als auch für die Studierenden haben, ist die Anzahl der Projekte im Studium begrenzt. Liegt der Fokus auf Prozessen und Projektmanagement, kommt erschwerend hinzu, dass die Studierenden erheblich mehr Zeit auf die Erstellung der eigentlichen Software und den damit verbundenen technischen Herausforderungen und Schwierigkeiten verwenden, als auf das eigentliche Projektmanagement. Aufgrund der geringen Anzahl an möglicher Projekte während des Studiums, ist es außerdem nicht möglich, die vielen unterschiedlichen Vorgehensweisen zu behandeln, mit denen Software heutzutage entwickelt wird.

Aufgrund des zeitlichen Umfangs der Projekte ist es in der Regel auch nicht möglich, ein Projekt mehrmals zu durchlaufen, was den Studierenden die Möglichkeit nimmt, unterschiedliche Strategien und Methoden am selben Beispiel auszuprobieren. Risiken, die zum Scheitern des Projekts führen werden häufig durch den Lehrenden auf ein Minimum reduziert, indem das Vorgehen Schritt für Schritt vorgegeben wird. Das wiederum nimmt den Studierenden die Freiheit zu experimentieren und bewusst oder unbewusst Fehler zu machen, die ihrerseits wiederum eine wertvolle Erfahrung darstellen können.

1.1. Planspiele

Eine interessante Möglichkeit, eine Übung an einem vollständigen Projekt ohne großen Zeiteinsatz für Studierende und Lehrende anzubieten, stellen Planspiele dar. Hierbei wird ein Projekt als abstraktes Modell in Form einer Simulation bereitgestellt. Die Studierenden haben die Möglichkeit, sowohl auf die Rahmenbedingungen, als auch auf den Projektverlauf Einfluss zu nehmen, und die Auswirkungen zu beobachten.

Planspiele ermöglichen ein »planmäßiges Durchspielen einer bestimmten Situation, eines Vorhabens als Modellfall (besonders im militärischen Bereich)« [Dudenredaktion, 2006]. Die Idee, reale Situationen in starker Abstraktion in Spielform nachzubilden, ist schon weit über tausend Jahre alt. So entstanden die ersten Planspiele in Form des *Chaturanga* und des *Schachspiels* schon im frühen Mittelalter im asiatischen Raum. Im Laufe der Zeit wurde dieser Ansatz weiterentwickelt, um vor allem im militärischen Bereich, Strategien und Taktiken in einem risikofreien Raum testen und perfektionieren zu können. Mit der Entwicklung der Computertechnologie entstanden auch die ersten computergestützten Planspiele; zuerst im Militärbereich ab dem II. Weltkrieg, kurze Zeit später auch in der Wirtschaft.

Heute werden Planspiele in den unterschiedlichsten Bereichen eingesetzt. Vorherrschend sind immer noch die militärische und betriebswirtschaftliche Anwendung, aber auch viele weitere Gebiete wie beispielsweise der Katastrophenschutz, Politik, oder das Projektmanagement sind hinzugekommen.

Auch im Bereich der Softwaretechnik werden Planspiele vor allem in der Ausbildung eingesetzt. In Abschnitt 1.2 werden beispielhaft einige existierende Planspiele vorgestellt. Ziel dieser Spiele ist es, den Lernenden ein tieferes Verständnis zu ermöglichen, indem sie die Problematik selbst erkennen und erfahren, anstatt sie nur von außen erzählt zu bekommen [Reich, 2008, S. 95]. Gleichzeitig fördern Spiele auch die Motivation und den Spaßfaktor beim Lernen [Prensky, 2007]. Dazu werden die relevanten Aspekte in Spielform umgesetzt. Durch das Prinzip der *Gamification*, bei dem gängige Konzepte aus konventionellen Spielen, wie beispielsweise Ranglisten, Erfolge, oder Quests unter Berücksichtigung des Lernziels in die Lernumgebung integriert werden, kann so die Motivation der Spieler gesteigert werden.

Klassifikation

Planspiele können laut [Blötz, 2008] nach vier unterschiedlichen Arten klassifiziert werden:

- Das **Spielmedium** bestimmt die Form der Darstellung der Spielinhalte und die Interaktion mit dem Spiel. Mögliche Formen sind beispielsweise rechnergestützten Spielen, bei denen das Spiel in Form einer direkt durch den Spieler bedienten Software vorliegt, und formularbasierte

Ansätze, bei denen der Spieler seine Entscheidungen mittels Formulare direkt oder indirekt an eine zentrale Simulationssoftware weitergibt. Planspiele können aber auch völlig ohne Computer auskommen und beispielsweise mit Stift und Papier, oder in Form eines Brettspiels durchgeführt werden.

- Der **Modellbereich** legt das Bezugssystem des Spiels fest und beschreibt, welche Thematik durch das Planspiel behandelt wird. Das kann beispielsweise eine Branche, ein Funktionsbereich, ein Prozess, oder eine bestimmte Problemsituation sein.
- Mit der **Tutoriellen Begleitung** wird die didaktische Unterstützung des Spielers beschrieben. Das kann beispielsweise durch einen Trainer, durch in das Spiel integrierte Hilfsfunktionen und Tutorials passieren, oder auch völlig unbegleitet sein.
- Das **Soziale Arrangement** beschreibt die Spielersituation. Spielt ein Spieler alleine gegen das Modell, so spricht man von *Individualplanspielen*. Spielen mehrere Spieler miteinander oder gegeneinander spricht man von *Gruppenplanspielen*. Im Gegensatz zu den Gruppenplanspielen kennen sich bei *Fernplanspielwettbewerben* die Spieler normalerweise gegenseitig nicht. Während Gruppenplanspiele häufig einen kooperativen Charakter haben, steht bei Fernplanspielwettbewerben in der Regel der Konkurrenzaspekt im Vordergrund.

Der Ansatz dieser Arbeit beschränkt sich auf rechnergestütze Planspiele im Modellbereich der Softwaretechnik und des Projektmanagements. Die tutorielle Begleitung und das soziale Arrangement haben im Kontext dieser Arbeit keine Relevanz, da sie sich auf die Realisierung des Simulationskerns beschränkt, und didaktische Aspekte nicht behandelt werden.

Der restliche Teil dieser Arbeit bezieht sich auf Planspiele im Kontext der Softwaretechniklehre. Wenn nicht explizit anders gesagt, ist mit *Lehre* die Lehre im Bereich Softwaretechnik und Management von Softwareprojekten gemeint, und mit *Planspiel* ein Planspiel das zur Verdeutlichung der dort gelehrten Inhalte, bzw. zu deren Übung gedacht ist.

Vorteile für die Lehre

Der Einsatz von rechnergestützen Planspielen in der Lehre hat einige sehr konkrete Vorteile für den Lernprozess:

- Da die Simulation nicht an die *Realzeit* gebunden ist, kann das Projekt jederzeit pausiert, oder auch in deutlich höherer Geschwindigkeit durchgeführt werden, als es in der Realität ablaufen würde. Diese Möglichkeit des Zeitraffens erlaubt es, ein Projekt, das gegebenenfalls mehrere Jahre dauert, in wenigen Stunden durchzuspielen.
- Bei geeigneter Umsetzung ist der Projektverlauf vollständig reproduzierbar. Insbesondere lassen sich so die einzelnen Vorgänge auch im Nachhinein leichter nachvollziehen.
- Zeitraffer und Reproduzierbarkeit ermöglichen es, unterschiedliche Strategien innerhalb desselben *Szenarios* auszuprobieren. Durch das Abspeichern relevanter Zustände während des Projektverlaufs muss das Projekt dazu nicht zwangsläufig von Beginn an durchgeführt werden.
- Dem Spieler entsteht dabei weder das Risiko mit einem realen Projekt zu scheitern, was negative finanzielle Folgen oder Auswirkungen auf Reputation und Karrierechancen, und im Lehrbetrieb gegebenenfalls Auswirkung auf die Benotung haben kann, noch das Risiko dabei viel Zeit zu verlieren.
- Auswirkungen der im Projektverlauf getroffenen Entscheidungen können zeitnah beobachtet werden. Während in realen Projekten die Auswirkungen mancher Entscheidungen erst nach einigen Wochen oder Monaten sichtbar werden, können diese in Planspielen durch die Simulation unmittelbar angezeigt werden. Das erleichtert es dem Spieler, Aktion und Resultat in Zusammenhang zu bringen, und steigert dadurch den Lerneffekt.
- Durch eine geeignete Aufzeichnung des Projektverlaufs kann der Studierende hinter die Kulissen schauen, und somit gegebenenfalls leichter nachvollziehen, wie sein Projektergebnis zustande gekommen ist.

1.2. Existierende Planspiele zur Softwaretechniklehre

Die Idee, Planspiele in der Softwaretechniklehre einzusetzen, ist nicht neu. Erste Projekte, die sich mit diesem Thema befassen, lassen sich schon zu Beginn der 90er Jahre finden. Auch heute noch werden spielerische Ansätze verfolgt und dazu entsprechende Spiele entwickelt.

Diese Spiele behandeln das Themengebiet der Softwaretechnik entweder auf der Makroebene des Prozess- und Projektmanagements, oder widmen sich sehr konkreten Teilaspekten der Softwareentwicklung, wie beispielsweise dem Schreiben von gutem Code, Techniken zum Erfassen von Anforderungen oder Architekturmustern. Ein Beispiel für ein Spiel, das sich mit einem Teilaspekt beschäftigt ist *Pex4Fun* [Tillmann, 2013]. Hier kann man die Grundzüge des Programmierens auf Basis generierter White-Box-Tests und anhand verschiedener Programmierpuzzle spielerisch erlernen und üben. Da sich diese Arbeit auf Planspiele beschränkt, die sich mit der abstrakteren Ebene des Prozess- und Projektmanagement beschäftigen, werden in diesem Kapitel nur Spiele behandelt, die sich auf diesem Abstraktionsgrad bewegen.

Neben den rechnergestützten Planspielen, gibt es auch einige Spiele in Karten- oder Brettspielform, wie beispielsweise in [Baker, 2005], [Nassal, 2014], oder [Taran, 2007] beschrieben. Da solche Spiele ohne ein aufwändiges Simulationsmodell im Hintergrund auskommen müssen, fallen sie nicht in den Fokus dieser Arbeit und werden im Folgenden nicht besprochen.

1.2.1. Beispiele für rechnergestützte Planspiele

In der Vergangenheit wurden einige Planspiele entwickelt, um den Studierenden die Grundzüge der gängigen Prozessmodelle und des Projektmanagements für Softwareprojekte näher zu bringen. Meist konzentrieren sich die einzelnen Spiele dabei auf einen bestimmten Schwerpunkt, den sie spielerisch behandeln.

Tabelle 1.1 gibt eine Übersicht über die am häufigsten in der entsprechenden Literatur zitierten Spiele, und vergleicht sie anhand verschiedener Kriterien. Die Informationen basieren auf den Angaben in [Pieper, 2013] und eigenen Untersuchungen im Rahmen dieser Arbeit. Die einzelnen Spiele werden im Folgenden kurz erläutert.

	OSS [Sharp, 2000]	SESAM [Ludewig, 1992]	SimjavaSP [Shaw, 2005]	SimSE [Navarro, 2006]	SimVBSE [Jain, 2006]	Incredible Manager [de Oliveira Barros, 2006]	MO-SEProcess [Ye, 2007]
öffentlich verfügbar	×	✓	×	✓	✓	✓	×
Datum der letzten Version	2000	2003	2005	2006	2006	2006	2007
Spieler in der Rolle des Projektmanagers	×	✓	✓	✓	✓	✓	×
Simulation menschlicher Aspekte	×	×	×	×	×	×	×
unterschiedliche Szenarien enthalten	×	×	×	✓	×	×	×
Wiederholbarkeit von Spielabschnitten	×	✓	×	✓	×	×	×
für Wiederverwendung entworfen	×	×	×	×	×	×	×
eigene Modelle möglich	×	✓	×	✓	×	×	×
dokumentiertes Modell	×	✓	×	×	×	×	×

Tabelle 1.1.: Vergleich existierender Planspiele [Pieper, 2013]

OSS

OSS [Sharp, 2000] steht für *Open Software Solutions* und ist der Name eines fiktiven Softwarehauses. Dieses ist Teil eines Softwaretechnik-Fernkurses der *Open University* in Milton Keynes. Der Studierende tritt diesem Softwarehaus als neuer Mitarbeiter bei, und absolviert in diesem Rahmen verschiedenen Aufgaben zur Softwareentwicklung. Im Gegensatz zu vielen anderen Planspielen in diesem Bereich, nimmt er dabei nicht die Rolle eines Projektleiters ein.

Das OSS führt unterschiedliche Projekte durch, und stellt dem Studierenden eine Reihe von Ressourcen zur Verfügung, mit denen er seinen Aufgaben lösen muss. Dazu gehören beispielsweise die Einarbeitung in Prozessmodelle und Richtlinien, sowie deren Überprüfung anhand vorgegebener Werkstücke. Weitere Aufgaben sind beispielsweise die Modellierung eines Roboterverhaltens mittels UML. Das entstandene Modell kann anschließend anhand einer Animation geprüft werden. Auch Projektmanagementtätigkeiten, wie das Aufstellen eines Projektplans, sind Teil der Aufgabensammlung.

OSS benötigt kein allgemeines Simulationsmodell. Zwar werden Ergebnisse der einzelnen Aufgaben teilweise animiert, um dem Spieler zu verdeutlichen, wo Probleme liegen, eine eigentliche Simulation findet jedoch nicht statt. OSS ist vielmehr eine multimediale Lernplattform, deren Aufgaben geschickt in eine Storyline eingefügt wurden.

SESAM

SESAM [Ludewig, 1992] ist eines der ersten echten Planspiele im Bereich der Softwaretechnik. Der Name ist ein Akronym für *Software Engineering Simulation durch Animierte Modelle*. Ziel war es hier, eine Übungsumgebung für angehende Projektleiter zu entwickeln, um ihnen ein praktischeres Lernen zu ermöglichen. Dazu werden Teile des Softwareentwicklungsprozesses modelliert, und diese Modelle in einem Planspiel verwendet. Der technische Ansatz von SESAM wird in Bezug auf die Ergebnisse dieser Arbeit am Ende der Arbeit in Abschnitt 5.3.1 genauer diskutiert.

Das Spiel SESAM basiert auf dem dort als *QA-Modell* bezeichneten Modell, das vor allem Effekte bezüglich der Qualitätssicherung im Rahmen der Softwareerstellung enthält. Es beschränkt sich jedoch nicht auf diese Aktivität der Softwareentwicklung, sondern unterstützt den gesamten Entwicklungsprozess. Dabei werden unter anderem Aspekte wie Kommunikationsaufwand und Erfahrung der Entwickler berücksichtigt.

SESAM wird dem Spieler über eine grafische Oberfläche zugänglich gemacht. Er kann dort die Informationen, die er für seine Entscheidungen als Projektleiter benötigt, in Form von Textmeldungen abrufen, seinen Teammitgliedern Anweisungen geben und durch den zeitlichen Projektverlauf navigieren. Am Ende seines Projekts übergibt der Spieler das entstandene Produkt dem Kunden und erhält eine Bewertung in Form einer Textnachricht.

SimjavaSP

SimjavaSP [Shaw, 2005] ist ein interaktives webbasiertes Planspiel, das an der University of Tasmania entwickelt wurde. In diesem Spiel schlüpft der Spieler in die Rolle eines Projektmanagers, der ein Team leitet, das eine hypothetische Software entwickelt. Dabei kann der Spieler sowohl das Wasserfallmodell, als auch das Spiralmodell spielerisch kennenlernen. Spielziel ist es, die Software innerhalb einer vorgegebener Zeit und eines bestimmten Budgets in akzeptabler Qualität fertigzustellen.

SimjavaSP basiert auf einer ereignisorientierten Simulation und ist vollständig in Java implementiert. Die Wahl dieser Simulationstechnik begründen die Autoren mit der Tatsache, dass Softwareentwicklungsprozesses in der Regel nicht linear sind und sich kontinuierliche Simulationsmethoden nicht eigenen, um damit zufällige Ereignisse abzubilden.

Die zu bearbeitenden Projekte sind als Menge der notwendigen Aktivitäten modelliert, die benötigt werden um die gefragte Software zu erstellen. Während des Entwicklungsprozesses können verschiedene außergewöhnliche Ereignisse, wie beispielsweise kaputte Hardware, geänderte Anforderungen, Kündigung oder Krankheit auftreten, die das Spiel realistischer machen sollen.

Eine Kombination aus textuellem und grafischem Feedback soll dem Spieler helfen, das Spielgeschehen und die Zusammenhänge zwischen seinen Entscheidungen und deren Auswirkungen auf das Projekt besser zu verstehen.

SimSE

Das Spiel *SimSE* [Navarro, 2006] ist im Rahmen der Dissertation von Emily Navarro entstanden. In ihrer Arbeit hat sie untersucht, ob der Einsatz von Simulationen in der Softwaretechniklehre den gleichen Nutzen bringt, wie er in anderen Bereichen beobachtet werden konnte. Um diese Hypothese zu überprüfen, wurde zunächst eine interaktive, grafische Simulationsumgebung erstellt. Passend zu dieser Umgebung wurden Modelle erstellt, die anschließend in einer Studie verwendet wurden. Die so entstandene Simulationsumgebung stellt in Verbindung mit den entwickelten Modellen ein vollwertiges Planspiel dar. SimSE stellt darüber hinaus Werkzeuge bereit, um eigene Modelle zu entwickeln. Die technischen Aspekte dieses Ansatzes werden in Abschnitt 5.3.2 in Bezug auf die Ergebnisse dieser Arbeit diskutiert.

Ähnlich wie bei den anderen hier vorgestellten Planspielen, übernimmt der Spieler die Rolle des Projektleiters, mit dem Ziel, ein Softwareprojekt oder Teile davon erfolgreich durchzuführen. Ihm steht dabei ein Mitarbeiterteam zur Verfügung, dem er vordefinierte Anweisungen geben kann, die dann von den einzelnen Mitarbeitern ausgeführt werden. Ausgangssituation, mögliche Anweisungen und deren Auswirkungen sind in den einzelnen Modellen definiert.

SimSE ist das einzige Planspiel, welches im Rahmen der Literaturrecherche gefunden werden konnte, das mehr als ein Szenario bzw. Modell enthält. Mit SimSE lassen sich damit unterschiedliche Situationen durchspielen. Enthalten sind sechs Modelle:

- Das **Waterfall Model** zum Erlernen des dort definierten Vorgehens,
- das **Inspection Model**, mit dem die Spieler Codeinspektionen planen und durchführen können,
- das **Incremental Model**, mit dem ein iterativer Entwicklungsprozess erlernt werden kann,
- das **Extreme Programming Model**, das einige der Aspekte des gleichnamigen agilen Vorgehens enthält,
- das **Rapid Prototyping Model**, das ähnlich wie Waterfall Model und Inspection Model aufgebaut ist, jedoch einige Aspekte, wie beispielsweise das Budget, ignoriert, und sich auf wesentliche Aspekte der eigentlichen Softwareerstellung konzentriert,
- und das **Rational Unified Process Model**, welches das RUP-Vorgehen realisiert.

Zu allen Modellen sind die grundlegende Idee, sowie die wesentlichen Modellannahmen in [Navarro, 2006] beschrieben. Eine Dokumentation des Modellaufbaus, insbesondere der verwendeten Formeln und Parameter, ist nicht vorhanden. Der Nutzer muss sich somit darauf verlassen, dass die dort modellierten Zusammenhänge richtig sind und sich das Modell entsprechend korrekt verhält. Eine Validierung der Modelle wird in der Arbeit ebenfalls nicht beschrieben.

SimVBSE

Mit Hilfe des Spiels *SimVBSE* [Jain, 2006] der University of Southern California wird versucht, die Prinzipien des Value-Based-Software-Engineering und die dahinterstehende Theorie besser verständlich zu machen.

SimVBSE setzt dabei auf direktes Feedback nach den einzelnen Spieleraktionen, um dadurch bessere Lerneffekte zu erzielen. Diese basieren primär darauf, dem Spieler die Zusammenhänge anhand von Beispielen geschickt darzustellen, um die verschiedenen Aspekte des Value-Based-Software-Engineering zu verdeutlichen.

Das Spielerlebnis ist auf sieben verschiedene Räume aufgeteilt, beispielsweise dem *CEO's Briefing Room*, um dem Spieler die Aufgabenstellung mitzuteilen, oder die *Lounge*, um die Kunden zu treffen. Der Spieler hat in den verschiedenen Räumen unterschiedliche Aktionsmöglichkeiten, die einerseits den Projektverlauf beeinflussen, andererseits häufig mit direkten oder indirekten Kosten verbunden sind, die es zu berücksichtigen gilt.

Eine Besonderheit stellt der *Situation Room* dar, in dem, unabhängig vom eigentlichen Projekt, unterschiedliche Szenarien mit den dazu passenden möglichen Managemententscheidungen dargestellt werden, aus denen der Spieler wählen kann. Das Ergebnis und die Bewertung seiner Entscheidung wird dem Spieler anschließend direkt dargestellt. Außerdem gibt es jeweils einen Verweis zum entsprechenden Lerninhalt.

Das ganze Spiel basiert auf einem regelbasierten Simulationskern, der den Spielverlauf mittels

mehrerer Ursache-Wirkung-Regeln simuliert. Die einzelnen Spieleraktionen wirken sich hauptsächlich auf die Parameter der Regeln aus und beeinflussen so indirekt den Projektverlauf.

Laut [Jain, 2006] wird SimVBSE von den Autoren außerdem dazu genutzt, allgemein Hypothesen zur VBSE anhand verschiedener Fallstudien zu untersuchen.

The Incredible Manager

Auch in *The Incredible Manager* [de Oliveira Barros, 2006] übernimmt der Spieler die Rolle eines Projektmanagers, der für Planung, Ausführung und Controlling eines Projekts verantwortlich ist.

Zu Beginn stellt der Spieler dazu unter Berücksichtigung von Kosten und Zeitrahmen einen Projektplan auf. Dabei trifft er auch diverse Entscheidungen bezüglich des Entwicklungsteams, Aufwände für Qualitätssicherung und Risikomanagement und weiteren projektrelevanten Aspekten. Anschließend kann er das Projekt ausführen. Falls notwendig, kann er während des Projektverlaufs korrigierend eingreifen. Rückmeldung bekommt der Spieler über eine grafische Darstellung der relevanten Parameter, wie beispielsweise der Kundenzufriedenheit.

Aus technischer Sicht, ist das Spiel in die drei Komponenten *Simulationsmodell*, *Simulationsengine* und *Grafikmaschine* strukturiert. Das Simulationsmodell enthält die Simulationsvorschriften in Form eines System Dynamics Modells. Dazu wurden die in [Abdel-Hamid, 1991] beschriebenen Modelle aufgegriffen und erweitert. Das Modell kann mittels der Simulationsengine ausgeführt werden. Die Grafikmaschine übersetzt den Simulationsverlauf in eine multimediale Repräsentation für den Spieler.

Basierend auf ihrer Erfahrung mit *The Incredible Manager* schlagen die Autoren vor, Planspiele in drei einzelnen Schichten zu entwickeln, die aufeinander aufbauen. In der untersten Schicht befindet sich das Simulationsmodell, das für die Berechnung des Projektverlaufs zuständig ist, darauf aufsetzend befindet sich das Storymodell, das die mathematische Repräsentation des Simulationsmodells auf die eigentlichen Spielelemente wie beispielsweise Artefakte oder Entwickler abbildet. Zu oberst liegt das Grafikmodell, das die Elemente des Storymodells anzeigbar macht, und dem Spieler über ein entsprechendes Interface eine Interaktion mit diesen Elementen ermöglicht.

MO-SEProcess

Basierend auf den Ergebnissen von SimSE, wurde mit *MO-SEProcess* [Ye, 2007] der Versuch unternommen, mit *Second Life* eine bestehende Spielumgebung als Rahmenanwendung für ein Softwaretechnikplanspiel zu verwenden. Second Life stellt eine Simulation der realen Welt in Form eines sogenannten Massiven-Mehrspieler-Online-Spiels bereit, in der virtuelle Charakter leben und interagieren können.

Für MO-SEProcess wurde das Waterfall Model aus SimSE mit Hilfe der von Second Life bereitgestellten Werkzeuge nachgebaut und in Second Life integriert. Die Spieler übernehmen dabei die einzelnen Charaktere des SimSE-Szenarios.

1.2.2. Diskussion

Bei vielen der betrachteten Projekte steht der didaktische Aspekt deutlich im Vordergrund. Entsprechend einfach sind daher meist die hinter den Spielen liegenden Simulationsmodelle, die nur soweit Details enthalten, dass der benötigte Spielablauf für den jeweiligen Lehrzweck plausibel dargestellt werden kann. Eigenschaften, die außerhalb dieses fokussierten Lehrinhalts stehen, werden in der Regel nicht berücksichtigt. Dieses Defizit macht sich nach eigener Erfahrung beim Spielen bemerkbar. So ist die Reaktion der Spiele auf die Entscheidungen des Spielers häufig zu trivial und vorhersehbar, um ein realistisches Spielerlebnis zu schaffen. Durch ein detaillierteres Simulationsmodell könnten diese Spiele verbessert werden.

Bei der Entwicklung der Spiele ist häufig viel Aufwand in die Präsentation der Spielinhalte, die Interaktionsmöglichkeiten des Spielers mit dem Spiel und das didaktische Begleitkonzept geflossen. Dadurch sind für die Lehre wertvolle Planspiele entstanden, die auch erfolgreich eingesetzt werden können.

Scheinbar vergleichsweise wenig Aufwand wurde in die Modelle investiert, die für die Simulation des Spielgeschehens verantwortlich sind. Es finden sich bei den einzelnen Projekten meist kaum

Angaben dazu, welche konkreten Regeln in den Modellen umgesetzt wurden, und wie diese zustande gekommen sind. Es fehlen auch entsprechende Hinweise auf die Validierung der Simulationsmodelle und somit der Nachweis, dass diese die Realität adäquat abbilden. Eine Ausnahme bildet hier die Arbeit [Drappa, 2000a], welche sowohl die Entwicklung, als auch die Validierung des im Kontext von SESAM entwickelten QA-Modells beschreibt.

Mit *SESAM* und *SimSE* unterstützen zwei der sieben untersuchten Projekte die Entwicklung eigener Planspiele. Solche Baukästen habe das Potential den Entwicklungsprozess von Planspielen zu beschleunigen und zu vereinfachen. Sie sind daher für diese Arbeit besonders interessant und werden in Abschnitt 5.3 im Kontext der Ergebnisse dieser Arbeit gesondert diskutiert. Ein Nachteil solcher Baukästen ist beispielsweise ihre Limitierung auf die darin vorgesehenen Konzepte. So ist beispielsweise nur das vorgesehene Benutzerinterface nutzbar und neue Interaktionskonzepte sind nicht umsetzbar. Die Baukästen sehen es auch nicht vor, neue Spielkonzepte einzuführen. So ist es nicht möglich damit Spiele für andere Plattformen, wie Smartphones oder Webbrowser zu entwickeln, oder Mehrspielerszenarien umzusetzen.

Diese Limitierung beschränkt sich auf die Werkzeuge selbst. Die dahinterstehenden Konzepte zur Erstellung der Simulationsmodelle werden davon zunächst nicht berührt. So war es beispielsweise möglich, die Modelle von SimSE ohne die Nutzung des SimSE-Baukastens in Second Life zu integrieren.

Keine der betrachteten Arbeiten liefert ein Simulationsmodell, das darauf ausgelegt ist, in neuen Planspielen verwendet zu werden. Es werden lediglich Hilfestellungen angeboten, ein eigenes Modell zu entwickeln. Für die Entwicklung neuer Planspiele wäre es hilfreich, ein einsatzbereites Modell zu haben, das entweder direkt, in einer veränderten Form, oder zumindest als Referenz verwendet werden kann. Diese Arbeit schlägt ein solches Modell vor. Dazu ist neben einer entsprechenden Dokumentation des Modells auch eine daraufhin ausgelegte Architektur notwendig. Ein weiteres Kriterium ist, dass sowohl Modell als auch Ausführungsumgebung gepflegt und auf einem aktuellen technischen Stand gehalten werden.

Diese Arbeit unterscheidet sich von den hier betrachteten Arbeiten. Während bestehende Arbeiten sich damit befasst haben, wie man das Erstellen von Modellen unterstützen und vereinfachen kann und wie sich didaktische Konzepte mit Planspielen umsetzen lassen, beschäftigt sich diese Arbeit vor allem mit der Umsetzung eines konkreten Simulationsmodells. Dieser Unterschied zeigt sich auch im unterschiedlichen Evaluationsansatz der Arbeiten. Während sich die Evaluation zu den in diesem Abschnitt diskutierten Planspielen fast ausschließlich auf die didaktischen Aspekte, Lernerfolge und die Nützlichkeit von Planspielen in der Lehre im Allgemeinen beziehen, konzentriert sich die Evaluation dieser Arbeit vor allem darauf, zu zeigen, dass das entwickelte Simulationsmodell die Realität adäquat abbildet.

1.3. Motivation der Arbeit

Obwohl der Einsatz von Planspielen die Lehre verbessern kann, werden solche Spiele im Bereich der Softwaretechnik im Zuge der Informatikausbildung eher selten eingesetzt. Ein wesentlicher Grund dafür ist mitunter der hohe Aufwand ein solches Spiel für den eigenen Lehrbetrieb zu erstellen [Hunecker, 2009]. Dazu gehört die Entwicklung des didaktischen Konzepts, das Erstellen der Spielmaterialien, wie Texte, Grafiken, Animation, etc., die Umsetzung der Benutzerschnittstelle und vor allem das hinter dem Spiel stehende Simulationsmodell, mit dem die Abläufe des Spiels berechnet werden.

Während der Entwicklung eines neuen Planspiels müssen eine Reihe wichtiger Entscheidungen getroffen werden. Dazu gehören beispielsweise, welches Ziel das Spiel verfolgt, wie die Spielmechanik und Storyline aussehen sollen, welche Entscheidungsmöglichkeiten der Spieler hat, und wie die Spielinhalte geschickt dargestellt werden.

Eine der wesentlichsten Entscheidungen dabei ist, wie die Vorgänge im Spiel simuliert werden. Die Simulation, die den Kern jedes Planspiels bildet, bestimmt erheblich, wie gut das Spiel die Realität und die zu lernende Situation abbildet, und hat somit einen großen Einfluss auf den Lerneffekt. Dabei stehen dem Entwickler zwei grundsätzliche Möglichkeiten zur Verfügung. Er kann selbst eine auf das Spiel zugeschnittenes Simulation entwickeln, oder eine schon vorhandene Simulation für seine Zwecke anpassen.

Wie der Umfang dieser Arbeit zeigt, ist eine Neuentwicklung eines Planspiels, insbesondere des damit

verbundenen Simulationsmodells, mit erheblichem Aufwand verbunden. Alleine die Einarbeitung in die teils fachfremden Gebiete, wie beispielsweise Psychologie oder Arbeitswissenschaften, kann viel Zeit in Anspruch nehmen. Trotzdem ist es notwendig sich mit diesen Bereichen auseinander zu setzen, da sie einige wichtige Aspekte, wie beispielsweise das Lernverhalten oder die Auswirkungen unterschiedlicher Führungsstrategien behandeln, die für viele Planspiele relevant sind. Dazu kommt die eigentliche Konzeption und Implementierung des Simulationsmodells, das Abstimmen der einzelnen Parameter aufeinander und entsprechende Schritte zur Qualitätssicherung, um sicherzustellen, dass das geschaffene Modell auch tatsächlich das gewünschte Verhalten aufweist.

Die Untersuchung existierender Planspiele im Vorfeld dieser Arbeit hat gezeigt, dass eine Entwicklung auf Basis eines bestehenden Projekts ebenso schwierig ist. Existierende Planspiele beschränken sich häufig auf einen bestimmten Aspekt der Softwareentwicklung und berücksichtigen die anderen Aspekte nur bedingt oder gar nicht. Passt der neue Lerninhalt nicht zufällig zu einer der vorhandenen Lösungen, so ist man gezwungen, zumindest Anpassungen an der bestehenden Software vorzunehmen. Das ist bei den meisten Modellen, wenn überhaupt, dann nur in geringem Umfang vorgesehen und selten ausreichend dokumentiert. In vielen Fällen reicht jedoch auch das nicht aus, und man ist gezwungen, ein eigenes Simulationsmodell zu entwickeln, das die gewünschten Zusammenhänge abbildet. Ein solcher Zusammenhang wird im Folgenden als *Effekt* bezeichnet.

Def.: Ein **Effekt** beschreibt den quantitativen oder qualitativen Zusammenhang zwischen einzelnen Merkmalen der Simulation unter bestimmten Bedingungen; zu konkreten Zeitpunkten im Simulationsverlauf oder über den zeitlichen Verlauf hinweg.

Ein möglicher Effekt ist beispielsweise die Aussage: „Bearbeitet ein Entwickler ein Artefakt, wächst sein Wissen über dieses Artefakt.“ Es sind auch komplexere Effekte denkbar: „Eine hohe Kopplung zwischen den Artefakten des Produkts führt zu einer niedrigeren Qualität bei dessen Erstellung.“

In der Literatur findet sich eine Vielzahl an Aussagen über Effekte im Bereich der Softwareentwicklung, wobei nur die wenigsten davon quantifiziert und mathematisch exakt beschrieben sind. Bei der Entwicklung eines entsprechenden Modells, das diese Effekte abbildet, müssen diese Effekte in der Regel zuerst quantifiziert werden.

Wird ein Modell nur unter Berücksichtigung ausgewählter Effekte entwickelt, fehlen in der Regel die nicht berücksichtigten Aspekte. Diese können aber in der Realität durchaus Auswirkungen auf den Projektverlauf haben. Komplexe Zusammenhänge werden dann oft nicht adäquat abgebildet. Im schlimmsten Fall werden unbeabsichtigt falsche Effekt erzeugt. Werden unvollständige oder unzulängliche Modelle in der Lehre eingesetzt, und werden die Studierenden nicht ausreichend auf dieses Defizit hingewiesen, besteht die Gefahr, dass falsche Zusammenhänge gelernt, und ein damit verbundenes Fehlverhalten angewöhnt wird.

Einer der komplexesten und daher am schwierigsten zu modellierender Aspekt ist das menschliche Verhalten im täglichen Arbeitsleben. Da dies ein schwer zu erfassender Aspekt ist, wird er von vielen Planspielen nur sehr oberflächlich oder gar nicht berücksichtigt. Zumindest im Bereich des Projektmanagements, aber auch allgemein im Gesamtkontext der Softwareentwicklung, sind Mitarbeiter, ihr Verhalten und der richtige Umgang damit, wesentlich für den Projekterfolg mitverantwortlich. Die Verwendung eines bestehenden Modells für ein Planspiel mit dem Ziel Projektmanagementaspekte zu vermitteln scheitert daher oft schon am Fehlen eines Modells, das diese Verhaltensaspekte der Projektbeteiligten ausreichend unterstützt. So kann es beispielsweise einen Unterschied machen, ob die Mitarbeiter ihre Aufgaben selbst bestimmen dürfen, wie es bei agilen Ansätzen oft der Fall ist, oder ob der Projektleiter die Aufgaben streng den einzelnen Mitarbeitern zuweist.

Im Rahmen der Vorarbeiten zu dieser Arbeit konnten einige Aspekte für die Schwierigkeiten bei der Entwicklung eines Simulationsmodells im Softwaretechnikbereich identifiziert werden.

Fehlende Datenbasis Häufig liegen keine genauen und quantitativen Daten über die zu modellierenden Aspekte vor. Softwareprojekte sind häufig sehr individuell und daher schwer vergleichbar. Es ist somit schwierig, eine Datenbasis über mehrere Projekte hinweg zu erstellen, um allgemeine Aussagen über Softwareprojekte treffen zu können. Dazu kommt, dass die Projekte typischerweise über einen zumindest mehrmonatigen Zeitraum und mit einer zum Teil erheblichen Anzahl an Beteiligten

stattfinden. Eine konsequente Erfassung der Daten über die Projektlaufzeit ist daher mit einem hohen Aufwand und entsprechenden Kosten verbunden.

Die realistische Simulation der menschlichen Aspekte in einem Projekt stellt ebenfalls eine große Herausforderung dar. Das Verhalten der Projektbeteiligten wird von einer Vielzahl von Faktoren beeinflusst, die noch nicht vollständig erforscht, und deren Zusammenhänge oft nicht exakt bekannt sind. Das liegt mitunter daran, dass sich Aspekte, wie beispielsweise Müdigkeit oder Motivation nur sehr schwer messen lassen. Entsprechend schwierig ist eine adäquate Abbildung dieser nur eingeschränkt bekannten Zusammenhänge in einem Simulationsmodell.

Einarbeitungsaufwand Da bei der Projektarbeit die unterschiedlichsten Faktoren aus vielen Teilbereichen eine Rolle spielen, erfordert die Entwicklung und Bewertung eines umfassenden Simulationsmodells eine entsprechend hohe Einarbeitungszeit in die Grundlagen der einzelnen Disziplinen. Neben der Softwaretechnik selbst, sind häufig vor allem Aspekte aus den Bereichen der Arbeitspsychologie, Gruppendynamik und Sozialpsychologie, Kognitionspsychologie und Lerntheorie, sowie Ökonomie und Arbeitswissenschaften relevant.

Komplexität Einzelnen Effekte sind in der Realität oft nicht unabhängig voneinander zu beobachten. Die gesammelten Daten zu einem Effekt sind dann auf das Zusammenspiel unterschiedlicher Mechanismen zurückzuführen. Modelle auf Basis solcher Daten müssen sorgfältig erstellt werden. Wird beispielsweise eine Aussage über den Lernprozess getroffen, indem während des Lernvorgangs der Lernfortschritt beobachtet und gemessen wird, beinhalten die gesammelten Daten nicht nur das gelernte Wissen, sondern auch das Wissen, welches der Proband während der Zeit des Lernens wieder vergessen hat. Wird diese Aussage auf naive Weise mit einer zweiten Aussage über das Vergessen von Wissen kombiniert, die auf vom Lernen unabhängigen Daten beruht, wird der Vergessensprozess doppelt modelliert, was dazu führt, dass die Mitarbeiter in der Simulation weniger schnell lernen als in der Realität.

Effekte können untereinander in gewünschte oder unerwünschte Wechselwirkung treten. Überschneiden sich zwei oder mehr Effekte, die gleichzeitig auftreten, können dabei neue, zum Teil falsche Effekte entstehen. Es besteht auch die Möglichkeit, dass sich überschneidende Effekte gegenseitig teilweise oder vollständig aufheben. Während einzelne klar umrissene Effekte noch relativ einfach zu modellieren sind, ist es deutlich schwieriger eine Vielzahl gleichzeitig auftretender und sich gegenseitig beeinflussender Effekte zu beherrschen.

Umfang Die Effekte, die bei der Erstellung von Software im Rahmen eines Projekts auftreten, sind zahlreich, wie schon die Fülle an relevanten Disziplinen vermuten lässt. Daher ist es quasi unmöglich alle Effekte in einem Modell adäquat abzudecken. Die Herausforderung besteht darin zu entscheiden, welche Effekte wichtig sind und welche Effekte gefahrlos ausgelassen werden können.

Selbst wenn nur die notwendigen Effekte berücksichtigt werden, erreichen die Modelle in vielen Fällen trotzdem einen erheblichen Umfang.

Tuning Die einzelnen Aspekte des Simulationsmodells müssen gut aufeinander abgestimmt werden. Dabei ist zum einen die Nähe zur Realität, zum anderen aber auch der eigentliche didaktische Zweck des Modells relevant. Das Ziel ist, dass das Modell am Ende alle Effekte aufweist, die für seinen Einsatz benötigt werden. Da für die beobachteten Effekte in der Regel nur qualitative Aussagen vorliegen, besteht ein Großteil des Aufwands beim Modelltuning darin, die einzelnen Parameter des Modells quantitativ so aufeinander abzustimmen, dass die gewünschten Effekte erzeugt werden. Je nach Komplexität des Modells ist dieser Vorgang sehr aufwändig.

Validierung Bevor das entwickelte Modell eingesetzt werden kann, muss sichergestellt werden, dass es die geforderten Effekte gegenüber der Realität korrekt abbildet. Simulationsmodelle, die für Planspiele eingesetzt werden, haben typischerweise eine Vielzahl an Parametern, die das Verhalten des Modells beeinflussen. Ziel ist es, dass alle Effekte unter jeder möglichen Parameterbelegung und jeder möglichen Parameteränderung während des zeitlichen Verlaufs korrekt funktionieren. Das ist aufgrund der quasi unendlichen Menge an Belegungen und Verläufen schwierig.

1.4. Ziele, Abgrenzung und Ansatz der Arbeit

Das übergeordnete Ziel dieser Arbeit ist es, die Softwaretechnikausbildung durch den Einsatz von Planspielen zu verbessern, indem Möglichkeiten bereitgestellt werden, praktische Erfahrungen in einem geschützten Raum zu machen. Um das zu erreichen, soll die Entwicklung neuer Planspiele unterstützt werden. Ziel dabei ist es, Schwierigkeit, Aufwand und Entwicklungszeit bei der Erstellung simulationsgestützten Lehrmaterials zu verringern. Dazu sollen die Entwickler auf ein bestehendes Basismodell zurückgreifen können, welches nur noch an die jeweiligen Zwecke angepasst werden muss, statt immer neue Modelle entwickeln zu müssen.

Ziele bei der Erstellung des Simulationsmodells

Diese Arbeit beschreibt die Konzeption und Entwicklung eines solchen Basismodells, das als Grundlage für die Entwicklung neuer Planspiele im Softwaretechnik und Projektmanagementbereich dienen kann, indem es einen Kern für die dort benötigten Simulationsmodelle bereitstellt. Auf Basis der oben beschriebenen Motivation werden für dieses Modell folgende Anforderungen definiert:

G1 Das Modell ist so allgemein, dass es innerhalb des Softwaretechnik- und Projektmanagement-kontexts unabhängig von der Art und Gestaltung des Planspiels und dessen didaktischen Zielen möglichst universell eingesetzt werden kann.

G2 Das Modell unterstützt die Darstellung menschlicher Aspekte und menschlichen Verhaltens, um die Entwickler, als die wichtigste Projektressource, adäquat und detailliert darstellen zu können.

G3 Mit dem Modell lassen sich möglichst viele Aspekte der Softwareentwicklung ohne aufwändige Anpassungen abdecken.

G4 Das Modell ist anpassbar und erweiterbar. Es lässt sich dadurch weiterentwickeln und flexibel an neue Erkenntnisse und die konkreten Bedürfnisse der Spielentwickler anpassen.

G5 Das Modell unterstützt eine vollständige Reproduktion des Simulationsverlaufs.

G6 Der Zeitaufwand für das Tuning der Simulationsparameter soll gering sein.

Abgrenzung

Da der Rahmen dieser Arbeit beschränkt ist, können nur ausgewählte Aspekte abgedeckt werden. Deswegen mussten folgenden Einschränkungen getroffen werden:

- Ziel dieser Arbeit ist es die Lehre zu unterstützen. Dazu wird entsprechend von der Realität abstrahiert und nur der für die Lehre relevante Teil der Realität abgebildet. Daher ist das entstandene Simulationsmodell nicht in der Lage Vorhersagen für reale Projekte zu treffen. Die Simulation liefert stattdessen einen zufälligen, der vielen möglichen Projektverläufe, die innerhalb des gewählten Szenarios plausibel sind.

- Die modellierten Effekte liefern primär qualitative Aussagen und sind in der Regel quantitativ nicht belastbar. So lässt sich beispielsweise anhand des Modells feststellen, dass Mitarbeiter durch gewisse Projekttätigkeiten ihre Fähigkeiten um 30% gesteigert haben. Dabei ist die Aussage, dass sich die Fähigkeiten verbessert haben belastbar, jedoch nicht unbedingt die Aussage, dass es sich um genau 30% Steigerung handelt.

  Zum Teil werden Effekte in Planspielen auch absichtlich überzeichnet, um diese besser zu verdeutlichen. So kann es beispielsweise sinnvoll sein, dass die Mitarbeiter im Planspiel ihre Fähigkeiten schneller steigern, als in der Realität. Damit wird die Auswirkungen dieser Steigerung klarer sichtbar und der damit intendierte Lerneffekt verstärkt.

- Das Modell beschränkt sich auf Wissensarbeit. Unter Wissensarbeit wird das Erschaffen und die Weiterverarbeitung von Informationen verstanden (vgl. [DeMarco, 2002]). Sie grenzt sich ganz klar von der Erstellung und Bearbeitung von materiellen Dingen ab. Die Entwicklung von Software fällt unter Wissensarbeit. Etwaige materielle Dinge im Rahmen der Softwareentwicklung sind lediglich Träger der geschaffenen Informationen.

- Das Modell beschränkt sich auf die internen Arbeitsabläufe in einem Projekt. Äußere Einflüsse auf das Projekt, wie beispielsweise die Arbeitsumgebung, firmenpolitische Entscheidungen, Leitbilder, etc. werden weitgehend nicht berücksichtigt. Es beschränkt sich außerdem auf die reine Entwicklungsphase von Software. Auslieferung und Wartung werden im Allgemeinen nicht behandelt.

- Das Modell behandelt die einzelnen Aufgaben bei der Softwareentwicklung auf einer abstrakten Ebene. Die Simulation konkreter Aktivitäten, wie beispielsweise die Codierung wird dabei lediglich als die Bearbeitung eines Artefakts gesehen. Es wird somit zum Beispiel kein konkreter Code erzeugt, wenn ein Entwickler eine Codierungsaktivität vornimmt, sondern lediglich in abstrakten Werten erfasst, wie viel Code in welche Qualität er geschrieben hat.

- Das Modell selbst ist nicht für den direkten Einsatz zu didaktischen Zwecken konzipiert und bleibt dem Spieler verborgen. Es ist vielmehr dazu gedacht, didaktische Modelle auf dessen Basis zu entwickeln, oder diese mit Daten aus dem Simulationsmodell zu füllen. Diese Arbeit befasst sich nicht mit den didaktischen Aspekten von Planspielen, d. h. wie und in welchen Ausschnitten das Simulationsmodell den Studierenden präsentiert wird, um konkrete Lernziele zu erreichen.

Arbeitsansatz und Forschungsfragen

Bisherige Ansätze, Simulationsmodelle für Planspiele zu entwickeln, basieren hauptsächlich auf dem Prinzip die gewünschten Effekte explizit zu modellieren, um sie möglichst klar und direkt im Planspiel darstellen zu können. Dabei entsteht die Gefahr der unerwünschten Überlagerung von Effekten, oder die Auslassung von nicht berücksichtigten Effekten, wie in Abschnitt 1.3 dargestellt. Um diese Probleme zu vermeiden, untersucht diese Arbeit einen anderen Ansatz der Modellentwicklung, der darauf basiert, die gewünschten Effekte mittels Emergenz durch die Kombination grundlegender Teilmodelle zu erzeugen.

Ziel dabei ist es, eine große Zahl an Effekten durch einige wenige Teilmodelle zu erzeugen, und gleichzeitig die Wahrscheinlichkeit zu verringern, dass dabei falsche Effekte entstehen. Da das so entstandene Modell unabhängig von konkreten Effekten ist, ist es allgemeiner einsetzbar und die Wiederverwendbarkeit somit entsprechend höher.

Um Mitarbeiter mit individuellen Fähigkeiten und einem eigenständigen Charakter modellieren zu können, werden die Entwickler als eigenständige Individuen simuliert, die unabhängig voneinander ihre eigenen Interessen verfolgen und dabei auch emotionale Aspekte berücksichtigen. Dieses Konzept entspricht der Realität besser, als eine rein rationale Entscheidungsfindung, wie sie in vielen anderen Modellen umgesetzt wurde. Dazu verfolgt diese Arbeit den Ansatz, ein auf verschiedenen Erkenntnissen aus dem Bereich der Psychologie basierendes Handlungsplanungsmodell zu verwenden, das eine plausible Simulation des Mitarbeiterverhaltens ermöglicht, und dadurch zu einem realistischen Spielerlebnis führen kann.

Daraus ergeben sich folgende Fragestellungen, die mit dieser Arbeit beantwortet werden sollen:

RQ1 Wie gut lässt sich ein Simulationsmodell aus elementaren Teilmodellen zusammensetzen, sodass möglichst viele der in der Literatur beschriebenen Effekte durch Emergenz entstehen?

RQ2 Wie gut lässt sich so ein Modell konfigurieren und anpassen, um typische Planspielszenarien im Softwaretechnikbereich zu simulieren?

RQ3 Wie gut erfüllt so ein Modell die Ziele G1–G6?

RQ4 Wo liegen die Grenzen dieses Ansatzes?

1.5. Ergebnisse

In den folgenden Kapiteln wird gezeigt, dass es prinzipiell möglich ist, ein Modell mit dem beschriebenen Ansatz zu entwickeln, und das sich mit einem solchen Modell die gewünschten Effekte erzeugen lassen. Die Herausforderung liegt dabei in der Auswahl der richtigen Teilmodelle, der Spezifikation ihrer Schnittstellen und der Integration zu einem Gesamtmodell.

Simulationsmodell

Es hat sich gezeigt, dass der Ansatz, Effekte durch Emergenz zu erzeugen, einige Vorteile mit sich bringt. Es kann eine Vielzahl an Effekten mit wenigen Teilmodellen erzeugt werden. Das entstandene Simulationsmodell deckt die wesentlichen Aspekte aus Softwaretechnik und Projektmanagement ab. Unerwünschte Wechselwirkungen zwischen den Teilmodellen treten dabei nur selten auf. Es hat sich außerdem gezeigt, dass ein Aufbau des Gesamtmodells aus einzelnen Teilmodellen das Modell beherrschbar und nachvollziehbar macht, obwohl die dort entstehenden Effekte komplex sind.

Das im Rahmen dieser Arbeit entstandene Simulationsmodell ist unabhängig von einem konkreten Planspiel. Es setzt auf einer allgemeine abstrakte Repräsentation von Softwareprojekten auf. Dadurch wird es dem Spielentwickler ermöglicht, eine Vielzahl an unterschiedlichen Situationen, Bedingungen und Abläufen alleine durch Konfiguration des Modells zu erzeugen, ohne das Simulationsmodell selbst verändern oder erweitern zu müssen. Es kann dadurch für eine Vielzahl an unterschiedlichen Planspielen eingesetzt werden. Der für die Konfiguration und Anpassung des Simulationsmodells notwendige Zeitaufwand liegt dabei erheblich unter dem Aufwand der Erstellung eines neuen, speziell für ein bestimmtes Planspiel geschaffenen Modells. Er verschiebt sich damit von der Entwicklung der Simulation, hin zur Gestaltung des Spiels, insbesondere seiner didaktischen Konzepte und deren Umsetzung.

Im Verlauf der Arbeit hat sich gezeigt, dass durch eine geeignete Architektur eine Erweiter- und Anpassbarkeit des Modells gewährleistet werden kann, die es einem Entwickler ermöglicht, fehlende Aspekte im Modell mit geringem Aufwand zu ergänzen. Dies ist notwendig, um eine langfristige Verbesserung und Weiterentwicklung des Simulationsmodells zu ermöglichen.

Agentenbasierter Ansatz und Simulationsberechnung

Für die Berechnung des Simulationsverlaufs wurde der Ansatz eines Multi-Agenten-Systems gewählt. Dieser erlaubt eine Struktur von Modell und zu simulierenden Szenarien, die nahe an der Realität liegt, und damit leicht nachvollziehbar ist. Alle Mitarbeiter werden durch unabhängige Agenten realisiert; das Simulationsverhalten ergibt sich aus deren Zusammenspiel.

Durch eine diskrete Berechnung des Simulationsverlaufs in beliebig großen Zeitschritten, lässt sich der Simulationsverlauf unterbrechen, der Simulationszustand manipulieren und alle Berechnungsschritte reproduzieren, vorausgesetzt, der zu Beginn herrschende Simulationszustand kann wiederhergestellt werden. Um das zu gewährleisten, wurden ein Persistierungskonzept für den Simulationszustand entwickelt.

Sowohl Modell als auch Ausführungsumgebung, sowie die dazugehörigen Werkzeuge wurden in Form eines gemeinsamen Frameworks realisiert, das es dem Spielentwickler ermöglicht, die im Rahmen dieser Arbeit entstandenen Ergebnisse ohne großen Einarbeitungsaufwand für eigene Planspiele zu verwenden.

Simulation menschlichen Verhaltens

Die Simulation menschlichen Verhaltens wird mittels eines eigenen Teilmodells umgesetzt, das auf aktuellen Handlungsplanungsmodellen der Psychologie basiert. Es berücksichtigt nicht nur rationale, sondern auch emotionale Einflüsse auf die Handlungsplanung. Dadurch, und durch eine detaillierte Konfigurierbarkeit der Mitarbeiter, wird eine realistische Darstellung der Projektarbeit ermöglicht, die es dem Spieler erlaubt, Erfahrungen hinsichtlich der Mitarbeiterführung zu machen, die in dieser Weise bislang nicht möglich waren.

Die Multi-Agenten-Architektur unterstützt dabei das individuelle Verhalten der einzelnen Mitarbeiter, die einzeln und voneinander unabhängig konfiguriert und mit unterschiedlichem Verhalten

ausgestattet werden können. Dadurch lassen sich unterschiedliche Charaktere mit ihren Vor- und Nachteilen für das Projekt erzeugen, anhand deren der Spieler den richtigen Einsatz und die Verwaltung seiner Mitarbeiter üben kann.

Grenzen

Das in dieser Arbeit beschrieben Simulationsmodell erzeugt eine Vielzahl an unterschiedlichen Effekten, die im Bereich der Softwareentwicklung und des damit verbundenen Projektmanagements relevant sind. Diese Effekte wurden in 25 Hypothesen formuliert, die im Rahmen der Evaluation am Simulationsmodell bestätigt werden konnten. Es gibt jedoch auch Aspekte, die nicht durch das Modell abgedeckt werden. Einige Modellierungsansätze erlauben es, die gewünschten Effekte direkt umzusetzen. Mit dem hier gewählten Ansatz ist dies meist nicht möglich, da der zu erzeugende Effekt durch das Zusammenspiel der Teilmodelle entstehen muss. Dazu reicht es nicht aus, den Effekt zu beobachten, sondern es muss auch bekannt sein, warum und wie dieser Effekt entsteht. Nur so können die vorhandenen Teilmodelle angepasst, oder dem Gesamtmodell neue Teilmodelle hinzugefügt werden, die den Effekt erzeugen.

Das Modell selbst ist nicht direkt geeignet, um dem Spieler die Zusammenhänge zu erklären. In vielen herkömmlichen Modellen sind die zu lehrenden Zusammenhänge direkt abgebildet und können somit aus dem Modell abgelesen werden. Das im Rahmen dieser Arbeit entwickelte Simulationsmodell erzeugt die Zusammenhänge durch das Zusammenwirken seiner Teilmodelle. Um diese zu erkennen reicht es deshalb nicht aus, das Modell selbst zu betrachten, sondern es muss ausgeführt werden, um die Zusammenhänge anhand des Simulationsverlaufs erkennen zu können.

1.6. Aufbau der Arbeit

Diese Arbeit stellt die Konzeption und Realisierung eines ausführbaren Simulationsmodells als Basis für Planspiele im Bereich der Softwareentwicklung vor. Sie gliedert sich in die dafür notwendigen Teile, die jeweils in den folgenden Kapiteln besprochen werden.

Kapitel 2 beschreibt das durch diese Arbeit entstandene Simulationsmodell. Hier wird der gewählte Simulationsansatz und die Gesamtarchitektur des Simulationsmodells eingeführt. Zuvor wird der Simulations- und Modellbegriff geklärt. Das restliche Kapitel gliedert sich in das Datenmodell, auf dem die Simulation arbeitet, die einzelnen Teilmodelle und deren Kombination zu einem Gesamtmodell, welches die operationale Semantik des Simulationsmodells beschreibt, sowie einen Teil, der sich mit der subjektiven Wahrnehmung der simulierten Projektbeteiligten und deren Handlungsplanung befasst.

Kapitel 3 beschreibt die Evaluation des Simulationsmodells. Es werden unterschiedliche Ansätze und Vorgehensweisen diskutiert und die für diese Arbeit ausgewählte Methode erläutert. Anschließend werden die Ergebnisse der Evaluation vorgestellt und bewertet.

Kapitel 4 zeigt, wie das entstandene Modell verwendet werden kann, um Planspiele für die Softwaretechnik- und Projektmanagementlehre zu entwickeln. Es wird das grundsätzliche Vorgehen erläutert, sowie anhand von Beispielen gezeigt, wie verschiedenen Szenarien umgesetzt werden können.

Kapitel 5 fasst die Arbeit zusammen und diskutiert ihre Ergebnisse. Es liefert außerdem einen Ausblick auf weiterführende Themen, die nicht in dieser Arbeit behandelt werden konnten.

2 Ein Simulationsmodell für Planspiele

Das Simulationsmodell eines Planspiels beinhaltet alle Regeln, die für die Berechnung des Spielverlaufs notwendig sind. Diese Regeln beziehen sich auf bestimmte Daten, die in einer entsprechenden Struktur vorliegen. Ein Modell lässt sich somit in zwei große Bereiche unterteilen: die *Statische Struktur* der Daten, auf denen die Simulation arbeitet, und die *Operationale Semantik*, die beschreibt, wie sich diese Daten über die Zeit entwickeln.

Um die umfangreiche *Operationale Semantik* besser beherrschbar zu machen, wird diese in zwei zunächst voneinander unabhängige Teilmodelle strukturiert.

Die *Tätigkeitsmodelle* enthalten alle Vorschriften zur Berechnung der Veränderungen, die eintreten, wenn ein Mitarbeiter eine konkrete Tätigkeit ausführt, wie beispielsweise das Reden mit einem Kollegen, oder Veränderungen, die ohne das aktive Zutun eines Projektbeteiligten über die Zeit geschehen, wie beispielsweise das Vergessen von Wissen.

Die *Verhaltensmodelle* hingegen bestimmen, welche Tätigkeiten die einzelnen Mitarbeiter zu einem bestimmten Zeitpunkt ausführen.

Tätigkeitsmodelle und Verhaltensmodelle nutzen eine Vielzahl kleinerer Teilmodelle, welche die einzelnen konkreten Aspekte, wie beispielsweise das Erlernen von Fähigkeiten, oder den Einfluss der Tageszeit auf die Arbeitsleistung beinhalten. Diese Modelle werden im Folgenden als *Elementarmodelle* bezeichnet.

Abbildung 2.1 zeigt eine Übersicht über den groben Aufbau des Simulationsmodells. Die Modellteile sind bis auf das Motivationsmodell zustandslos und nutzen die vorhandenen Daten aus dem Datenmodell, um die notwendigen Berechnungen durchzuführen. Das Verhaltensmodell berechnet dabei auf Basis des Projektzustands, wozu auch der Zustand der Mitarbeiter gehört, die Tätigkeiten, welche die Mitarbeiter zum betrachteten Zeitpunkt durchführen, und gibt diese an das Tätigkeitsmodell weiter. Dieses berechnet die Auswirkungen der Tätigkeiten auf den Projektzustand, und führt die entsprechenden Änderungen an den Daten durch, um den neuen Zustand der Simulation zu erzeugen.

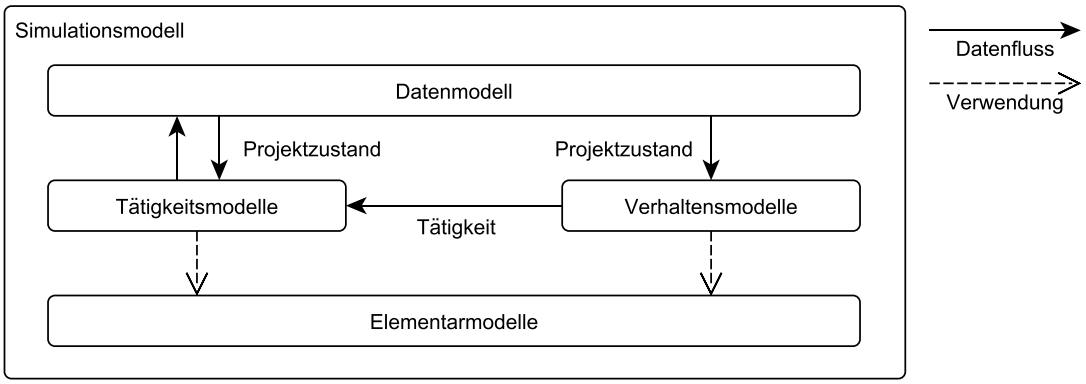


Abbildung 2.1.: Übersicht über den groben Aufbau des Simulationsmodells

Die meisten Elementarmodelle stammen aus Arbeiten der jeweiligen Fachbereiche, und wurden im Rahmen dieser Arbeit an den konkreten Zweck der Planspielsimulation angepasst. Durch eine derartige Strukturierung passt die Architektur des Simulationsmodells zum Ansatz dieser Arbeit, die für die Softwareentwicklung relevanten Effekte durch Emergenz der elementaren Teilaspekte zu erzeugen.

Dieser Aufbau des Simulationsmodells wurde auch unter den Aspekten der Erweiterbarkeit und Wartbarkeit gewählt. Er unterstützt zum einen die Wiederverwendbarkeit der einzelnen Elementarmodelle in unterschiedlichen Bereichen des Gesamtmodells, und zum anderen deren Austauschbarkeit. Das gesamte Simulationsmodell ist nach dem Konzept der Objektorientierung entworfen. Die einzelnen Bestandteile, sowohl der statischen Struktur, als auch der operationalen Semantik, sind in entsprechende Klassen und Methoden unterteilt. Mittels der Spezialisierung von Klassen und dem Überschreiben der dort enthaltenen Methoden lässt sich Struktur und Verhalten des Simulationsmodells individuell auf die konkrete Problemstellung anpassen. Die Elementarmodelle sind als seiteneffektfreie Methoden angelegt und können daher bei Bedarf mit geringem Aufwand angepasst werden. Für einige der durch die Elementarmodelle abgedeckten Aspekte gibt es nach aktuellem Stand der Wissenschaft mehrere, teils sich unterscheidende Aussagen und Ansätze diese zu modellieren. Da die Elementarmodelle weitestgehend voneinander unabhängig sind, lassen sie sich problemlos in unterschiedlichen Varianten umsetzten. Das Simulationsmodell erlaubt es damit, verschiedene Ansätze auszuprobieren und bestehende Umsetzungen durch andere zu ersetzen. Seine Architektur erfüllt somit die notwendigen Voraussetzungen für eine kontinuierliche Weiterentwicklung und Verbesserung.

Def.: Im Folgenden ist mit **Simulationsmodell** das im Rahmen dieser Arbeit entstandene Modell, zur Simulation von Softwareprojekten, für den Einsatz in Planspielen zu Lehre von Softwaretechnik und Projektmanagement gemeint.

Dieses Kapitel stellt die einzelnen Teile des Simulationsmodells vor. Nach einigen grundlegenden Aspekten zu Simulation und Modell, wird der in dieser Arbeit verwendete Simulationsansatz im Detail vorgestellt. Anschließend werden das Datenmodell, die einzelnen Elementarmodelle, deren Kombination zum den Tätigkeitsmodellen und das Verhaltensmodell besprochen.

2.1. Simulation und Modell

Mittels Simulationen kann die Wirklichkeit in einer abstrakten Art und Weise nachgebildet werden [Winter, 2013]. Ziel dabei ist es, eine Umgebung zu schaffen, in der Experimente durchgeführt werden können, die in der Realität zu teuer, zu gefährlich, oder schlicht nicht möglich sind. Simulationen sind vor allem für Probleme bedeutend, für die eine mathematisch-analytische Lösung nicht möglich, oder zu rechenaufwändig ist. Um die Geschehnisse der Wirklichkeit nachzubilden, wird durch Abstraktion ein *Modell* geschaffen, mit dem experimentiert werden kann. Ziel ist es, die so erhaltenen Ergebnisse anschließend zurück auf die Wirklichkeit übertragen zu können. Modelle bilden den Kern jeder Simulation. Sie enthalten die Regeln und Zusammenhänge, die das Simulationsverhalten definieren.

2.1.1. Die drei Hauptmerkmale von Modellen

Stachowiak definiert für Modelle drei Hauptmerkmale [Stachowiak, 1973]: *Abbildungsmerkmal*, *Verkürzungsmerkmal* und *Pragmatisches Merkmal*.

Abbildungsmerkmal

> Modelle sind stets Modelle *von etwas*, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können.
>
> — Herbert Stachowiak [Stachowiak, 1973, S.131]

Bei Modellen unterscheidet man grundsätzlich zwischen Modellen mit *Vorbildcharakter* und Modellen mit *Abbildcharakter*. Modelle mit Vorbildcharakter beschreiben ein noch nicht existierendes Original, und können so beispielsweise eine Vorlage für dessen Realisierung liefern, oder eine Vorschau darauf bieten. Modelle mit Abbildcharakter hingegen beschreiben schon existierende Originale.

Das Planspielmodell hat Abbildcharakter. Es bildet Softwareprojekte aus der Realität ab, und beschreibt damit ein existierendes Original. Entsprechend kann hier auch von einem *deskriptiven Modell* gesprochen werden.

Def.: Jedes Modell bezieht sich auf ein **Original**. Im Rahmen dieser Arbeit ist mit Original das reale Projekt gemeint, welches durch das Planspiel nachgebildet wird. Im Folgenden wird, je nach Kontext, auch die gesamte Klasse der Softwareprojekte, die durch das Simulationsmodell abgedeckt werden, als Original bezeichnet.

Verkürzungsmerkmal

> Modelle erfassen im Allgemeinen *nicht alle* Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant scheinen.
>
> — Herbert Stachowiak [Stachowiak, 1973, S.132]

Ein Modell ist somit eine Abstraktion des Originals, das einen Teil dessen Aspekte nicht enthält. Oft sind nicht alle Aspekte des Originals im Detail bekannt, und lassen sich daher nicht direkt modellieren, sondern müssen ausgelassen oder durch einfachere, ungenauere Repräsentationen ersetzt werden. Dient das Modell dem besseren Verständnis der Realität, können auch bekannte Aspekte des Originals bewusst ausgelassen werden, wenn sie für den Verstehensprozess nicht relevant sind. So kann der Umfang und die Komplexität des Modells reduziert werden, um so das Wesentliche besser herauszustellen.

Beide Punkte treffen auf das Planspielmodell zu. Softwareprojekte sind vielschichtig und komplex. Man kann davon ausgehen, dass eine Vielzahl von Faktoren den Verlauf und Ausgang eines solchen Projekts beeinflussen, von denen bis heute nur ein Teil bekannt ist. Es ist damit unmöglich, diese vollständig durch ein Modell abzubilden. Selbst wenn alle Faktoren bekannt wären, würde deren Zahl und Komplexität zu groß sein, um sie beherrschen zu können, und das Modell dabei verständlich, nachvollziehbar und damit wartbar zu halten. Es ist daher notwendig zu entscheiden, welche Aspekte für den Einsatz des Modells in Planspielen notwendig sind, und welche das Modell lediglich umfangreicher und komplexer machen, ohne einen erkennbaren Nutzen für das Planspiel zu haben.

Pragmatisches Merkmal

> Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion a) für *bestimmte* – erkennende und/oder handelnde, modellbenutzende – *Subjekte*, b) innerhalb *bestimmter Zeitintervalle* und c) unter Einschränkung auf *bestimmte gedankliche oder tatsächliche Operationen*.
>
> — Herbert Stachowiak [Stachowiak, 1973, S.133]

Kurz gesagt, ein Modell wird immer für einen bestimmten Zweck erstellt. Dieser Zweck ist Maßstab dafür, wie das Modell gestaltet wird und welche Aspekte es enthalten muss. Er hat somit unter anderem auch Einfluss auf die Verkürzung des Modells gegenüber dem Original.

Das Planspielmodell soll der Lehre im Bereich der Softwareentwicklung, und somit dem besseren Verständnis der dort relevanten Zusammenhänge dienen. Ziel bei der Modellentwicklung ist es daher, diese Zusammenhänge adäquat durch das Modell nachzubilden und Aspekte, die keinen oder nur geringen Einfluss darauf haben, auszulassen.

2.1.2. Modellgrenze und Schnittstelle zur Umgebung

Der Erfolg eines Projekts hängt von einer Menge unterschiedlicher Faktoren ab. Einige davon sind eng mit der Projektarbeit verknüpft und ihre Relevanz ist schnell ersichtlich. Beispielsweise hat die Erfahrung der Entwickler einen Einfluss auf die Qualität des Endprodukts. Es existieren jedoch auch viele Faktoren, deren Einfluss eher indirekter Natur ist. So kann beispielsweise die familiäre Situation der Mitarbeiter Einfluss auf deren Leistung haben. Die familiäre Situation hängt dabei wiederum von weiteren Faktoren ab.

Dieses Beispiel zeigt, dass es notwendig ist, bei der Entwicklung eines Modells klare Grenzen zu ziehen, um zu definieren, was Teil des Modells ist und was nicht. Dazu müssen zuerst die Grenzen des Originals festgelegt werden. Die Teile, die nicht mehr zum Original gehören werden als *Systemumgebung* bezeichnet, die Grenze zwischen Original und Systemumgebung als *Systemgrenze*.

Es ist in vielen Fällen nicht möglich die Systemgrenze exakt zu definieren. Dadurch entsteht ein unscharfer Übergang zwischen Original und Systemumgebung, dessen Entsprechung auf Modellseite im Folgenden als *Modellumgebung* bezeichnet wird.

Def.: Die **Modellumgebung** bildet alle Aspekte ab, die nicht Teil des Originals sind, dieses jedoch beeinflussen.

Für die Umsetzung dieser Aspekte, stehen zwei grundsätzliche Möglichkeiten zur Verfügung: die Aspekte werden ignoriert und damit im Rahmen der Verkürzung ausgelassen, oder die Aspekte werden durch die Modellumgebung geeignet abstrahiert.

Eine Möglichkeit der Abstraktion ist die Verwendung von Zufallswerten für Aspekte, die zwar in der Systemumgebung nicht zufällig entstehen, jedoch für einen außenstehenden Betrachter diesen Anschein erwecken. Im Rahmen dieser Arbeit wurden zur Abstraktion vor allem konstante Parameter eingesetzt. Diese eigenen sich besonders für Aspekte, die sich während des Zeitraums eines typischen Softwareprojekts nur selten oder nie ändern. Dazu gehören zum Beispiel die familiäre Situation der Mitarbeiter, oder dessen Charakterzüge. Solche Parameter können sich während des Projektverlaufs durch *Ereignisse* ändern.

Def.: Ein **Ereignis** ist eine Veränderung des Zustands bzw. der Konfiguration der Modellumgebung, des Modells, oder des Szenarios zu einem bestimmten Zeitpunkt im Simulationsverlauf. Diese Veränderung ist explizit definiert und nicht Teil des Modells.

Durch Ereignisse können beispielsweise einschneidende Veränderungen während des Projektverlaufs, im Rahmen der Storyline des Planspiels festgelegt werden. Dazu gehören zum Beispiel der Ausfall eines Mitarbeiters, Änderungen an Anforderungen des Produkts, oder die Schulung eines Mitarbeiters. Das Ereignis verändert dabei gezielt und außerhalb der eigentlichen Simulation deren Zustand. Der Zeitpunkt des Eintretens eines Ereignisses, oder dessen Auswirkungen können dabei durchaus vom Zustand oder Verlauf der Simulation abhängen. So kann beispielsweise ein zu einem bestimmten Zeitpunkt ausfallender Mitarbeiter anhand eines Vergleichs der unterschiedlichen Gesundheitszustände aller Mitarbeiter ausgewählt werden, um den schwächsten Mitarbeiter im Team ausfallen zu lassen.

2.1.3. Simulationskonzept

Die konkrete Gestaltung des Modells hängt unter anderem maßgeblich vom Simulationskonzept ab, mit dem das Modell ausgeführt werden soll. Im Bereich der Simulation von Softwareprojekten und bei der Umsetzung von Planspielsimulationen kamen in der Vergangenheit vor allem *System Dynamics*, *Ereignisorientierte Simulation* und *Multi-Agenten-Simulation* zum Einsatz.

Das Konzept der *System Dynamics* wurde von Jay W. Forrester in den 50er Jahren entwickelt [Forrester, 1972], um komplexe dynamische Systeme im volks- und betriebswirtschaftlichen Bereich zu untersuchen. Dabei wird zwischen *qualitativen* und *quantitativen* Modellen unterschieden.

Qualitative Modelle dienen primär dazu, Wirkungsketten, sogenannte *feedback loops*, zu analysieren, die mit positiver oder negativer Polarität auftreten können. Damit beschreiben sie den verstärkenden oder abschwächenden Einfluss der unterschiedlichen Aspekte eines Systems aufeinander.

Quantitative Modelle werden als Flussdiagramme entworfen, und bestehen aus mehreren Zustandsvariablen zwischen denen mit unterschiedlichen Raten abstrakte Werte fließen. Im Gegensatz zu den qualitativen Modellen, können die quantitativen Modelle als Differenzialgleichungssystem formuliert und damit simuliert werden.

Bei *ereignisorientierten Simulationen* wird der Simulationsverlauf anhand des Eintretens von diskreten Ereignissen berechnet. Ein dazugehöriges Modell besteht aus einem Zustandsmodell, einer Liste geplanter Ereignisse und Routinen zur Ausführung dieser Ereignisse. Außerdem wird ein Zeitkonzept benötigt, an dem die Ereignisse ausgerichtet werden können, um sie nach ihrem Eintrittszeitpunkt zu sortieren. Die Berechnung des Simulationsverlaufs erfolgt durch die sequentielle Ausführung der zu den Ereignissen in der Ereignisliste passenden Routinen. Routinen können dabei den Simulationszustand

verändern und die Ereignisliste durch das Hinzufügen oder Entfernen von Ereignissen manipulieren. Durch diese Art der Simulation wird der Zustand der Simulation nur für Zeitpunkte berechnet, an denen Ereignisse eintreten.

Multi-Agenten-Simulationen, basieren auf dem Zusammenspiel mehrerer aktiver Agenten, deren Verhalten individuell definiert ist. Durch das Zusammenspiel und die Wechselwirkung der einzelnen Agenten, die sich in einer gemeinsamen Umgebung befinden, entsteht das Systemverhalten [Jennings, 1998]. Mit diesem Konzept lassen sich vor allem Effekte die durch Emergenz entstehen gut nachbilden. Die Agenten interagieren dabei entweder direkt miteinander, oder indirekt über ihre Umwelt, die sie wahrnehmen und manipulieren können.

Das in dieser Arbeit vorgeschlagene Simulationsmodell basiert auf dem Ansatz der Multi-Agenten-Simulation. Dieser entspricht dem Sachverhalt der Wirklichkeit am besten. Die individuell am Projekt arbeitenden Teammitglieder lassen sich als einzelne individuelle Agenten umsetzen. Wie im Original entsteht so das Projektverhalten auch in der Simulation durch das Zusammenwirken der einzelnen Handlungen der Projektbeteiligten.

Jeder Agent lässt sich dabei individuell gestalten, so können unterschiedliche Handlungsmuster leichter umsetzen werden. Agenten haben grundsätzlich die Möglichkeit Tätigkeiten auszuführen, die zeitlich punktuell sind, oder sich über ein bestimmtes Zeitintervall erstrecken. Die zeitlich punktuellen Tätigkeiten werden durch entsprechende Routinen, ähnlich des Konzepts der Ereignisorientierten Simulation definiert, während die kontinuierlichen Tätigkeiten wie bei System Dynamics durch Differentialgleichungen beschrieben werden.

Im Gegensatz zu System Dynamics Modellen ist der Zustand des Simulationssystems deutlich detaillierter beschrieben und enthält ein Vielfaches der für ein System Dynamics Modell üblichen Variablen. Da die Veränderung dieser Variablen von den individuellen Tätigkeiten der Agenten abhängt, kann diese nicht durch einen Fluss zwischen den Variablen, wie bei System Dynamics üblich, beschrieben werden.

Um einen beobachtbaren zeitlichen Verlauf berechnen zu können, basiert das Simulationsmodell auf einem quantitativen und diskreten Ansatz. Der Spieler kann so nach der Berechnung jeden Zeitschritts die Entwicklung seines Projekts einsehen und falls notwendig steuernd eingreifen.

2.2. Framework

Das Simulationsmodell ist in ein Framework eingebettet, das ebenfalls im Rahmen dieser Arbeit entstanden ist. Dieses Framework unterstützt die Entwicklung und Verwendung des Simulationsmodells. Abbildung 2.2 zeigt die einzelnen Bestandteile des Frameworks in Form eines UML-Paketdiagramms.

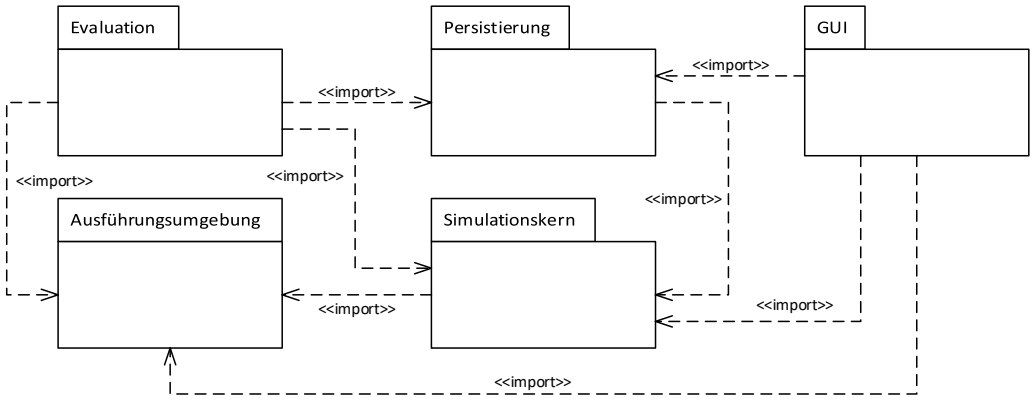


Abbildung 2.2.: UML-Paketdiagramm zur Paketstruktur des Frameworks

- Das Paket **Ausführungsumgebung** enthält die Ausführungsumgebung für Multi-Agenten-Systeme. Es definiert die dafür erforderlichen Klassen und die Ausführungslogik.
- Das Paket **Simulationskern** enthält das Simulationsmodell, bestehend aus Datenmodell, Elementarmodellen, Tätigkeitsmodellen, Verhaltensmodell und allen darin benötigten Hilfsfunktionen und -strukturen.
- Das Paket **Persistierung** enthält die Funktionalität zum Speichern und Laden von Szenarien.
- Das Paket **GUI** enthält die grafische Oberfläche, die verwendet werden kann, um den Zustand der Simulation und deren Verlauf zu beobachten und steuernd einzugreifen.
- Das Paket **Evaluation** enthält Strukturen, Algorithmen und Skripte, die dabei helfen Simulationsläufe automatisiert zu erstellen, auszuführen und auszuwerten, um sowohl das Simulationsmodell, als auch damit erstellte Spiele besser validieren zu können.

Die Ausführungsumgebung beinhaltet die für die Berechnung des Simulationsfortschritts notwendige Steuerstruktur. Ihre Hauptaufgabe ist es, die Agenten so zu steuern, dass die Konsistenz der gesamten Simulation gewährleistet ist, und der zeitliche Rahmen dabei aufrechterhalten bleibt. Dazu gehört zum Beispiel, dass die Zeit einzelner Agenten nicht mehrfach verbraucht wird. So wird beispielsweise verhindert, dass ein Mitarbeiter an einer eigenen Aufgabe arbeitet und gleichzeitig einem Kollegen bei dessen Aufgabe hilft. Außerdem regelt das Steuermodell alle übergeordneten Prozesse, die nicht zur eigentlichen Simulation gehören.

Das Kernstück der Ausführungsumgebung ist der Simulator, mit dem der Verlauf einer Simulation berechnet werden kann. Eine *Simulation* besteht aus einer Menge von Agenten, der Menge von Ressourcen, auf denen die Agenten arbeiten und einer Umgebung, welche die Rahmenparameter der Simulation definiert. Um den Verlauf einer Simulation zu überwachen oder mitzuprotokollieren, können ihr sogenannte *Monitore* zugewiesen werden. Mit ihnen kann die Simulation einfach beobachtet und ausgewertet werden. Abbildung 2.3 zeigt die Klassenstruktur des Pakets Ausführungsumgebung in Form eines UML-Klassendiagramms.

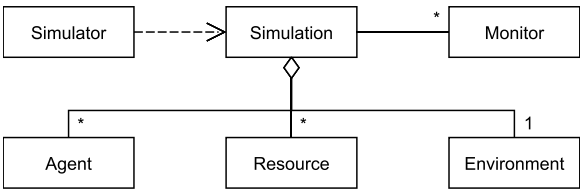


Abbildung 2.3.: Klassenstruktur des Pakets Ausführungsumgebung

Die Berechnung des Simulationsverlaufs erfolgt in diskreten Zeitschritten. Die Länge dieser Schritte ist zunächst frei wählbar. Das in diesem Abschnitt beschriebene Vorgehen wird benötigt, um Simulationsmodell und Agenten unabhängig der Zeitschrittgröße realisieren zu können.

Der Simulator teilt den gesamten Zeitraum der Simulation in diese Zeitschritte auf. Dabei wird sichergestellt, dass jeder Zeitpunkt durch genau ein Zeitintervall abgedeckt wird. Bei der Berechnung des Simulationsverlaufs werden alle Zeitschritte einzeln und in chronologischer Reihenfolge berechnet. So entstehen eine Menge von Zuständen, die den Simulationsverlauf diskret beschreiben. Ein Eingreifen in die Simulation ist nur zwischen der Berechnung der Zeitschritte möglich. Da die Zeitschritte beliebig kurz sein können, stellt dies jedoch keine Einschränkung dar.

Innerhalb eines Zeitschritts führen die Agenten ihre Handlungen aus. Da die Agenten im Original parallel agieren, muss die Länge des Zeitschritts klein genug gewählt werden, um den durch die sequentielle Berechnung entstehenden Fehler möglichst gering zu halten. Je kürzer die Schrittlänge, umso besser wird das parallele kontinuierliche Handeln der Agenten abgebildet. Wie die Agenten, so werden auch die Monitore in jedem Zeitschritt getriggert. Im Gegensatz zu den Agenten dürfen diese den Simulationszustand jedoch nicht verändern.

Das Simulationsmodell ist so konzipiert, dass alle Berechnungen in beliebig viele Zeitschritte aufgeteilt werden können. Da eine wechselseitige Interaktion zwischen den Agenten in einigen Fällen

nur zu Beginn eines Zeitschritts erfolgen kann, hat die Wahl der Schrittlänge durchaus einen Einfluss auf das Simulationsergebnis. Grundsätzlich gilt, je größer die Schrittlänge, umso ungenauer sind die Simulationsergebnisse, je kleiner die Schrittlänge, desto höher ist die benötigte Rechenzeit. Im Laufe der Entwicklung des Simulationsmodells hat sich gezeigt, dass eine Schrittlänge von einer Stunde Simulationszeit ein guter Kompromiss zwischen der Genauigkeit der Ergebnisse und der Performance der Berechnung des Simulationsverlaufs ist.

Da die Handlungen der Agenten nicht zwangsläufig am durch den Simulator definierten Zeitraster ausgerichtet sind, besteht die Möglichkeit, dass eine Handlung mehr oder weniger Zeit benötigt, als durch den Zeitschritt vorgegeben wird. Benötigt der Agent nur einen Teil des Zeitschritts, kann er dies dem Simulator mitteilen, der diesen Zeitschritt dann entsprechend aufteilt.

Das ist sinnvoll, wenn Agenten aufeinander warten müssen. Beispielsweise wollen Agent A und Agent B gemeinsam eine Aufgabe ab 9:00 Uhr bearbeiten. Agent A ist um 9:00 Uhr bereit, während Agent B erst um 9:02 mit einer anderen Aufgabe fertig wird. Anstatt den ganzen Zeitschritt von einer Stunde ungenutzt zu lassen, kann Agent A entscheiden, nur 5 Minuten zu warten, und so gemeinsam mit Agent B ab 9:05 mit der geplanten Aufgabe zu beginnen. Agent A teilt dazu dem Simulator mit, dass er nur 5 Minuten seines Zeitschritts abgearbeitet hat. Der Simulator kann diesen Zeitschritt so in zwei Zeitschritte mit 5 und 55 Minuten aufteilen. Dadurch entsteht ein zusätzlicher Synchronisierungspunkt zwischen den beiden Agenten um 9:05 Uhr.

Folgender Algorithmus in Pseudocode zeigt den Ablauf der Berechnung eines einzelnen Zeitschritts für alle Agenten.

Data : auszuführender Zeitschritt $[t_{Start}, t_{Ende}]$, Menge der beteiligten *Agenten*
1 $t_a := t_{Start}\ \forall a \in Agenten$;
2 *offene Agenten* $:= \{a \in Agenten \mid t_a < t_{Ende}\}$;
3 **while** *offene Agenten* $\neq \emptyset$ **do**
4 wähle zufälligen Agent $a \in$ *offene Agenten* mit $t_a \leq t_{a'}\ \forall a' \in$ *offene Agenten*;
5 führe Zeitschritt $[t_a, t_{Ende}]$ für Agent a aus;
6 $t_a :=$ Zeitpunkt bis zu dem Agent a den Zeitschritt ausgeführt hat;
7 *offene Agenten* $:= \{a \in Agenten \mid t_a < t_{Ende}\}$;
8 **end**

Dieser Ansatz stellt ein alternatives Konzept zur allgemeinen Verkürzung der Schrittlänge dar, und hat den Vorteil, dass Schritte nur verkürzt werden, wenn ein Agent das für notwendig erachtet. Ein Nachteil des Konzepts ist, dass fehlerhafte Agenten den Simulationsverlauf stören können, indem sie keinen oder einen nur geringen zeitlichen Fortschritt ausführen. Um zu verhindern, das einzelne Agenten den Simulationsablauf blockieren, fordert der Simulator ein Minimum an Fortschritt für jeden Zeitschritt. Alle Agenten müssen so implementiert werden, dass sie dieses Minimum immer erfüllen.

Der Ansatz der dynamischen Unterteilung von Zeitschritten weist Ähnlichkeiten zur ereignisorientierten Simulation auf. Da ein gleichmäßiges Zeitraster jedoch einfacher umzusetzen ist und in den meisten Situationen ausreicht, wird eine zusätzliche Unterteilung in Abhängigkeit des Simulationsverlaufs nur in Ausnahmefällen angewendet. Dabei bleibt das Raster der Zeitschritte erhalten und die zusätzliche Unterteilung nach außen verborgen.

Benötigt der Agent mehr Zeit für eine, gegebenenfalls atomare Handlung, kann er den Zeitschritt überziehen. Besonders bei sehr kleinen Zeitschritten kann es passieren, dass eine Handlung mehr Zeit beansprucht, als ihr durch den Zeitschritt zur Verfügung steht. Um diese Handlungen nicht künstlich aufteilen zu müssen, kann das Zeitkonto über die zur Verfügung stehende Zeit hinaus belastet werden. Diese Überbuchung muss im nächsten Zeitschritt ausgeglichen werden, damit die Gesamtzeitbilanz der Simulation stimmt. Dazu besitzt jeder Agent ein Zeitkonto, um eine Mehrfachbelegung seiner Zeit zu verhindern. Dieses Zeitkonto speichert die schon konsumierte Zeit, und definiert damit den frühesten Zeitpunkt, an dem der Agent eine weitere Handlung durchführen kann. Liegt dieser Zeitpunkt hinter dem Beginn t_{Start} eines Zeitschritts, wird t_a im ersten Schritt des oben gezeigten Algorithmus nicht auf den Startzeitpunkt des Zeitschritts gesetzte, sondern auf den Zeitpunkt bis zu dem das Zeitkonto belastet ist. Trotz des Ausgleichs können durch solche Überbuchungen zeitliche Anomalien entstehen, die das Simulationsergebnis verfälschen können. Daher sollten Überbuchungen nur in zeitlich sehr kleinem Rahmen verwendet werden.

Der Fehler der Simulation gegenüber dem Original, der durch die sequentielle Abarbeitung der Agenten entsteht, wird durch entsprechend kleine Zeitschritte so gering, dass er das Simulationsergebnis nicht mehr in relevanter Größe beeinflusst. Sobald einzelne Agenten um gemeinsame Ressourcen konkurrieren, wird durch die Sequenzierung derjenige Agent bevorzugt, der zuerst getriggert wird. Dieser Fehler wird durch eine, in jedem Zeitschritt erneut zufällig berechnete Reihenfolge der Agenten kompensiert.

Das restliche Kapitel beschäftigt sich mit dem *Simulationsmodell*. Die Pakete *Persistierung*, *GUI* und *Evaluation* werden in späteren Kapiteln besprochen.

2.3. Datenmodell

Dieser Abschnitt beschreibt das *Datenmodell* auf dem das Simulationsmodell basiert. Das Datenmodell definiert die *statische Struktur* der Simulation, und damit den Aufbau des Simulationszustands und dessen Schnittstellen zum operationalen Teil der Simulation. Es basiert auf einem objektorientierten Ansatz und kann mittels Verfeinerung an ein konkretes Planspiel angepasst werden. Da dabei die Schnittstellen erhalten bleiben, wird der Rest des Simulationsmodells durch solche Verfeinerungen nicht beeinflusst.

2.3.1. Einheiten und spezielle Datentypen

Das Simulationsmodell arbeitet mit Werten, für die teilweise keine etablierte Metrik existiert, oder diese vom konkreten Einsatzzweck abhängig ist. So lässt sich beispielsweise eine Fähigkeit im Allgemeinen nicht direkt messen, und es existiert keine typische Einheit, in der Fähigkeiten angegeben werden können. Um die korrekte Verwendung dieser Werte, insbesondere an den Schnittstellen der Teilmodelle sicherzustellen, wurden dafür eigene Einheiten definiert. Dieser Abschnitt beschreibt die verwendeten Einheiten und deren Definition. Außerdem werden hier die für die statische Struktur notwendigen speziellen Datentypen und die darauf ausführbaren Operationen beschrieben.

Das Simulationsmodell basiert in einigen Teilen auf den Ergebnissen empirischer Untersuchungen. Die darauf aufgebauten Modelle, wie sie in der entsprechenden Literatur zu finden sind, beziehen sich daher auf einen durchschnittlichen Probanden der jeweils untersuchten Grundmenge. Da das Simulationsmodell auf diesen Modellen aufbaut, wird auch dort meist von einer durchschnittlichen Person als Basisfall ausgegangen. Einige der Einheiten und Parameter beziehen sich deshalb auf einen *durchschnittlichen Projektbeteiligten*. Dieser steht für eine Person, die bezogen auf ihre Eigenschaften, wie beispielsweise Arbeitsleistung oder Fähigkeiten, einer durchschnittlichen Person entspricht, die auf dem jeweiligen Gebiet tätig ist.

Arbeitsumfang In der Softwaretechnik existieren verschiedene Methoden um Aufwände und Umfänge von Arbeitspaketen zu bestimmten und in Zahlen zu fassen. Die Einheit der so ermittelten Werte, wie beispielsweise *Lines of Code (LOC)* oder *Function Points (FP)*, hängt dabei von der gewählten Methode ab. Um das Simulationsmodell nicht von vornherein auf eine solche Methode festzulegen, wird zur Messung von *Arbeitsumfängen* eine neue abstrakte Einheit eingeführt.

Der Umfang von Artefakten, und somit der Arbeitsaufwand diese zu erstellen, wird in *Arbeitseinheiten* mit der Einheit U und Werten aus $[0, \infty]$ angegeben. $1U$ entspricht dabei dem Aufwand, den ein durchschnittlicher Projektbeteiligter innerhalb einer Stunde bewältigen, bzw. dem Umfang, den er in einer Stunde bearbeiten kann.

Arbeitsgeschwindigkeit Die *Arbeitsgeschwindigkeit* gibt an, wie schnell ein bestimmter Arbeitsfortschritt erzielt wird. Sie wird in $^{U}/_{h}$ mit Werten aus dem Bereich $[0, \infty]$ angegeben, und gibt damit an, wie viele Arbeitseinheiten pro Stunde bearbeitet werden.

Fähigkeit Die *Fähigkeit* gibt an, wie gut eine Person Aufgaben eines bestimmten Aufgabengebiets erledigen kann. Sie wird als Wert aus $[0, 1]$ angegeben. Ein Wert von 1 bedeutet dabei, dass die Person die maximal erreichbare Eignung besitzt, und somit alle Aufgaben aus dem Aufgabengebiet optimal

lösen kann. Ein Wert von 0 bedeutet, dass die Person keine Eignung für das Aufgabengebiet besitzt. Die Fähigkeiten einer durchschnittlichen Person liegen bei 0.5. Ein durchschnittlicher Programmierer hat beispielsweise eine Fähigkeit von 0.5 in Bezug auf das Erstellen von Code.

Aufgabenschwierigkeit Die *Schwierigkeit* einer Aufgabe wird, passend zur Fähigkeit, mittels eines Werts aus $[0, 1]$ angegeben. Der Wert gibt an, welche Fähigkeit notwendig ist, um die Aufgabe durchschnittlich gut bearbeiten zu können.

Qualität Die *Qualität* eines Elements wird als Wert aus dem Intervall $[0, 1]$ angegeben. 1 bedeutet dabei, dass das Element eine optimale Qualität aufweist, und damit zum Beispiel keine Fehler enthält. 0 bedeutet, dass das Element aufgrund seiner schlechten Qualität, zum Beispiel durch zu viele Fehler, vollständig unbrauchbar ist. Die Qualität eines durchschnittlichen abgeschlossenen Artefakts entspricht 0.7.

Defekt Ein *Defekt* ist ein Mangel an einem Element, der dessen Qualität herabsetzt. Ein Defekt wird als ein Triple *(Umfang, Schwierigkeit, Typ)* dargestellt. Dabei gibt *Umfang* den Arbeitsumfang an, der notwendig ist den Defekt zu beheben, *Schwierigkeit* legt die Aufgabenschwierigkeit fest, den Defekt zu finden und *Typ* gibt an, um welche Art von Defekt es sich handelt.

Die Defekttypen können frei definiert werden. Welche Defekttypen benötigt werden, hängt typischerweise von den Artefakten und Aktivitäten ab, die im Szenario definiert sind. Werden beispielsweise Defekte im Rahmen der Codierung betrachtet, so können diese beispielsweise in *fehlerhafte Funktionalität*, *Speicherleaks*, *Dokumentationsfehler*, *Sicherheitslücke*, etc. aufgeteilt werden.

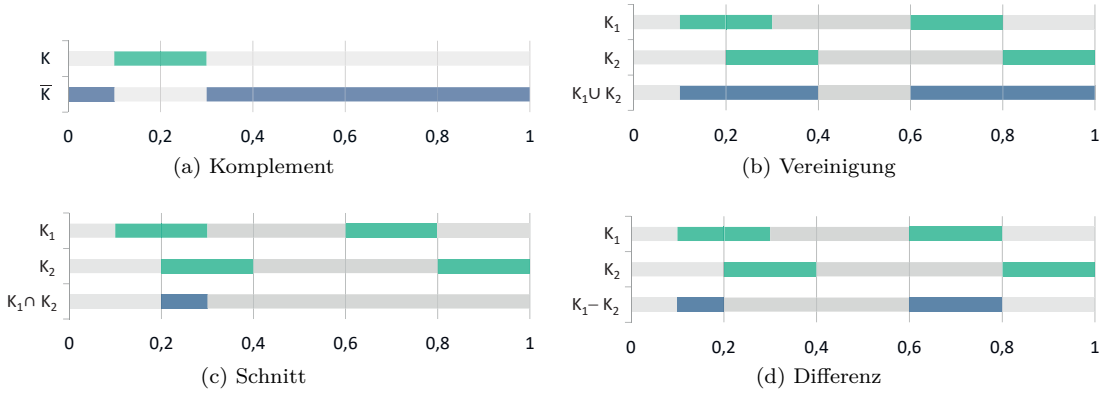
Artefaktwissen Im Gegensatz zu Fähigkeiten, bezieht sich *Artefaktwissen* auf ein klar abgegrenztes Wissensgebiet. Diese entspricht den Informationen, die ein Artefakt des zu entwickelnden Produkts enthält. Der Wissensstand einer Person wird als relativer Anteil zur vollständigen Kenntnis und Verständnisses dieser Informationen angegeben.

Dazu werden die Informationen, welche durch das Artefakt beschrieben sind, auf das Intervall $\mathbb{K} := [0, 1]$ abgebildet. Der Wissensstand wird als Menge disjunkter Teilintervalle $K \subset \mathbb{K}$ dieses Intervalls beschrieben, wobei die Teilintervalle diejenigen Bereiche des Artefakts markieren, welche die betrachtete Person kennt. So kann beispielsweise ausgedrückt werden, dass ein Mitarbeiter das erste und das letzte Drittel eines Anforderungsdokuments kennt, den Mittelteil jedoch nicht.

Für das Artefaktwissen sind folgende Operationen definiert, wobei sich dabei alle Operanden K auf das gleiche Artefakt beziehen müssen.

- Das **Komplement** $\overline{K}$, d. h. die Teilintervalle von $\mathbb{K}$, die nicht durch K abgedeckt sind, beschreibt das fehlende Wissen über das Artefakt.
- Die **Vereinigung** zweier Wissensstände $K_1 \cup K_2$ kann verwendet werden, um Wissen aus zwei Quellen zusammenzufassen. So kann beispielsweise ermittelt werden, über welches Wissen eine Gruppe von Personen insgesamt verfügt.
- Die **Schnittmenge** zweier Wissensstände $K_1 \cap K_2$ kann verwendet werden, um das gemeinsame Wissen zu beschreiben. So kann beispielsweise ermittelt werden, über welches Wissen jeweils alle Personen einer Gruppe verfügen.
- Die **Differenz** zweier Wissensstände $K_1 - K_2$ beschreibt, welches Wissen aus K_1 nicht in K_2 enthalten ist. So kann beispielsweise beschrieben werden, was eine Person von einer anderen noch lernen kann.
- Für ein Artefaktwissen K kann dessen **Umfang** $|K| = \sum_{[k_1,k_2]\in K}(k_2 - k_1)$ bestimmt werden, der angibt, welchen Anteil die bekannten Informationen in Bezug auf die im gesamten Artefakt vorhandenen Informationen abdecken. So kann mittels einer einzigen Zahl ausgedrückt werden, wie viel ein Mitarbeiter über ein Artefakt weiß.

Abbildung 2.4 zeigt eine Visualisierung der vier Operationen. Die potentiell im Artefakt vorhandenen Informationen sind dabei grau markiert, der Wissenstand der Operanden wurde grün eingefärbt und das Ergebnis der Operation ist blau dargestellt.



(a) Komplement (b) Vereinigung

(c) Schnitt (d) Differenz

Abbildung 2.4.: Operationen auf Artefaktwissen

2.3.2. Statische Struktur

Die *statische Struktur* definiert den Aufbau der Daten, die den Simulationszustand beschreiben. Ein solcher Zustand ist an einen konkreten Zeitpunkt im zeitlichen Kontinuum der Simulation gebunden, und entspricht einer Momentaufnahme des Projektverlaufs. Mittels der im Simulationsmodell definierten operationalen Semantik kann auf Basis eines Zustands, ein weiterer, vom betrachteten Zustand aus gesehen in der Zukunft liegender Zustand berechnet werden.

Ein Vorteil dieses auf Zuständen basierten diskreten Ansatzes der Simulation ist die Möglichkeit, Zustände zu sichern und wiederherstellen zu können. Dadurch ist es einfach, Spielstände zu speichern, und somit Sicherungspunkte zu erstellen, um ausgehend von diesen, unterschiedliche Projektverläufe durchspielen zu können. Es besteht außerdem die Möglichkeit, einen Simulationszustand von außen zu manipulieren, um bestimmte Ereignisse umzusetzen oder den Simulationsverlauf gezielt zu verändern, und so weitere, im Simulationsmodell nicht vorgesehene Aspekte zu realisieren.

Ein spezieller Simulationszustand ist der Startzustand der Simulation. Dieser kann um zusätzliche Rahmenparameter angereichert werden und wird *Ausgangsszenario* genannt. Rahmenparameter können beispielsweise die Projektlaufzeit oder zusätzlichen Ereignisse sein, die während der Simulation eintreten.

Def.: Ein **Szenario** beschreibt den Zustand eines Projekts und die relevanten Rahmenparameter unter denen es abläuft. Es enthält zusätzliche Ereignisse, die in seinem Verlauf auftreten, und gegebenenfalls weitere Besonderheiten, die nicht durch das Simulationsmodell abgedeckt werden können.

Dieser Abschnitt beschreibt den strukturellen Aufbau eines Simulationszustands mit seinen Elementen, deren Attributen und den Beziehungen zwischen ihnen. Die Struktur ist an die des *Eclipse Process Framework* (EPF) angelehnt, das dazu dient, Entwicklungsprozesse zu modellieren. Durch die so entstandene Ähnlichkeit lassen sich Prozesse, die im EPF vorliegen, leichter in das Planspielformat übertragen. Da die EPF-Struktur nicht für Simulationszwecke konzipiert ist, diente sie für diese Arbeit nur als erster Ansatz, und wurde durch Änderungen und Erweiterungen an die Verwendung für Simulation und Planspiel angepasst. Abbildung 2.5 zeigt eine Übersicht der Struktur in Form eines UML-Klassendiagramms. Die einzelnen Elemente werden im Rest dieses Abschnitts genauer definiert und erklärt.

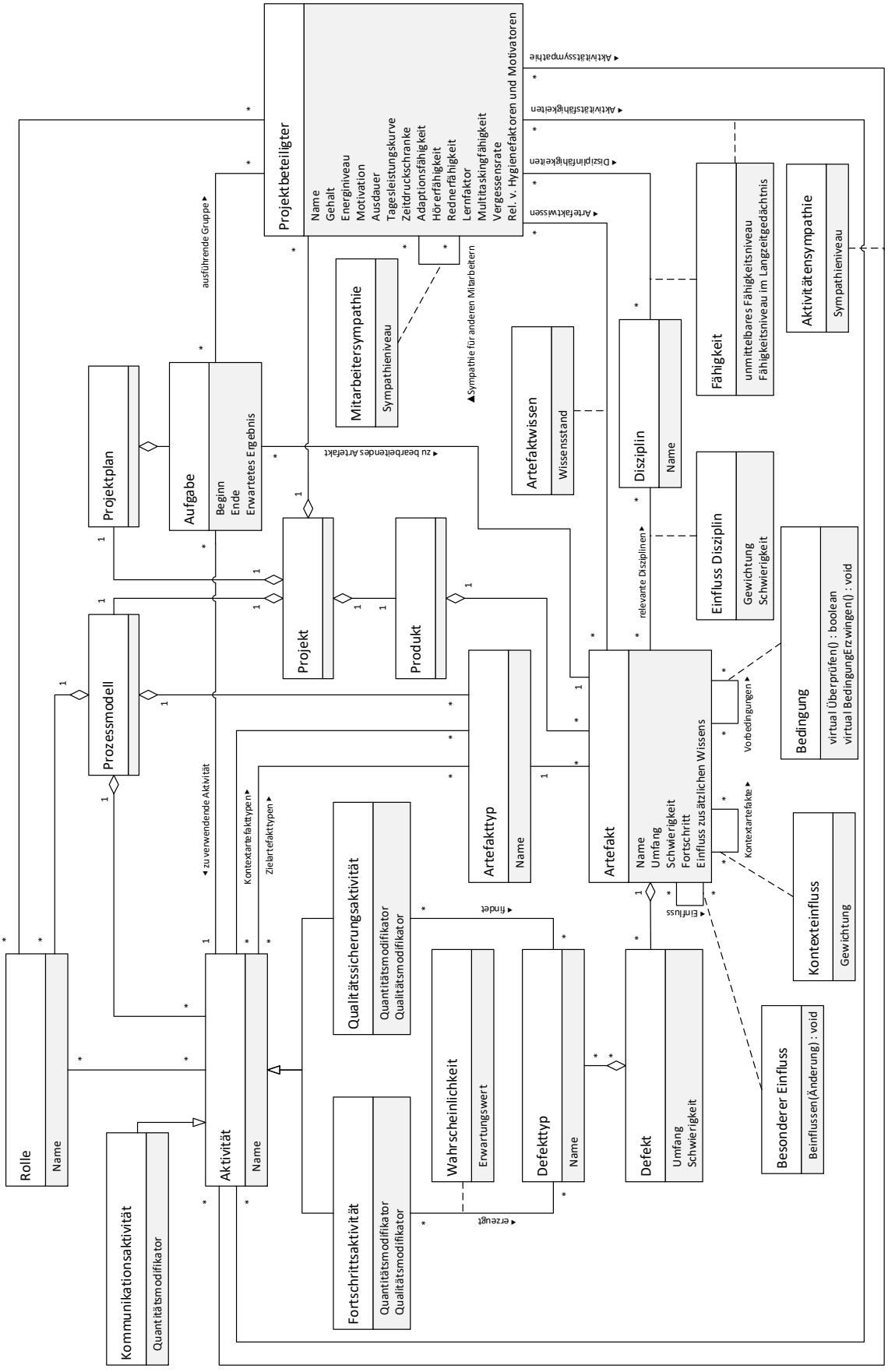


Abbildung 2.5.: UML-Klassendiagramm zum Aufbau des Simulationszustands

Der Simulationszustand lässt sich in drei Teilen betrachten, die gemeinsam die wesentlichen Aspekte des Projekts festlegen: *Produkt*, *Team* und *Prozessmodell*.

Das Produkt definiert das vom Spieler zu erstellende Softwaresystem, als einer Menge von zueinander in Beziehung stehenden *Artefakten*. Ein Artefakt stellt die kleinste Bearbeitungseinheit des Projekts dar.

Das *Team* besteht aus einer Menge einzelner *Projektbeteiligter*, die gesteuert vom Spieler, der die Rolle des Projektleiters einnimmt, die einzelnen Artefakte, und somit das gesamte Produkt, entwickeln. Die einzelnen Projektbeteiligten verfügen über unterschiedliche Eigenschaften, die sich entsprechend auf ihr Verhalten, ihre Leistung, und damit den Projekterfolg auswirken.

Das *Prozessmodell* legt die *Aktivitäten* fest, die dem Team zur Verfügung stehen, um das Produkt zu entwickeln. Im Allgemeinen legen Prozessmodelle neben den Aktivitäten vor allem auch deren Einsatz fest. Dazu wird beschreiben, zu welchem Zeitpunkt im Projektverlauf, und unter welchen Bedingungen welche Aktivitäten durchgeführt, und welche Dokumente dabei erstellt werden. Diese Aspekte des Prozessmodells sind nicht Teil des Simulationszustands, da sie alleine vom Spieler in seiner Rolle als Projektleiter umzusetzen sind.

Im Folgenden werden die wesentlichen Bestandteile des Simulationszustands näher erläutert. Eine Übersicht über alle im Szenario vorkommenden Attribute findet sich außerdem in Anhang A.1.

Artefakte

Ein Softwareprodukt besteht typischerweise aus einer Vielzahl an Bausteinen, die als *Artefakte* bezeichnet werden. Dazu gehören nicht nur der geschriebene Quellcode und die begleitenden Dokumente für den Benutzer, wie zum Beispiel das Handbuch, sondern auch die Dokumente, die während der Entstehung der Software erstellt werden, wie beispielsweise die Definition der Anforderungen, das Architekturdesign, Tests und Inspektionen, Gesprächsprotokolle, Projektpläne, Verträge, Kostenschätzungen, Risikoanalysen und vieles mehr.

Im Rahmen der reinen Softwareentwicklung können alle Artefakte als Dokumente angesehen werden, die Wissen enthalten. Dieses Wissen kann formal formuliert sein, zum Beispiel in Form von Programmcode, mathematischen oder logischen Ausdrücken, oder auch nicht formal, wie zum Beispiel in einem Benutzerhandbuch. Die Artefakte der reinen Softwareentwicklung grenzen sich damit klar von den Artefakten andere Projekte ab, die auch materielle Aspekte umfassen können, wie beispielsweise den Prototypen eines Steuergeräts für ein Fahrzeug. Das Simulationsmodell beschränkt sich auf reine Wissensarbeit. Handwerkliche Aspekte zur Erstellung von materiellen Artefakten unterliegen anderen Regeln, als denen zur Erstellung von Dokumenten [DeMarco, 2002]. Das in dieser Arbeit vorgeschlagene Simulationsmodell beschränkt sich auf die Erstellung reiner Software und unterstützt daher keine materiellen Artefakte.

Jedes Produkt wird durch die Menge seiner Artefakte und deren Beziehungen untereinander beschrieben. Ein Artefakt beschreibt dabei einen Teil des Endprodukts, oder ein Zwischenergebnis dessen Entwicklungsprozesses. Es stellt die kleinste Bearbeitungseinheit im Simulationsmodell dar. Jedes Artefakt wird über eine Menge von Parametern und seinen aktuellen Zustand beschrieben. Parameter sind durch das Szenario vorgegeben und werden, im Gegensatz zum Zustand, durch die Simulation nicht verändert.

Artefakte werden als Ressourcen des Multi-Agenten-Systems mit folgenden Parametern realisiert:

- Über den **Artefakttyp** wird das Artefakt einer bestimmten Kategorie, wie beispielsweise Anforderungsdokument, Codefragment, oder Komponententest zugeordnet. Artefakttypen können im Rahmen des Szenarios frei durch den Spielentwickler definiert werden.

- Der **Umfang** legt den Erstellungsaufwand des Artefakts in Arbeitseinheiten fest.

- Die **Schwierigkeit** gibt an, wie schwer die Erstellung des Artefakts grundsätzlich ist. Sie bezieht sich entsprechend der Definition der Aufgabenschwierigkeit auf die benötigten Fähigkeiten.

- Artefakten können zusätzliche für sie relevante **Disziplinen** zugeordnet werden, die spezielle Fähigkeiten definieren, die zur Bearbeitung des Artefakts notwendig sind. Solche Disziplinen sind durch den Spielentwickler frei definierbar, und können sich beispielsweise auf den Umgang

mit Werkzeugen, Methoden, Formalismen, oder spezielle technische Kenntnisse beziehen. So können beispielsweise für die Implementierung eines bestimmten Artefakts, zusätzlich zu den Fähigkeiten zur Programmierung im allgemeinen, auch Kenntnisse zur Verwendung von Datenbanksystemen gefordert werden. Für jede relevante Disziplin wird dabei deren Anteil am Artefakt und die benötigten Fähigkeiten in Form einer Schwierigkeit festgelegt.

- Mit einem Faktor zwischen 0 und ∞ wird der **Einfluss zusätzlichen Wissens** auf die Bearbeitung des Artefakts festgelegt. In einigen Situationen spielt es eine teils erhebliche Rolle, wie viel ein Projektbeteiligter über ein zu erstellendes oder schon teilweise fertiggestelltes Artefakt weiß. Muss beispielsweise ein Teil eines Artefakts überarbeitet werden, hat der entsprechende Autor dieses Teils hier deutliche Vorteile. Ein anderes Beispiel ist ein Kunde, der Wissen über die Anforderungen des Produkts hat, und somit an deren Definition beteiligt werden muss. Ein Faktor von 1 bedeutet, dass zusätzliches Wissen keinen Einfluss auf die Bearbeitung des Artefakts hat, Faktoren größer 1 geben an, dass ein Wissensvorsprung zu einer Steigerung der Produktivität beiträgt, während Faktoren kleiner 1 bedeuten, dass fehlendes Wissen über den zu bearbeitenden Teil die Produktivität herabsenkt. Die genauen Auswirkungen werden später im Rahmen der Elementarmodelle definiert.

- Mittels **Vorbedingungen** lassen sich die Voraussetzungen definieren, die gelten müssen, um ein Artefakt bearbeiten zu können. Vorbedingungen beziehen sich auf den zum jeweiligen Zeitpunkt geltenden Zustand des Produkts, und somit auf die anderen Artefakte und deren Attribute. Dabei bezieht sich jede Vorbedingung auf genau ein Artefakt und dessen Attribute. Beispielsweise kann so festgelegt werden, dass Artefakt B erst bearbeitet werden kann, wenn Artefakt A zu mindestens 50% fertiggestellt ist. Wird Artefakt A nach der Bearbeitung von Artefakt B nochmals verändert, muss die Vorbedingung gegebenenfalls wiederhergestellt werden, indem der Fortschritt von Artefakt B entsprechend zurückgesetzt wird.

  Um eigene Vorbedingungen umzusetzen, kann eine eigene Klasse von der Bedingungsklasse abgeleitet werden, welche die entsprechenden Prüfmechanismen umsetzt. Die einzelnen Vorbedingungen werden im Rahmen der Szenarioerstellung instanziiert.

- **Kontextartefakte** beschreiben diejenigen anderen Artefakte des Produkts, die einen Einfluss auf die Bearbeitung des betrachteten Artefakts haben. Im Gegensatz zu den Vorbedingungen, bilden Kontextartefakte keine Voraussetzungen für die Bearbeitung, sondern beeinflussen, abhängig von ihrem Zustand, die Produktivität bei der Erstellung des Artefakts. Dazu wird für jedes Kontextartefakt dessen Einfluss in Form eines Faktors zwischen 0 und 1 angegeben. Eine genaue Definition des Faktors findet im Rahmen des dazugehörigen Elementarmodells statt.

- Ein **besonderer Einfluss** ist eine Beziehung zwischen zwei Artefakten, mit denen sich weitergehende Abhängigkeiten umsetzen lassen. Werden Änderungen an einem Artefakt gemacht, beeinflusst dies in manchen Fällen auch andere Artefakte. Wird beispielsweise ein Codeartefakt verändert, muss entsprechend das dazugehörige Dokumentationsartefakt angepasst werden.

Folgende Attribute bilden den Zustand des Artefakts:

- Der **Fortschritt** bzw. die *Vollständigkeit* gibt an, welcher Anteil des Artefakts schon erstellt wurde. Er wird in Arbeitseinheiten zwischen 0 und dem Umfang des Artefakts angegeben. Aus Fortschritt und Umfang kann der *relative Fortschritt* berechnet werden, der angibt, zu welchem Teil das Artefakt schon erstellt wurde.

- Die Menge der **Defekte** beschreibt die Fehler, die während der Bearbeitung des Artefakts gemacht wurden.

- Die **Qualität** ist ein abstraktes Maß, das die Menge der Defekte in Abhängigkeit des Fortschritts des Artefakts in einer einzigen Zahl, wie in Formel 2.1 gezeigt, zwischen 0 und 1 zusammenfasst.

$$Q(\Lambda, c) = \max\left(1 - \frac{\sum_{\lambda \in \Lambda} s_\lambda \cdot 2}{c}, 0\right)$$

$$\begin{aligned} Q &: \text{Funktion zur Umrechnung von Defekten in Qualität} \\ \Lambda &: \text{Menge der Defekte des Artefakts} \\ s_\lambda &: \text{Umfang des Defekts } \lambda \\ c &: \text{Fortschritt des Artefakts} \end{aligned} \tag{2.1}$$

  Zur Berechnung der Qualität wird der Aufwand, der zur Behebung aller Defekte benötigt wird, in Relation zum Umfang des schon fertigen Teils des Artefakts gesetzt. Eine Qualität von 1 wird erreicht, wenn dieser Teil fehlerfrei ist. Eine Qualität von 0 wird erreicht, wenn der Aufwand zur Behebung der Fehler mindestens 50% des bis dahin investierten Entwicklungsaufwands umfasst. Dieser Wert hat sich im Verlauf der Entwicklung des Simulationsmodells als geeignet herausgestellt, da sich das Qualitätsniveau so ohne weitere Umrechnungen als Einflussfaktor für viele andere Aktivitäten verwenden lässt.

Aktivitäten

Eine *Aktivität* beschreibt eine Aktion, die ein Projektbeteiligter auf einem Artefakt ausführen kann, um dessen Zustand zu verändern. Typische Projektaktivitäten im Rahmen der Softwareentwicklung sind beispielsweise Anforderungsanalyse, Architekturdesign, Implementierung, oder Testen.

Aktivitäten werden, entsprechend ihrer Zielsetzung, in die Klassen *Fortschrittsaktivität*, *Qualitätssicherungsaktivität* und *Kommunikationsaktivität* eingeteilt. Diese Unterteilung ist notwendig, da die Veränderungen welche durch diese Tätigkeiten an den Artefakten vorgenommen werden im Detail unterschiedlich simuliert werden müssen.

Fortschrittsaktivitäten beschreiben Aktivitäten, die dem Zweck dienen Artefakte zu vervollständigen. Mittels ihnen wird neue Information geschaffen und in den durch das Artefakt repräsentierten Dokumenten abgelegt. Eine typische Aktivität dieser Klasse ist beispielsweise die Codierung eines Teils der Software.

Qualitätssicherungsaktivitäten dienen dazu, Defekte, die während der Ausführung der Fortschrittsaktivitäten in den Artefakten entstanden sind, aufzudecken. Qualitätssicherungsaktivitäten beschreiben somit die Aktivitäten der analytischen Qualitätssicherung. Typische Aktivitäten sind beispielsweise Testen oder die Durchführung von Inspektionen.

Kommunikationsaktivitäten ermöglichen den Austausch von Wissen zwischen den Projektbeteiligten. Dazu gehört vor allem das Gespräch zwischen zwei einzelnen Personen und das Gespräch in Gruppen.

Jede Aktivität wird über folgende Parameter definiert, die nicht von der Simulation verändert werden können:

- Die Menge der **Zielartefakttypen** gibt an, auf welche Artefakttypen die jeweilige Aktivität angewendet werden kann, um deren Zustand zu verändern. Damit kann beispielsweise verhindert werden, dass Anforderungsdokumente codiert werden, und stattdessen die für deren Erstellung vorgesehene Aktivität der Anforderungsdefinition verwendet werden muss.

- Die Menge der **Kontextartefakttypen** schränkt die Kontext- und Vorbedingungsartefakte, die bei der Ausführung der betrachteten Aktivität berücksichtigt werden anhand des Artefakttyps ein. So kann beispielsweise festgelegt werden, dass für die Codierung eines Artefakts Entwürfe relevant sind, während Testspezifikationen keinen Einfluss haben. Umgekehrt kann für das Testen des Artefakts der Kontext auf die Testspezifikationen eingeschränkt werden.

- Die Projektbeteiligten, die eine bestimmte Aktivität ausführen dürfen, werden über die Menge der **ausführenden Rollen** beschränkt. So kann beispielsweise verhindert werden, dass der Kunde an der Codierung teilnimmt.

- Mittels des **Quantitätsmodifikators** kann der Effekt auf das bearbeitete Artefakt skaliert werden. Bei Fortschrittsaktivitäten skaliert der Modifikator den erzielten Fortschritt, bei Qualitätssicherungsaktivitäten die Menge der gefundenen Defekte, und bei Kommunikationsaktivitäten die Menge des übertragenen Wissens. So können beispielsweise zwei Testaktivitäten definiert werden, die händisches Testen automatisierten Tests gegenüberstellt. Durch den Quantitätsmodifikator der Testaktivitäten kann die deutlich höhere Effizienz der automatisierten Testausführung realisiert werden.

Fortschrittsaktivitäten und Aktivitäten der Qualitätssicherung verfügen jeweils über zwei zusätzliche Parameter:

- Passend zum Quantitätsmodifikator skaliert der **Qualitätsmodifikator** die Qualität, mit der eine Aktivität durchgeführt wird. Dadurch wird bei Fortschrittsaktivitäten der Umfang der ins Artefakt eingebrachten Defekte beeinflusst. Bei Qualitätsaktivitäten hingegen beeinflusst der Modifikator, welche Defekte, bezogen auf ihre Schwierigkeit, aufgedeckt werden können.

- Mittels der **Defekttypen** wird festgelegt, welche Arten von Defekten eine Fortschrittsaktivität erzeugen kann, bzw. welche Arten von Defekte mit einer Qualitätssicherungsaktivität aufgedeckt werden können. Beispielsweise kann eine Codierungsaktivität sowohl Defekte in der Funktionalität, als auch in der Dokumentation erzeugen. Mittels einer Testaktivität kann jedoch der Anteil der Dokumentationsfehler nicht aufgedeckt werden. Für Fortschrittsaktivitäten wird zusätzlich zu jedem Defekttyp eine Wahrscheinlichkeit angegeben, die bestimmt, mit welcher Häufigkeit dieser Defekttyp auftritt.

Projektbeteiligte

Das Projektteam, das der Spieler in seiner Rolle als Projektleiter führt, besteht aus einem oder mehreren *Projektbeteiligen*. Darunter fallen vor allem die Entwickler, aber auch weitere Personen, wie beispielsweise der Kunde, gelten als Projektbeteiligte. Diese werden im Folgenden auch als *Mitarbeiter* bezeichnet. Alle Projektbeteiligte werden im Rahmen des Multi-Agenten-Systems als einzelne Agenten realisiert.

Ein Projektbeteiligter nimmt eine oder mehrere Rollen im Projekt ein, welche die Aktivitäten einschränken, die er ausführen kann. Jeder Projektbeteiligte wird über verschiedene Eigenschaften charakterisiert, die in Form von Parametern festgelegt sind, und nicht durch die Simulation verändert werden können. Er verfügt außerdem über eine Menge an Attributen, die seinen aktuellen Zustand repräsentieren. Der Projektbeteiligte ist die umfangreichste Entität des Simulationsmodells.

Jeder Projektbeteiligte besitzt eine Vielzahl an Eigenschaften, die seine Fähigkeiten und sein Verhalten in der Simulation bestimmen. Diese Eigenschaften werden im Simulationsmodell in ihrer Gesamtheit im weitesten Sinne als *Charakter* der Person bezeichnet. Sie lassen sich nach [Urban, 2000] in die Kategorien *Physis*, *Kognition*, *Emotion* und *sozialer Status* einteilen. Die den Charakter beschreibenden Eigenschaften ändern sich typischerweise nur langsam und werden daher über den Projektzeitraum als konstant angenommen. Einige von ihnen können sich durch entsprechende Ereignisse jedoch auch schlagartig ändern. Charaktereigenschaften gehen in die Simulationsberechnung als feste Parameter ein.

Im Rahmen dieser Arbeit wurden mehrere Charaktereigenschaften und Zustandsattribute als für den Simulationsverlauf relevant identifiziert, und entsprechend als Teil des Charakters eines Projektbeteiligten definiert. Die einzelnen Eigenschaften und Attribute werden mit den jeweiligen Teilmodellen eingeführt, die diese verwenden.

Projektplan

Jedes Projekt verfügt über einen *Projektplan*. Er ist das primäre Instrument, um es dem Projektleiter zu ermöglichen, seinen Mitarbeitern Anweisungen zu geben. Der Projektplan wird vom Spieler erstellt und gepflegt und stellt die wichtigste Schnittstelle zwischen ihm und der Simulation dar.

Im Simulationsmodell ist der Projektplan als eine Menge von Aufgaben definiert. Eine *Aufgabe* wiederum besteht aus *Start-* und *Endzeitpunkt*, der *Aktivität*, die durchgeführt werden soll und dem

Artefakt auf das sich die Aktivität bezieht. Zu jeder Aufgabe werden die Mitarbeiter angegeben, welche die Aufgabe bearbeiten sollen.

Zusätzlich kann zu jeder Aufgabe angegeben werden, welches *Ergebnis* erwartet wird. Wird eine Fortschrittsaktivität durchgeführt, wird das Ergebnis durch den angestrebten Fertigstellungsgrad des Artefakts zum Ende der Aufgabe beschrieben. Bei Qualitätssicherungsaktivitäten wird das erwartete Qualitätslevel nach Bearbeitung der Aufgabe angegeben. Für Kommunikationsaktivitäten kann kein angestrebtes Ergebnis angegeben werden. Das erwartete Ergebnis kann verwendet werden, um Mitarbeiter anzuweisen nur einen Teil des Artefakts zu bearbeiten. Zum Beispiel kann ein Modul nur zur Hälfte von einem Mitarbeiter codiert werden, die zweite Hälfte übernimmt dann ein anderer Mitarbeiter.

2.3.3. Notation und Beispiele

Um die eingeführte Struktur besser zu verdeutlichen, werden die genannten Aspekte hier in einigen Beispielen veranschaulicht.

Um Szenarien grafisch darstellen zu können wird die in Abbildung 2.6 dargestellt Notation verwendet. Es handelt sich dabei nicht um eine Modellierungssprache, sondern lediglich um eine Darstellungsform, die dem besseren Verständnis im Rahmen dieser Arbeit dienen soll.

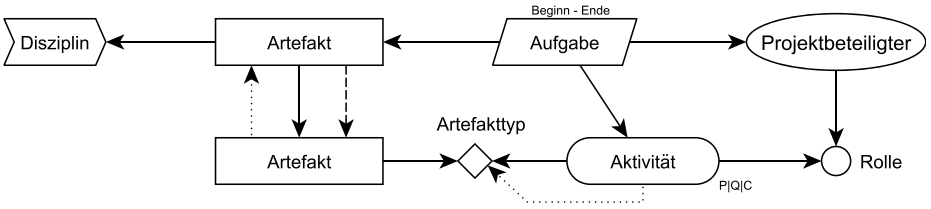


Abbildung 2.6.: Notation für die Struktur von Szenarien

Artefakte werden durch Rechtecke dargestellt, deren *Artefakttypen* durch kleine Rauten. Um zu zeigen, dass ein Artefakt einem bestimmten Typ entspricht, wird ein Pfeil vom Artefakt zum Typ gezogen.

Bedingt ein Artefakt die Bearbeitung eines anderen, so wird ein Pfeil vom bedingenden zum bedingten Artefakt gezogen. Der Pfeil kann mit der Bedingung beschriftet werden. Steht ein Artefakt im Kontext eines anderen, wird ein gepunkteter Pfeil vom Artefakt zu dessen Kontextartefakt gezogen. Der Pfeil kann mit der Gewichtung des Einflusses beschriftet werden. Für besondere Einflüsse eines Artefakts auf ein anderes werden gestrichelte Pfeile verwendet.

Disziplinen werden durch pfeilartige Rechtecke dargestellt. Ist eine Disziplin für ein Artefakt relevant, so wird dies durch einen Pfeil vom Artefakt zur Disziplin angezeigt. Der Pfeil kann mit dem Anteil der Disziplin am Artefakt beschriftet werden.

Aktivitäten werden durch Rechtecke mit abgerundeten Seiten dargestellt. Falls notwendig, wird der *Aktivitätstyp* zusätzlich mit P für Fortschrittsaktivitäten, Q für Qualitätssicherungsaktivitäten und C für Kommunikationsaktivitäten annotiert. Um darzustellen, welche Artefakttypen sich durch diese Aktivität bearbeiten lassen, wird ein Pfeil von der Aktivität zum Artefakttyp gezogen. Um anzuzeigen, dass eine Aktivität bestimmte Artefakttypen als Kontext berücksichtigt, wird ein gepunkteter Pfeil von der Aktivität zum Artefakttyp gezogen.

Rollen werden durch Kreise dargestellt. Um zu zeigen, dass eine Aktivität nur von einer bestimmten Rolle durchgeführt werden kann, wird ein Pfeil von der Aktivität zur Rolle gezogen.

Projektbeteiligte werden als Ovale dargestellt. Um zu zeigen, dass ein Projektbeteiligter eine bestimmte Rolle einnimmt, wird ein Pfeil vom Projektbeteiligten zur Rolle gezogen.

Aufgaben des Projektplans werden durch Parallelogramme dargestellt. Um darzustellen, welches Artefakt mit welcher Aktivität bearbeitet wird und welche Personen daran beteiligt sind, werden Pfeil zu den entsprechenden Elementen gezogen. Aufgaben können bei Bedarf mit ihrem Zeitraum beziehungsweise ihrer Dauer annotiert werden.

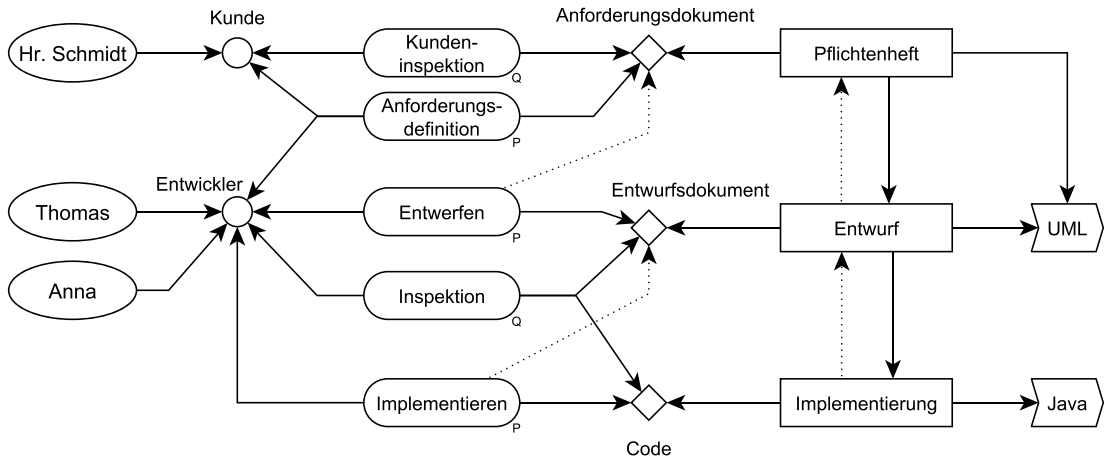


Abbildung 2.7.: Struktur eines Beispielszenarios

Abbildung 2.7 zeigt die Struktur eines einfachen Beispielszenarios. Auf der rechten Seite ist das Produkt, bestehend aus den drei Artefakten *Pflichtenheft* vom Typ *Anforderungsdokument*, *Entwurf* vom Typ *Entwurfsdokument* und *Implementierung* vom Typ *Code* zu sehen. Diese Artefakte sind über nicht näher spezifizierte Vorbedingungen verknüpft, sodass sie in dieser Reihenfolge bearbeitet werden müssen. Dabei bildet das jeweilige Vorgängerartefakt den Kontext seines Nachfolgers. So hat die Vollständigkeit und Qualität des Pflichtenhefts Einfluss auf die Bearbeitung des Entwurfs, und die Vollständigkeit und Qualität des Entwurfs Einfluss auf die Bearbeitung der Implementierung. Für Pflichtenheft und Entwurf ist die Disziplin *UML*, für die Implementierung die Disziplin *Java* zusätzlich relevant.

Zur Erstellung der Artefakte stehen die Aktivitäten *Anforderungsdefinition*, *Entwerfen* und *Implementieren* zur Verfügung, die jeweils nur auf Artefakte des entsprechenden Typs angewendet werden können. Zur Qualitätssicherung stehen die Aktivitäten *Kundeninspektion* und *Inspektion* zur Verfügung, wobei nur der Kunde, der die Anforderungen kennt, die Kundeninspektion durchführen kann. Inspektion ist auf die Entwickler beschränkt, da diese sich mit der Softwareentwicklung auskennen. Die Rolle des Kunden wird von *Hr. Schmidt* wahrgenommen, die am Projektbeteiligten Entwickler sind *Thomas* und *Anna*.

Typischerweise wird ein Produkt in mehr als drei Artefakte aufgeteilt. So können beispielsweise das Pflichtenheft, Entwurf und Implementierung in ihre einzelnen Komponenten zerlegt werden. Abbildung 2.8 zeigt eine mögliche Verfeinerung des Pflichtenhefts. Es wurde dazu in seine einzelnen Bestandteile aufgeteilt. Da die dabei entstandenen Artefakte teilweise aufeinander aufbauen, wurde mittels Vorbedingungen deren Bearbeitungsreihenfolge fixiert. So werden beispielsweise die *Anwendungsfälle* durch die *Szenarien* näher beschrieben. Deshalb können die Szenarien erst bearbeitet werden, wenn die dazugehörigen Anwendungsfälle definiert wurden. Zusätzlich zu den Vorbedingungen bestehen zwischen den Artefakten Kontextabhängigkeiten. So haben beispielsweise nicht nur die Informationen im Artefakt *Anwendungsfälle* Einfluss auf die Bearbeitung der Szenarien, sondern auch die des Artefakts *Produktskizze und Kundengespräch*. Insbesondere fließen alle im Rahmen des Pflichtenhefts erstellten Artefakte in das Gesamtdokument *Pflichtenheft* ein.

Am Beispielszenario in Abbildung 2.7 fällt auf, dass keine Kommunikationsaktivität definiert wurde. Dafür nimmt der Kunde direkt an der Erstellung des Pflichtenhefts teil. Abbildung 2.9 zeigt eine mögliche Verfeinerung dieses Sachverhalts. Der Kunde ist dabei nicht mehr direkt an der Erstellung der Anforderungsdokumente beteiligt, sondern steht stattdessen für Frage und Inspektionen zur Verfügung. Die neue Aktivität *Kundengespräch* kann zwischen Kunde und Entwickler stattfinden und als Gesprächsthema Artefakte des Typs Anforderungsdokument, wie beispielsweise das Pflichtenheft haben.

Im in Abbildung 2.7 gezeigten Beispielszenario steht zur Qualitätssicherung der Implementierung nur die Aktivität *Inspektion* zur Verfügung. Um auch Tests zu erlauben, kann das Szenario wie

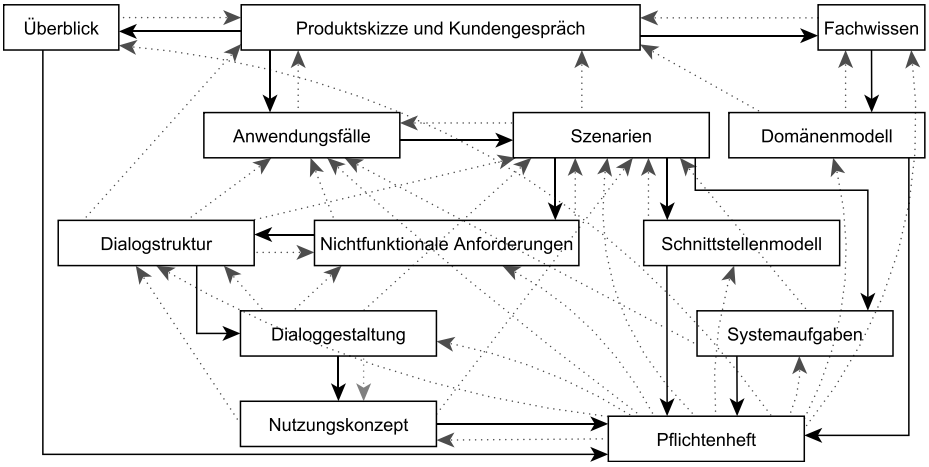


Abbildung 2.8.: Verfeinerung des Artefakts Pflichtenheft (Erstveröffentlichung in [Nassal, 2017])

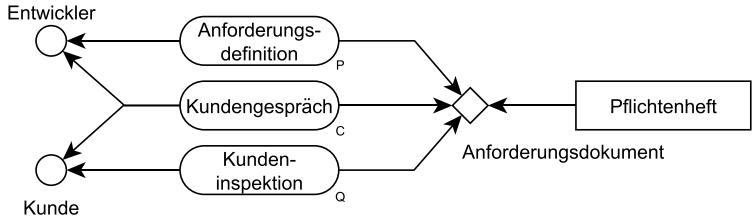


Abbildung 2.9.: Verfeinerung der Kundenkommunikation

in Abbildung 2.10 um manuelle Tests erweitert werden. Da systematisches Testen ohne geeignete Referenzdokumente wenig Sinn ergibt, wird dafür ein zusätzliches Artefakt *Testvorschriften* vom Typ *Testspezifikation* eingefügt, das den Testkontext der Implementierung darstellt, und entsprechend von der neuen Aktivität *manuelles Testen* miteinbezogen wird. Im hier gezeigten Fall basieren die Testvorschriften auf den Informationen des Entwurfs, und sind somit über Vorbedingungen und Kontext entsprechend mit diesem verknüpft.

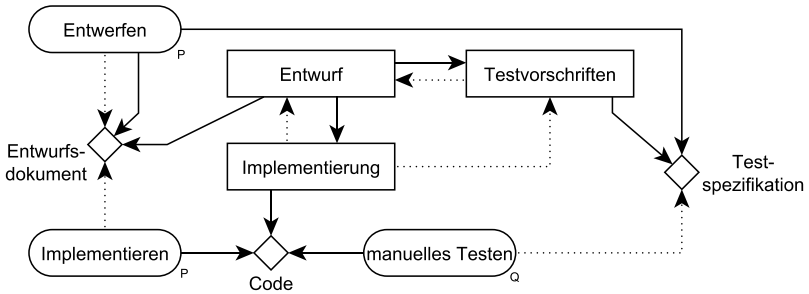


Abbildung 2.10.: Erweiterung durch manuelle Tests

Alternativ oder zusätzlich zu manuellen Tests kann das Szenario auch um automatisierte Tests erweitert werden. Zusätzliche zu den Testvorschriften muss dazu, wie in Abbildung 2.11 gezeigt, ein Artefakt für die Implementierung der Tests vom Typ *Testcode* hinzugefügt werden, das als Testkontext der Implementierung verwendet wird. Da die Tests auf den im Entwurf definierten Schnittstellen basieren, existiert neben den Beziehungen zu den Testvorschriften auch zum Entwurf eine entsprechende Abhängigkeit.

Im Vergleich zu den manuellen Tests verschiebt sich der Aufwand bei automatisierten Tests von der Durchführung der Testaktivität, hin zur Implementierung der Tests. Automatisierte Tests haben den

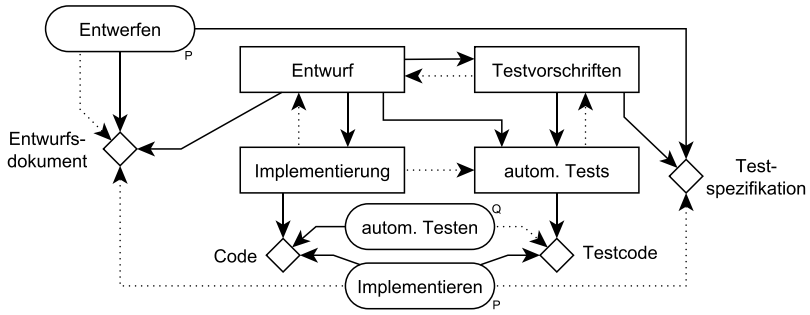


Abbildung 2.11.: Erweiterung durch automatisierte Tests

Vorteil, dass der Aufwand zur Implementierung der Tests nur einfach auftritt, während der Aufwand des manuellen Testens nach jeder Änderung der Implementierung investiert werden muss. Um den Unterschied zwischen den beiden Testverfahren mittels eines Planspiels zu verdeutlichen, können beide Verfahren, wie hier gezeigt, angeboten werden, wobei der Quantitätsmodifikator der Aktivität zum automatischen Testen deutlich höher gesetzt wird, als der des manuellen Testens. Dadurch benötigen die automatisierten Tests nach ihrer Implementierung, im Vergleich zu den manuellen Tests, nur noch geringe Zeit, um die Defekte im zu testenden Artefakt aufzudecken. Ebenso werden die im Szenario für diese Testaktivität angegebenen notwendigen Fähigkeiten auf ein Minimum gesetzt, da das reine Ausführen der Test sehr einfach ist. Welche Defekte gefunden werden, entscheidet sich damit hauptsächlich über die Qualität der Implementierung der Tests, die entsprechend als Kontext der Testaktivität definiert wird.

2.4. Elementarmodelle

Der Simulationsverlauf ergibt sich aus dem Zusammenspiel unterschiedlicher elementarer Aspekte. Diese Aspekte werden durch die im Simulationsmodell enthaltenen Elementarmodelle realisiert. Die einzelnen Modelle sind dabei bis auf wenige Ausnahmen in sich abgeschlossen, und von den anderen Modellen unabhängig. Dadurch wird die Wartbarkeit, Erweiterbarkeit und Wiederverwendbarkeit der einzelnen Modelle und des gesamten Simulationsmodells sichergestellt.

Die Elementarmodelle lassen sich anhand des Elementaraspekts den sie abdecken in die folgenden Kategorien einordnen:

- Die **Physis- und Leistungsmodelle** behandeln Aspekte der physischen Leistungsbereitschaft. Dazu gehören die körperliche Basisleistung, Leistungsschwankungen im Tagesverlauf, Auswirkungen von Zeit- und Leistungsdruck, sowie Erschöpfung und Erholung.
- Die **kognitiven Modelle** behandeln Aspekte der kognitiven Leistung der Projektbeteiligten. Dazu gehört der Erwerb von Fähigkeiten und Wissen, Vergessen und Verlernen, und der Einfluss von Erfahrung und Wissen auf die Arbeitsleistung.
- Die **Modelle der Artefaktbearbeitung** legen fest, wie sich die Eigenschaften und der Zustand von Bearbeitern und Produkt auf den am Artefakt erzielten Fortschritt auswirken, wenn Mitarbeiter am Produkt arbeiten.
- Die **Kommunikationsmodelle** beschreiben den Wissensaustausch zwischen Projektbeteiligten. Dabei wird sowohl die Kommunikation zwischen zwei einzelnen Personen, als auch Gespräche in Gruppen abgedeckt.
- Die **Motivationsmodelle** behandeln die Auswirkungen der emotionalen Aspekte auf die Projektarbeit.
- Das **Modell zum Aufgabenwechsel** legt fest, wie sich der Wechsel zwischen verschiedenen Aufgaben und die zeitlichen Unterbrechungen einer Aufgabe auf die Produktivität auswirken.

Die einzelnen Modelle werden im Folgenden im Detail beschrieben. Zu jeder Kategorie existiert eine eigene Klasse, welche die entsprechenden Elementarmodelle enthält, und die im Spiel instanziiert wird, um die Modelle zugreifbar zu machen. Da die Elementarmodelle innerhalb einer Kategorie häufig aufeinander abgestimmt sind, liegen diese als einzelne Methoden einer gemeinsamen Klasse, und nicht als jeweils selbstständige Klassen vor. Um Modelle zu verändern, um beispielsweise spielspezifisches Verhalten zu realisieren, kann diese Klasse vererbt werden. Durch Überschreiben der entsprechenden Methoden können anschließend die einzelnen Elementarmodelle angepasst werden.

Alle Modelle sind, mit Ausnahme des Motivationsmodells, zustandslos. Sie lassen sich in Form eines Datenflussdiagramms visualisieren. Abbildung 2.12 zeigt die dafür benötigten Elemente. Die Schnittstellen des betrachteten Modells werden durch Rechtecke symbolisiert. Die Verarbeitung der über die Schnittstellen gelieferten Daten wird durch einen mit der entsprechenden Funktion beschrifteten Kreis dargestellt. Um zu zeigen, welche Daten zwischen den Elementen ausgetauscht werden, werden Pfeile verwendet, die gegebenenfalls mit dem entsprechenden Typ der fließenden Daten beschriftet sind.

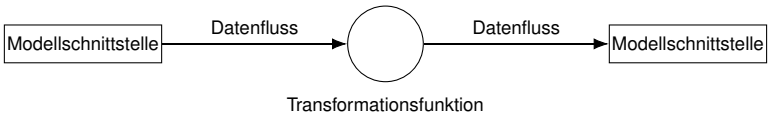


Abbildung 2.12.: Notationselemente für Modelle und deren Kombination

Einige der im Folgenden beschriebenen Modelle basieren auf der Funktion des beschränkten Wachstums, das in Formel 2.2 beschrieben ist. Da alle Berechnungen im Simulationsmodell diskret vorgenommen werden, wird in der Regel die rekursive Form, wie in Formel 2.3 gezeigt, verwendet.

$$
\begin{aligned}
&B(t, S, B_0, k) = S + (B_0 - S) \cdot e^{-k \cdot t} \\
&B_0 : \text{Anfangsbestand} \\
&S : \text{Schranke} \\
&k : \text{Wachstumskonstante} \\
&t : \text{Zeit}
\end{aligned}
\tag{2.2}
$$

$$
\begin{aligned}
&B_{n+1} = B_n \cdot e^{-k \cdot t_\Delta} + S \cdot (1 - e^{-k \cdot t_\Delta}) \\
&t_\Delta : \text{Zeitdifferenz wischen Schritt } n \text{ und Schritt } n+1
\end{aligned}
\tag{2.3}
$$

Im Simulationsmodell wird die Wachstumsfunktion unter anderem dazu verwendet, Parameteränderungen über die Zeit zu glätten. Häufig wird die Änderung eines Parameters berechnet, indem für diesen Parameter zunächst ein neuer Wert ermittelt wird. Dieser basiert nur auf dem aktuellen Zustand, und berücksichtigt die Vergangenheit nicht. Die meisten Parameter beziehen sich jedoch nicht nur auf den letzten, sondern auch auf die davor vergangenen Zeitschritte. Dementsprechend ist die Vergangenheit und auch die Länge des neuen Zeitschritts zu berücksichtigen, wenn der neue Wert dem bestehenden hinzugefügt wird.

Abbildung 2.13 zeigt die Funktionsweise einer solchen Glättung. Die gestrichelte Kurve zeigt die Rahmenfunktion, die dem tatsächlichen Werten zum jeweiligen Zeitpunkt entspricht. Eine solche Kurve zeigt beispielsweise die jeweils momentane persönliche Einschätzung eines Mitarbeiters bezüglich seiner Arbeitssituation zum jeweiligen Zeitpunkt t. Die farbigen Kurven hingegen zeigen die Einschätzung zur Gesamtsituation mit unterschiedlichen Glättungsfaktoren. Zum Zeitpunkt $t = 8.1$ würde der befragte Mitarbeiter seine Arbeitssituation insgesamt als sehr gut einschätzen, obwohl er sie zum konkreten Zeitpunkt als schlecht einschätzt. Es werden somit die in der Vergangenheit herrschenden Einschätzungen im Gesamtbild berücksichtigt und kleinere Schwankungen gedämpft.

Erst wenn eine veränderte Situation über längere Zeit anhält, wird sie sich entsprechend auswirken. Um diese Glättung zu realisieren, wird der neu berechnete Wert als Schranke der Wachstumsfunktion

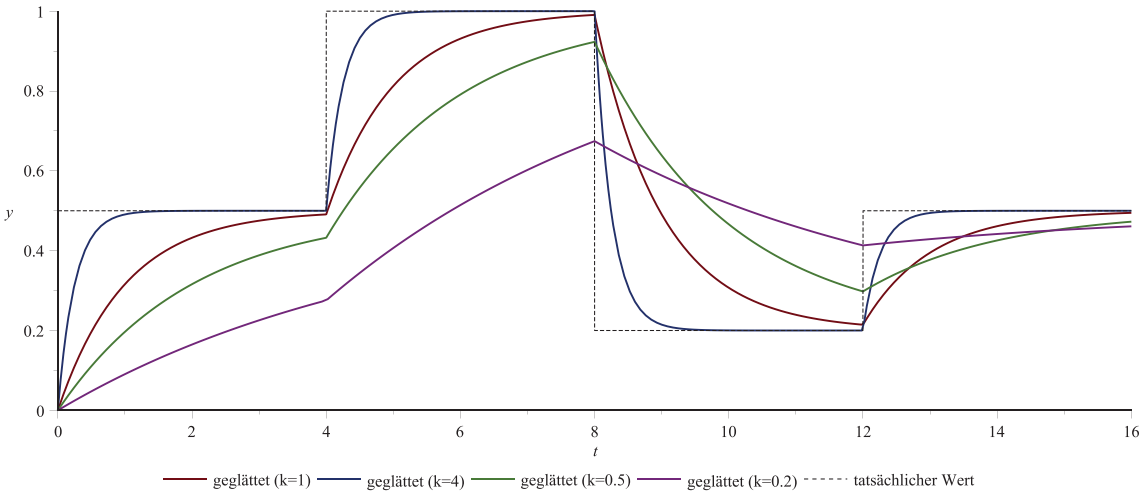


Abbildung 2.13.: Glättungsfunktion

eingesetzt und die Dauer des Zeitschritts als Zeitdifferenz verwendet. Mittels der Wachstumskonstante kann bestimmt werden, wie stark der Verlauf über die Zeit geglättet wird, beziehungsweise wie groß die Schwankungen des betrachteten Parameters sein dürfen. Formel 2.4 zeigt die zu Abbildung 2.13 gehörige Berechnung der Kurven als abschnittsweise Funktion.

$$G(t) = \begin{cases} 0.0, & t = 0 \\ 0.5, & 0 < t \leq 4 \\ 1.0, & 4 < t \leq 8 \\ 0.2, & 8 < t \leq 12 \\ 0.5, & 12 < t \leq 16 \end{cases}$$

$$F(t) = \begin{cases} 0, & t = 0 \\ B(t \quad, G(t), F(0) \ , k), & 0 < t \leq 4 \\ B(t-4 \ , G(t), F(4) \ , k), & 4 < t \leq 8 \\ B(t-8 \ , G(t), F(8) \ , k), & 8 < t \leq 12 \\ B(t-12, G(t), F(12), k), & 12 < t \leq 16 \end{cases} \tag{2.4}$$

$B(t, S, B_o, k)$: Funktion des beschränkten Wachstums

$G(t)$: Funktion des Momentanwerts

$F(t)$: Glättungsfunktion

Viele der im Folgenden gezeigten Elementarmodelle werden mittels mathematischer Formeln beschrieben. Eine Übersicht über die dabei verwendeten mathematischen Symbole findet sich in Anhang A.2. Durch die einzelnen Elementarmodelle werden außerdem einige neue Parameter eingeführt, die Teil des Charakters der Projektbeteiligten sind und diesen näher beschreiben. Eine Übersicht über diese Parameter findet sich in Anhang A.1.

2.4.1. Physis- und Leistungsmodelle

Die Arbeitsleistung der Projektbeteiligten ist durch ihre physischen Kapazitäten begrenzt. Die Physis- und Leistungsmodelle beschreiben die Zusammenhänge zwischen dem Zustand des Projektbeteiligten, seinen Eigenschaften, und der sich daraus ergebenden grundsätzlichen Leistungsfähigkeit. Jeder Projektbeteiligte verfügt über die drei Eigenschaften *Energieniveau*, *Leistung im Tagesverlauf* und die *am Stück gearbeitete Zeit*, die hierfür wichtig sind [Graf, 1960] [Seiwert, 1999] [Martin, 1994] [Rohmert, 1983].

Energieniveau Der wichtigste Einflussfaktor in Bezug auf die physische Leistungsfähigkeit ist die körperliche Verfassung des Projektbeteiligten. Das Simulationsmodell abstrahiert diesen Aspekt durch ein *Energieniveau*, das Teil des Zustands der Projektbeteiligten ist. Das Energieniveau gibt den Gesundheitszustand, die momentane Fitness, und das körperliche Wohlbefinden des Projektbeteiligten als einen einzelnen abstrakten Wert zwischen 0 und ∞ an.

Da die Energie einer Person nicht direkt messbar ist, und es keine etablierte Einheit für diesen Wert gibt, wird sie als relativer Faktor in Bezug auf die durchschnittliche Leistung des durchschnittlichen Projektbeteiligten angegeben. Dabei entspricht ein Wert von 1 diesem Durchschnittswert. Werte kleiner 1 stehen für eine unterdurchschnittliche, Werte größer 1 für eine überdurchschnittliche körperliche Verfassung.

Die Verrichtung von Arbeit im allgemeinen, sowie diverse zusätzliche Stressfaktoren, können das Energieniveau verringern. Erholung und andere Kompensationsmaßnahmen hingegen steigern es.

Um den Verlauf des Energieniveaus und dessen Auswirkungen auf die Produktivität zu berechnen, enthält das Simulationsmodell jeweils ein Elementarmodell für *Erschöpfung*, *Erholung* und den *Einfluss des Energieniveaus auf die Arbeitsleistung*, die weiter unten in diesem Abschnitt beschrieben werden.

Leistung im Tagesverlauf Ein weiterer Faktor, der die Leistungsfähigkeit der Projektbeteiligten beeinflusst, ist die Tageszeit zu der die Leistung erbracht werden soll. Jeder Mensch besitzt eine individuelle Leistungskurve, welche seine persönlichen Leistungs- und Konzentrationsschwankungen im Tagesverlauf beschreibt [Seiwert, 1999]. Ursache für diese Schwankung ist der Biorhythmus des Menschen, der unter anderem erheblich vom Tageslicht und den Schlaf- und Wachgewohnheiten abhängt.

Jeder simulierte Mitarbeiter besitzt eine individuelle *Tagesleistungskurve* über die seine Grundleistung in einem bestimmten Zeitfenster berechnet werden kann. Die *REFA-Normkurve* (Abbildung 2.14) stellt die Leistungsschwankungen eines durchschnittlichen Menschen innerhalb eines 24h Intervalls dar. Zu erkennen ist ein absolutes Maximum morgens um 10 Uhr, ein lokales Mittagstief um 14 Uhr, und ein lokales Abendhoch um 21 Uhr.

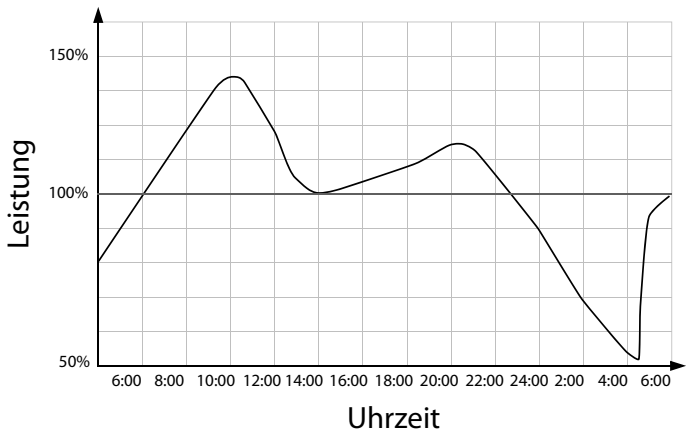


Abbildung 2.14.: REFA-Normkurve [Seiwert, 1999]

Die REFA-Normkurve dient im Simulationsmodell als Ausgangswert, von dem aus die individuellen Kurven der einzelnen Projektbeteiligten entwickelt werden können, um Mitarbeiter in Bezug auf ihre tageszeitabhängige Leistungsfähigkeit genauer zu charakterisieren. Eine solche individuelle Charakterisierung wirkt sich entsprechend direkt und indirekt auf die Simulationsergebnisse aus. So macht es beispielsweise einen Unterschied, zu welcher Tageszeit ein Mitarbeiter schwierige Aufgabe bearbeitet. Die Standardkurve der Mitarbeiter ist so skaliert, dass über das Zeitfenster von 8 bis 16 Uhr ein Durchschnittswert von 1.0 erreicht wird.

Pausen Ein weiterer wichtiger Aspekt in Bezug auf die Leistung eines Mitarbeiters sind die am Stück gearbeitete Zeit und die Pausen, die ein Mitarbeiter macht. Dazu enthält der Zustand jedes Projektbeteiligten einen Wert für die am Stück gearbeitete Zeit in Stunden. Dieser Wert steigt mit der in Arbeit investierten Zeit, und sinkt mit den Pausen. So wird nicht nur berücksichtigt, dass eine Pause gemacht wurde, sondern auch wie lange diese war.

Zeit- und Leistungsdruck

Zeit- und Leistungsdruck beeinflussen die Leistung der Mitarbeiter. Arbeiten Menschen unter Druck, treten laut DeMarco [DeMarco, 2002] zwei Effekte auf:

- Die einzigen möglichen Reaktionen der Mitarbeiter auf Zeitprobleme sind Zeitverschwendung zu vermeiden, nicht kritische Aufgaben zu verschieben und Überstunden zu machen. Da die Denkgeschwindigkeit an sich durch Druck nicht gesteigert werden kann, ist eine Erhöhung der reinen Arbeitsleitung bei Wissensarbeitern durch Leistungsdruck nicht möglich. Durch das Konzentrieren auf die wichtige Aufgabe, das Verschieben unwichtiger Dinge und die Erhöhung der Arbeitszeit, lässt sich laut DeMarco eine kurzfristige Effizienzsteigerung erreichen. Die genannten Maßnahmen wirken sich unmittelbar auf die Produktivität der Mitarbeiter aus und greifen daher sofort.
- Druck verursacht zusätzlichen Stress und wirkt sich damit mittelfristig negativ auf die Arbeitseffizienz aus. Die Auswirkungen von Stress werden dabei in der Regel nicht sofort sichtbar, sondern verringert stetig das allgemeine Wohlbefinden und die Motivation der Mitarbeiter.

Der Zeitdruck unter dem ein Mitarbeiter steht, berechnet sich aus den Fälligkeiten und dem noch geschätzten offenen Arbeitsaufwand der ihm zugewiesenen Aufgaben. Über die Arbeitsgeschwindigkeit des Mitarbeiters und den noch offenen Arbeitsaufwand wird die benötigte Arbeitszeit für den Abschluss aller Aufgaben berechnet, und im Verhältnis zur noch verfügbaren Arbeitszeit betrachtet. Ist für eine Aufgabe beispielsweise Arbeit im Umfang von 50U zu erledigen, es sind jedoch nur noch 40h Zeit bis zur Fälligkeit vorhanden, so ist der Zeitdruck bei einer Arbeitsgeschwindigkeit von $1^{U}/_{h}$ $\frac{50U}{40h} = 1.25$. Der Mitarbeiter muss dementsprechend 25% schneller bzw. länger arbeiten, um die Fälligkeit einhalten zu können.

Der Zeitdruck eines Mitarbeiters, bezogen auf eine einzelne Aufgabe, berechnet sich wie in Formel 2.5 dargestellt.

$$\rho(\tau) = \frac{t_{\Delta\text{benötigt}}}{\max(t_{\Delta\text{Rest}}, 0.1)}$$
$$t_{\Delta\text{benötigt}} = \frac{c_{\text{offen}} \cdot t_{\Delta\text{Anteil}}}{v}$$

$$\begin{aligned}
\rho(\tau) &: \text{Zeitdruck in Bezug auf eine Aufgabe } \tau \\
t_{\Delta\text{benötigt}} &: \text{geschätzte Zeit bis zur Fertigstellung der Aufgabe } \tau \\
t_{\Delta\text{Rest}} &: \text{verbleibende Zeit bis zur Fälligkeit der Aufgabe } \tau \\
c_{\text{offen}} &: \text{offene Arbeitseinheiten der Aufgabe } \tau \\
t_{\Delta\text{Anteil}} &: \text{eigener zeitlicher Anteil an der Aufgabe } \tau \\
v &: \text{geschätzte eigene Arbeitsgeschwindigkeit}
\end{aligned} \tag{2.5}$$

Dazu wird zuerst berechnet, wie viel Zeit der Mitarbeiter für den noch offenen Aufgabenteil unter Annahme einer bestimmten Arbeitsgeschwindigkeit benötigt. Dabei wird der angestrebte Fertigstellungsgrad des Artefakts, der in der Aufgabe angegeben ist, berücksichtigt. Die Geschwindigkeit basiert auf Erfahrungswerten des Mitarbeiters, die wie in Abschnitt 2.6.1 beschrieben zustande kommen. Außerdem ist zu berücksichtigen, dass die Aufgabe gegebenenfalls von mehreren Mitarbeitern bearbeitet wird, und der betrachtetet Mitarbeiter so nur einen gewissen Anteil der Aufgabe zu bearbeiten hat. Die berechnete benötigte Zeit wird anschließend in ein Verhältnis zur noch vorhandenen Zeit gesetzt. Um zu verhindern, dass der Zeitdruck gegen Ende der Aufgabe gegen ∞ steigt, ist die Restzeit mit

0.1 nach unten beschränkt. Die Aufgaben im Projektplan liegen unabhängig der Tagesarbeitszeit der einzelnen Mitarbeiter bezogen auf einen 24h Tag vor. Die vorhandene Zeit aus dem Projektplan muss daher in die Arbeitszeit des Mitarbeiters umgerechnet werden.

Um den Zeitdruck eines Mitarbeiters, bezogen auf den gesamten Projektplan zu ermitteln, werden alle Aufgaben berücksichtigt, die der Mitarbeiter zum betrachteten Zeitpunkt zu erledigen hat. Der Zeitdruck ergibt sich aus dem Maximum des Zeitdrucks der einzelnen Aufgaben und dem Zeitdruck über alle Aufgaben hinweg. Dazu wird, äquivalent zu den einzelnen Aufgaben, die Summe der benötigten Aufwände aller Aufgaben in Relation zum betrachteten Zeitfenster gesetzt. Formel 2.6 zeigt die Kombination der einzelnen Zeitdruckwerte zum Zeitdruckwert über den gesamten Projektplan für einen bestimmten Mitarbeiter b.

$$
\begin{aligned}
&\mathrm{T}' = \{\tau' \in \mathrm{T} | b \in B_{\tau'} \wedge t_{\mathrm{Beginn}_{\tau'}} \leq t \leq t_{\mathrm{Ende}_{\tau'}}\} \\
&\rho = \begin{cases} \max\left(\max\limits_{\tau \in \mathrm{T}'} \rho(\tau), \dfrac{\sum_{\tau \in \mathrm{T}'} t_{\Delta\mathrm{benötigt}_\tau}}{\max\limits_{\tau \in \mathrm{T}'} t_{\Delta\mathrm{Rest}_\tau}}\right) & , \mathrm{T}' \neq \emptyset \\ 0 & , \text{sonst} \end{cases}
\end{aligned}
$$

T' : Menge der Aufgaben des Mitarbeiters b im Projektplan, die zum Zeitpunkt t aktiv sind
τ : Menge aller Aufgaben des Projektplans
$B_{\tau'}$: Menge der Mitarbeiter, die an der Aufgabe τ' beteiligt sind
ρ : Zeitdruck des Mitarbeiters b zum Zeitpunkt t, bezogen auf den gesamten Projektplan
$t_{\Delta\mathrm{benötigt}_\tau}$: die noch für Aufgabe τ benötigte Zeit (siehe Formel 2.5)
$t_{\Delta\mathrm{Rest}}$: verbleibende Zeit bis zur Fälligkeit der Aufgabe τ

(2.6)

Das Ergebnis spiegelt nicht den tatsächlichen Zeitdruck über den gesamten Projektplan hinweg wider. Betrachtet wird nur der für den Mitarbeiter gerade relevante Zeitbereich. Da für den subjektiv vom Mitarbeiter empfundenen Zeitdruck nur aktuelle Aufgaben relevant sind, ist diese Herangehensweise passend. Für das Zeitmanagement des Gesamtplans ist der Projektleiter, beziehungsweise der Spieler verantwortlich.

Es kann vorkommen, dass einem Mitarbeiter zum betrachteten Zeitpunkt keine Aufgabe zugewiesen ist. In diesem Fall ist sein Zeitdruck 0. Für einige Aufgaben, wie beispielsweise der Kommunikation, lassen sich nur schwer Fortschritte und Ziele quantisieren. Solchen Aufgaben wird pauschal ein durchschnittlicher Zeitdruck von 1.0 zugeordnet.

Zeitdruck wirkt sich laut DeMarco nur auf die Arbeitsgeschwindigkeit positiv aus, jedoch nicht auf die Qualität der Arbeit [DeMarco, 1982]. Um die Auswirkungen des Zeitdrucks ρ auf die Arbeitsgeschwindigkeit zu berechnen, definiert das Simulationsmodell die Zeitdruckfunktion P wie in Formel 2.7 dargestellt.

$$
\mathrm{P}(\rho, \hat{\rho}) = \begin{cases} \rho \cdot 0.4 + 0.6 & , \rho < 1.0 \\ \dfrac{1 + 0.25 \cdot \hat{\rho}}{1 + e^{-2.0 \cdot (1 + 0.25 \cdot \hat{\rho}) \cdot (\rho - 1)} \cdot 0.25 \cdot \hat{\rho}} & , \rho \geq 1.0 \end{cases}
$$

$\mathrm{P}(\rho)$: relative Änderung der Arbeitsgeschwindigkeit durch ρ
ρ : Zeitdruck
$\hat{\rho}$: individuelle Zeitdruckschranke des Mitarbeiters

(2.7)

Die Funktion ist so gestaltet, dass sie auf das Intervall $[0.6, 1 + 0.25 \cdot \hat{\rho}]$ abbildet. $\hat{\rho}$ ist eine für jeden Mitarbeiter individuelle Größe, die angibt, bis zu welchem Grad sich seine Leistung durch Zeitdruck erhöhen lässt, und die unter anderem von seiner physischen Belastbarkeit abhängt. Für den durchschnittlichen Mitarbeiter hat sich im Rahmen der Evaluation des Simulationsmodells, eine durchschnittliche maximale Steigerung von 25%, und ein Abfall auf maximal 60% der Leistung bei

komplettem Wegfall von Leistungserwartungen als funktionierende Werte herausgestellt, die sich auch mit den in [Abdel-Hamid, 1991] angegebenen Werten decken.

Für Zeitdruckwerte ≥ 1.0 entspricht die Funktion der Logistischen Funktion mit der Schranke $1 + 0.25 \cdot \hat{\rho}$, dem Basiswert 1.0 und einer Proportionalitätskonstante von 2. Abbildung 2.15 zeigt den Einfluss des Zeitdrucks auf die Arbeitsgeschwindigkeit für unterschiedliche Zeitdruckschranken.

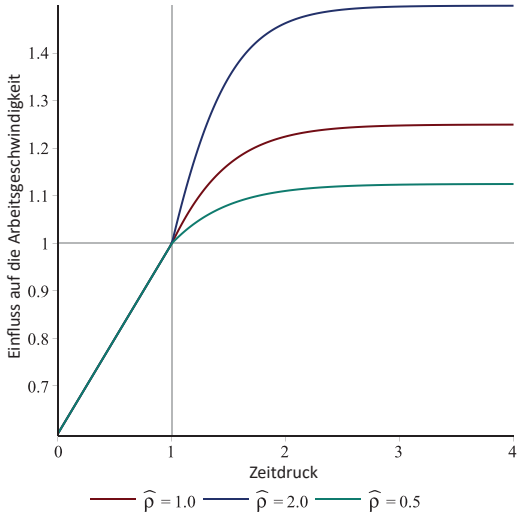


Abbildung 2.15.: Änderung der Arbeitsgeschwindigkeit in Abhängigkeit des Zeitdrucks

Arbeitsleistung

Die physische Leistung gibt an, wie leistungsfähig ein Projektbeteiligter bezogen auf seine physischen Eigenschaften und seinen physischen Zustand ist. Dieser Grundleistung geht als Basiswert in einige Berechnungen, unter anderem des Arbeits- und Lernfortschritts ein. Das Elementarmodell für die physische Leistung verwendet Zeitdruck, Tagesleistungskurve und Energieniveau eines Mitarbeiters, um seine durchschnittliche physische Arbeitsleistung für ein gegebenes Zeitintervall zu berechnen. Abbildung 2.16 zeigt den Aufbau des Modells.

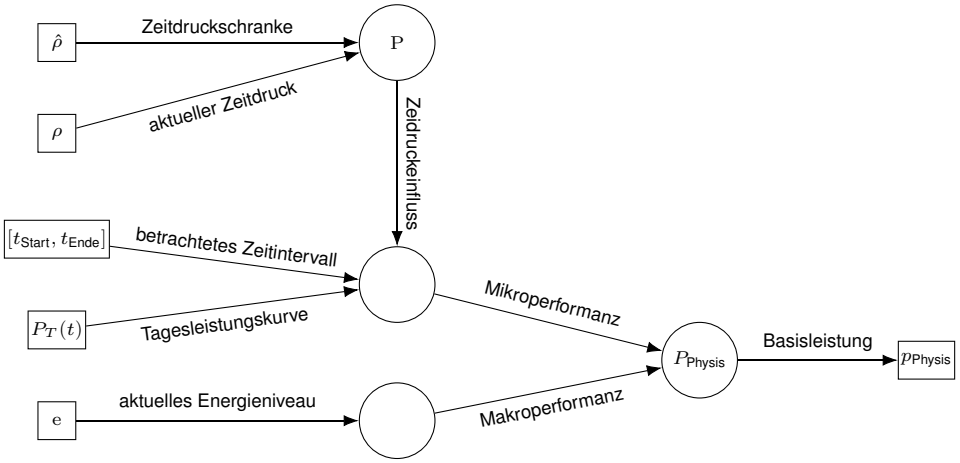


Abbildung 2.16.: Berechnung der physischen Grundleistung eines Mitarbeiters

Um das Modell besser verständlich zu machen, wurde die Arbeitsleistung in das Produkt aus *Mikroperformanz* und *Makroperformanz* zerlegt.

Die Makroperformanz bildet die längerfristige Leistungsentwicklung über Wochen, Monate und Jahre hinweg ab. Sie geht direkt aus dem Energieniveau des betrachteten Mitarbeiters hervor, das den langfristigen Gesundheits- und Fitnesszustand abbildet, und durch die Modelle für Erschöpfung und Erholung verändert wird.

Die Mikroperformanz behandelt kurzfristige Effekte, die sich im Zeitbereich von Minuten, Stunden und Tagen ändern. Sie setzt sich aus der Leistungsschwankung im Tagesverlauf und den Zeitdruck unter dem der Mitarbeiter steht zusammen. Dazu wird auf Basis der Tagesleistungskurve die mittlere Leistung im betrachteten Zeitintervall berechnet. Der Einfluss des Zeitdrucks lässt sich mit der oben beschriebenen Zeitdruckfunktion ermitteln.

Formel 2.8 zeigt die Berechnung der physischen Arbeitsleistung für ein konkretes Zeitintervall.

$$P_{\text{Physis}}(t_{\text{Start}}, t_{\text{Ende}}, \mathrm{e}, \rho, \hat{\rho}, P_T) = \mathrm{e} \cdot \mathrm{P}(\rho, \hat{\rho}) \cdot \frac{\int_{t_{\text{Start}}}^{t_{\text{Ende}}} P_T(t)\mathrm{d}t}{t_{\text{Ende}} - t_{\text{Start}}}$$

$$\begin{aligned} P_{\text{Physis}} &: \text{physische Arbeitsleistung} \\ [t_{\text{Start}}, t_{\text{Ende}}] &: \text{betrachtetes Zeitintervall} \\ \rho &: \text{Zeitdruck im betrachteten Zeitintervall} \\ \hat{\rho} &: \text{individuelle Zeitdruckschranke des Mitarbeiters} \\ P_T(t) &: \text{Tagesleistungskurve} \\ \mathrm{e} &: \text{Energieniveau} \end{aligned} \tag{2.8}$$

Erschöpfung und Erholung

Das Energieniveau der Projektbeteiligten ändert sich im Verlauf des Projekts durch das Zusammenspiel von *Erschöpfung* und *Erholung*. Arbeit und Stress verringern das Energieniveau, Freizeit und Erholung erhöhen es.

Ein Mitarbeiter wird langfristig bei einer gleichmäßigen Arbeitsbelastung ein bestimmtes Energieniveau erreichen und halten. Welches Niveau das ist, hängt zum einen von seiner Belastung, und zum anderen von seiner physischen Ausdauer ab. Da die physische Ausdauer eines Mitarbeiters durch den Projektleiter nicht beeinflusst werden kann, ist das Verhältnis zwischen Arbeit und Freizeit sein primäres Steuerinstrument, um Einfluss auf die physische Leistungsfähigkeit seiner Mitarbeiter zu nehmen. Dazu muss der Projektleiter Arbeits- und Erholungszeiten entsprechend der Konstitution seiner Mitarbeiter wählen, um ein optimales Ergebnis zu erhalten.

DeMarco beschreibt in [DeMarco, 2002] die Auswirkungen der Arbeitsbelastung auf die Leistungsfähigkeit. Eine kurzzeitige Erhöhung oder Verringerungen der Arbeitszeit hat keinen nennenswerten Einfluss auf die Effizienz, mit der Mitarbeiter ihre Arbeit verrichten. Eine langfristige Überbelastung führt hingegen zu einer dauerhaften Effizienzminderung. Überstunden sollten daher nur über einen kurzen Zeitraum eingesetzt werden, um Engpässe zu überbrücken, und sind kein geeignetes Mittel für eine langfristige Steigerung der Produktivität. DeMarco kommt außerdem zu der Erkenntnis, dass Halbtagskräfte deutlich effizienter arbeiten als Mitarbeiter mit 40h-Woche, da sie mehr Erholungszeit zur Verfügung haben.

Auf Basis dieser Erkenntnisse wurden für die Modelle der Erschöpfung und Erholung im Rahmen dieser Arbeit folgende Annahmen getroffen:

- Es gibt Arbeitsphasen, die zu einem Energieabfall führen und Erholungsphasen, die einen Energiegewinn bewirken.
- Das Energieniveau eines Mitarbeiters kann nicht beliebig weit steigen. Ein Mensch kann Energiereserven nur beschränkt aufbauen. Mehr Erholung kann die Leistungsfähigkeit ab einem gewissen Punkt nicht weiter steigern.
- Je schlechter es einem Menschen geht, umso schneller und einfacher lässt sich sein Zustand verbessern. Geht es einem Menschen schon sehr gut, ist es schwer das Wohlbefinden noch weiter zu steigern.

- Die Geschwindigkeit, mit der ein Mitarbeiter ermüdet, ist individuell. Im Simulationsmodell kann die Geschwindigkeit für jeden Mitarbeiter über den charakterspezifischen Parameter *physische Ausdauer* variiert werden.

- Das Optimum für die langfristig betrachtete tägliche Arbeitsleistung eines Mitarbeiters wird bei einer 40h-Woche erreicht. Dabei wird berücksichtigt, dass kürzere Arbeitszeiten zu einer höheren Effizienz führen, und bei längeren Arbeitszeiten mehr Zeit zur Verfügung steht, während der Leistung erbracht werden kann.

  Bei einer 5-Tage-Woche mit jeweils 8 Arbeitsstunden und 30 Tagen Urlaub im Jahr soll die Jahresenergiebilanz leicht positiv sein. Durch die leicht positive Bilanz können einige zusätzliche Überstunden abgedeckt werden. Diese Konfiguration entspricht einer typischen 40-Stunden-Woche, die sich in der westlichen Welt als Wert für ein ausgeglichenes Verhältnis zwischen Arbeit und Erholung etabliert hat. Studien, wie die des National Institute for Occupational Safety and Health [Caruso, 2004] zeigen, dass eine regelmäßige Überschreitung der 40h-Woche eine deutliche Steigerung des Krankheitsrisikos, insbesondere für Herzkreislauferkrankungen nach sich zieht, den sozialen Beziehungen schadet und insgesamt zu einem ungesünderen Lebensstil führt.

- Durch Überstunden kann kurzfristig eine Steigerung der Arbeitsleistung erreicht werden. Mittelfristig führen Überstunden jedoch zu einem Leistungsabfall.

  Bei einer 5-Tage-Woche mit jeweils 12 Arbeitsstunden ohne Urlaub soll das Energieniveau innerhalb eines Jahres von 100% auf ca. 20% fallen. Damit wird realisiert, dass die Leistung von Mitarbeitern, die mit zu viel Arbeit belastet werden, deutlich sinkt.

- Bei einer 5-Tage-Woche mit jeweils 8 Arbeitsstunden ohne Urlaub soll ein leichtes Energiedefizit entstehen. Dadurch wird realisiert, dass die Arbeitsleistung von Mitarbeiter, die keinen Urlaub machen, nach einiger Zeit unter ihrem eigentlichen Potential liegt.

Basierend auf diesen Annahmen wurden die Teilmodelle für Erschöpfung und Erholung definiert. Abbildung 2.17 zeigt den Aufbau des Modells zur Berechnung der Erschöpfung. Die Eingangswerte des Modells sind der aktuelle Energiewert, die am Stück gearbeitet Zeit vor dem zu berechnenden Zeitschritt in Stunden, die Dauer des neuen Zeitschritts in Stunden und der charakterspezifische Parameter für die physische Ausdauer. Bei der Berechnung des Energieverlusts ist außerdem der Zeitdruck zu berücksichtigen, der momentan auf dem Mitarbeiter lastet. Ein erhöhter Zeitdruck erhöht die Leistung des Mitarbeiters und verstärkt damit auch seine Erschöpfung.

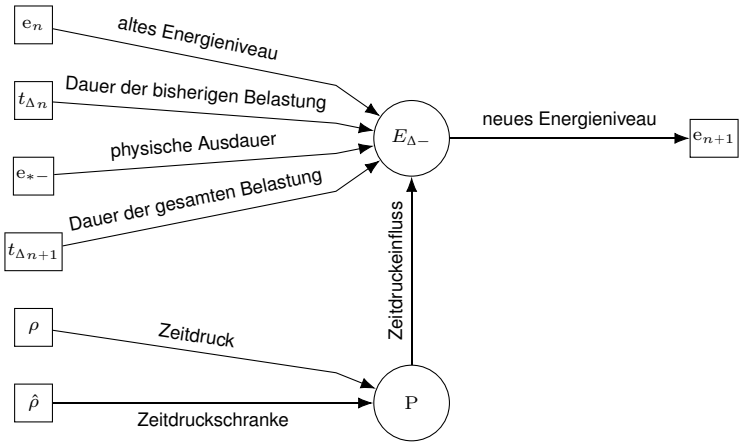


Abbildung 2.17.: Berechnung der Erschöpfung

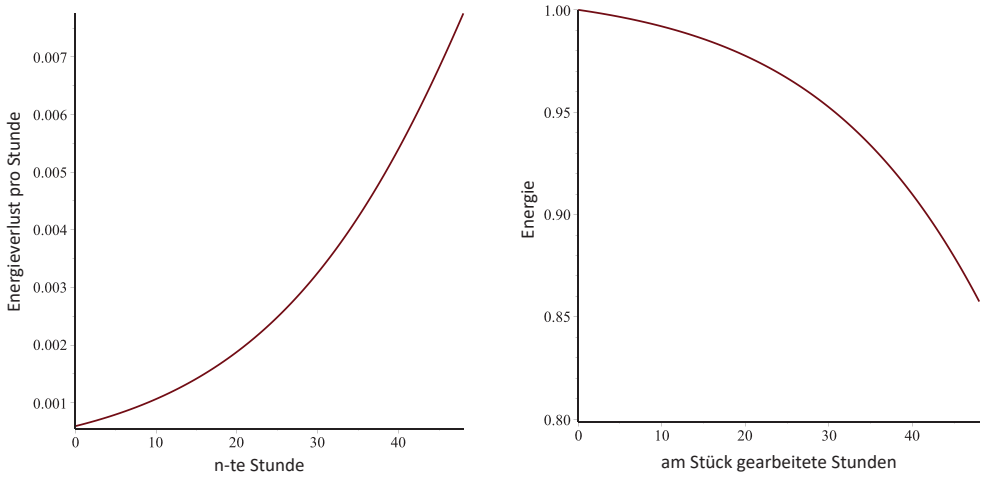
Formel 2.9 zeigt die Berechnung des Energieverlusts durch Erschöpfungsprozesse während der Arbeit. Dieser berechnet sich mittels der Logistischen Funktion mit der Kapazitätsschranke 1, einem

Minimalbestand von 0.01 und einer Proportionalitätskonstante von 0.06, die durch den charakterspezifischen Einflussfaktor um ±50% variiert werden kann. Da die Berechnung in diskreten Zeitschritten stattfindet, wird die rekursive Darstellung $e_\Delta = E_{\Delta-}(t_{\Delta n}, e_{*-}, \rho, \hat{\rho}) - E_{\Delta-}(t_{\Delta n+1}, e_{*-}, \rho, \hat{\rho})$ der Logistischen Funktion verwendet, wobei $t_{\Delta n}$ die am Stück gearbeitete Zeit vor und $t_{\Delta n+1}$ die am Stück gearbeitete Zeit nach dem betrachteten Zeitschritt angibt.

$$E_{\Delta-}(t_\Delta, e_{*-}, \rho, \hat{\rho}) = \left(1 + e^{-0.06\cdot(e_{*-}\cdot 0.5+0.5)\cdot t_\Delta \cdot \max(P(\rho,\hat{\rho}),1)} \cdot \left(\frac{1}{0.01} - 1\right)\right)^{-1}$$

$E_{\Delta-}$: Erschöpfung durch Arbeit
t_Δ : Dauer der Arbeit am Stück in Stunden
e_{*-} : charakterspezifische physische Ausdauer
ρ : Zeitdruck im betrachteten Zeitintervall
$\hat{\rho}$: individuelle Zeitdruckschranke des Mitarbeiters (2.9)

Abbildung 2.18 zeigt die Steigerung des Energieverlusts in Abhängigkeit der am Stück gearbeiteten Zeit. Je länger ein Mitarbeiter am Stück arbeitet, umso drastischer ist der Energieabfall. Durch diesen Ansatz wird die Auswirkung von langen Arbeitszeiten und Überstunden umgesetzt. Das Modell verhindert so, dass ein Mitarbeiter lange am Stück effizient arbeiten kann, und macht damit Überstunden und Nachtschichten unattraktiv.



(a) Energieverlust der n-ten Arbeitsstunde (b) Energieverlust nach n Arbeitsstunden

Abbildung 2.18.: Negative Wirkung von Arbeit auf das Energieniveau

Abbildung 2.19 zeigt das Modell zur Berechnung der Erholung die stattfindet, wenn der Projektbeteiligte nicht arbeitet. Die Eingangswerte des Modells sind das aktuelle Energieniveau und die Dauer der Erholung in Stunden. Der neue Energiewert berechnet sich, wie in Formel 2.10 gezeigt, mittels der Funktion für beschränktes Wachstum mit einer Wachstumsschranke von 2.0 und einer Wachstumskonstante von 0.00022. Durch die Wachstumsschranke wird verhindert, dass ein Mitarbeiter unverhältnismäßige Energiereserven aufbaut. Da die Berechnung der Erholung in diskreten Zeitschritten erfolgt, wird die ebenfalls in Formel 2.10 gezeigte rekursive Form des beschränkten Wachstums verwendet. Abbildung 2.20 zeigt den zeitlichen Verlauf der Energie während der Erholungsphase.

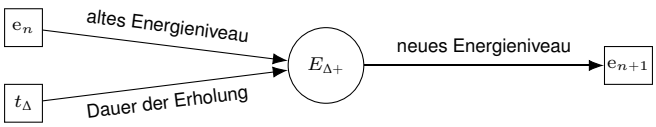


Abbildung 2.19.: Berechnung der Erholung

$$E_{\Delta+}(t_\Delta) = 2 - (2 - \mathrm{e}_0) \cdot e^{-0.00022 \cdot t_\Delta}$$
$$\mathrm{e}_{n+1} = \mathrm{e}_n \cdot e^{-0.00022 \cdot t_\Delta} + 2 \cdot \left(1 - e^{-0.00022 \cdot t_\Delta}\right) \tag{2.10}$$

$E_{\Delta+}$: Erholungsfunktion
t_Δ : Dauer der Erholung in Stunden
e_0 : aktuelles Energieniveau zu Beginn der Erholungsphase

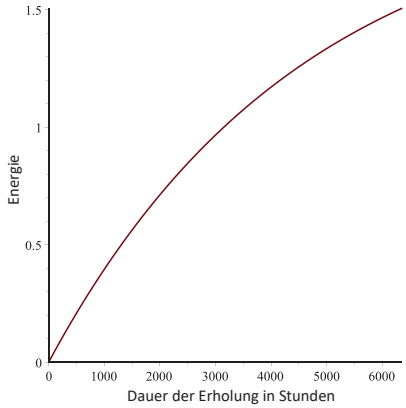


Abbildung 2.20.: Verlauf des Energieniveaus während der Erholungsphase

Erschöpfung und Erholung müssen gut aufeinander abgestimmt sein. Die Modelle und deren Parameter sind so gewählt, dass sich die beiden Modelle gegenseitig ausgleichen. In Abhängigkeit des Verhältnisses zwischen Arbeit und Freizeit, und unter Berücksichtigung der physischen Ausdauer des Mitarbeiters, stellt sich mittelfristig ein bestimmtes Energieniveau ein, das die Verfassung des Mitarbeiters beschreibt.

Tabelle 2.1 zeigt die Auswirkungen der täglichen Arbeitszeit auf das Energieniveau und die Produktivität eines durchschnittlichen Mitarbeiters, die sich rechnerisch aus dem Zusammenspiel von Erschöpfung und Erholung ergeben. Betrachtet werden die Auswirkungen nach einer Woche (obere Tabelle) und nach einem Jahr mit 47 Arbeitswochen und 5 Urlaubswochen (untere Tabelle). Dargestellt sind das Energieniveau und die Arbeitsleistung als die Anzahl der im betrachteten Jahr erbrachten Arbeitseinheiten. Weiterhin zeigt die Tabelle die sich daraus ergebende Effizienz als Verhältnis zwischen investierter Zeit und erwirtschafteter Arbeitseinheiten. Als 100% wird dabei die durchschnittliche Produktivität eines durchschnittlichen Mitarbeiters festgelegt, die laut Definition $1^U/_h$ beträgt (vgl. Abschnitt 2.3.1).

Anhand der Daten in der Tabelle lässt sich erkennen, dass die Parameter des Modells so gewählt wurden, dass die optimalen Ergebnisse für einen Durchschnittsmitarbeiter bei einer 40h-Woche liegen. Wird diese Arbeitszeit regelmäßig überschritten, sinkt die Arbeitsleistung trotz höherer investierter Stundenzahl. Wird die Arbeitszeit regelmäßig unterschritten, sinkt die Leistung aufgrund der geringen Stundenzahl obwohl die Produktivität steigt. Die optimale Arbeitszeit kann durch die charakterspezifische physische Ausdauer angepasst werden. So kann für jeden Mitarbeiter ein individuelles Optimum definiert werden.

Arbeitszeit p. Tag	Energie nach 1 Woche	Arbeitsleistung (abs.)	(rel. zu 8h)	Effizienz
4h	1.02	20U	50%	100%
6h	1.01	30U	75%	100%
8h	1.00	40U	100%	100%
10h	0.99	50U	125%	100%
12h	0.97	59U	148%	98%

Arbeitszeit p. Tag	Energie nach 1 Jahr (exkl. Urlaub)	(inkl. Urlaub)	Arbeitsleistung (abs.)	(rel. zu 8h)	Effizienz
4h	1.46	1.54	1207U	67%	129%
6h	1.22	1.34	1602U	88%	114%
8h	0.94	1.10	1814U	100%	97%
10h	0.61	0.81	1799U	99%	77%
12h	0.22	0.48	1506U	83%	53%

Tabelle 2.1.: Auswirkung unterschiedlicher Arbeitszeiten

Da sich das Energieniveau durch eine Erhöhung der Tagesarbeitszeit und dem damit verbundenen Ungleichgewicht von Erschöpfung und Erholung nur langsam verringert, tritt bei einer Erhöhung der Arbeitszeit zunächst eine Steigerung der Arbeitsleistung pro Tag ein. Kurzfristige Engpässe können so problemlos durch Überstunden bewältigt werden. Werden Überstunden zur Regel, sinkt jedoch die Arbeitsleistung.

Weitere Auswirkungen auf die Motivation des Mitarbeiters, seine Fehltage durch Krankheit bei Überbelastung und die damit verbundenen Auswirkungen auf seine Produktivität werden durch das Motivationsmodell abgebildet, das in Abschnitt 2.4.5 beschrieben wird.

2.4.2. Kognitive Modelle

Jeder Projektbeteiligte besitzt individuelle Fähigkeiten und Wissen. Die kognitiven Modelle definieren, wie Wissen und Fähigkeiten erworben werden, verloren gehen können und wie sie sich auf die Arbeitsleistung der Mitarbeiter auswirken.

Jeder Mitarbeiter verfügt über eine Menge an verschiedenen *Fähigkeiten*, die für das Projekt und die Arbeit daran relevant sind. Fähigkeiten können sich beim Durchführen von Aktivitäten oder durch besondere Ereignisse verbessern. Wird eine Fähigkeit über längere Zeit nicht verwendet, verschlechtert sie sich. Die Fähigkeit beschreibt, wie gut der Mitarbeiter eine bestimmte Aktivität durchführen kann oder eine bestimmte Disziplin beherrscht. Jeder Mitarbeiter besitzt einen Fähigkeitswert für jede im Szenario definierte Aktivität und Disziplin. Diese Werte sind Teil des Mitarbeiterzustands und können sich während des Projektverlaufs ändern.

Jeder Projektmitarbeiter besitzt bestimmtes *Wissen* über die einzelnen Artefakte, die für das Projekt relevant sind. Jedes dieser Artefakte bildet eine eigene Wissensdomäne. Der Zustand eines Projektbeteiligten enthält zu jedem Artefakt einen Wert, der das dazugehörige *Artefaktwissen* angibt. Dieses Wissen wird typischerweise erworben, indem der betreffende Mitarbeiter am entsprechenden Artefakt arbeitet und damit neues Wissen schafft, indem er das Artefakt studiert oder indem er mit anderen Projektbeteiligten kommuniziert, und so Teile deren Wissens erlangt.

Einfluss von Fähigkeiten auf die Arbeitsleistung

Bei der Ausführung von Aktivitäten spielen Fähigkeiten eine große Rolle. Sie beeinflussen, wie schnell und mit welcher Qualität die Aktivität durchgeführt wird. Das Elementarmodell für die kognitive Arbeitsleistung berechnet den Einfluss der Fähigkeiten auf die Produktivität eines Mitarbeiters. Es basiert auf dem von Hanakawa et al. vorgeschlagenen Ansatz [Hanakawa, 2002], welcher wiederum auf dem Produktivitätsmodell von Lord [Lord, 1968] basiert.

Abbildung 2.21 zeigt die Schnittstellen des Modells. Die Eingangsparameter sind das für die Aufgabe benötigte Fähigkeitsniveau, welches der Eigenschaft *Schwierigkeit* des zu bearbeitenden Artefakt entspricht, das Fähigkeitsniveau des Bearbeiters und einer Größe für die Relevanz der Fähigkeit.

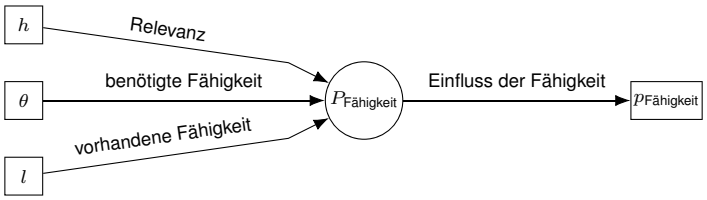


Abbildung 2.21.: Berechnung des Einflusses der Fähigkeiten auf die Produktivität

Das Modell berechnet auf Basis der Differenz zwischen benötigter und vorhandener Fähigkeit einen Faktor, der angibt, wie sich die Produktivität durch den Einfluss der Fähigkeit verändert. Das Modell ist so konzipiert, dass sich die Produktivität nicht ändert, wenn die vorhandene Fähigkeit der benötigten entspricht. Besitzt der Mitarbeiter bessere Fähigkeiten als benötigt, steigt seine Leistung bis maximal auf das doppelte, bei schlechteren Fähigkeiten sinkt die Leistung bis minimal auf 0 ab. Die Größe für die Relevanz gibt dabei an, wie stark sich die Abweichung der vorhandenen von der benötigten Fähigkeit auf die Leistung auswirkt. Bei einem Relevanzwert von 0 ist die Leistung von der Fähigkeit unabhängig, je größer der Wert ist, umso stärker sinkt bzw. steigt die Leistung mit der Differenz zur benötigten Fähigkeit.

Formel 2.11 zeigt die Berechnung des Einflusses und entspricht im Wesentlichen dem Modell in [Hanakawa, 2002], das auf der Verteilungsfunktion der Normalverteilung basiert. Das Integral lässt sich als die Fehlerfunktion darstellen, die nummerisch effizient berechnet werden kann [Abramowitz, 1974].

$$P_{\text{Fähigkeit}}(\theta, l, h) = 2 \cdot \int_{-\infty}^{h\cdot(l-\theta)} \frac{e^{-\frac{1}{2}t^2}}{\sqrt{2\pi}} dt$$

$$\begin{aligned} P_{\text{Fähigkeit}} &: \text{Einfluss der Fähigkeiten auf die Produktivität} \\ \theta &: \text{benötigtes Fähigkeitsniveau} \\ l &: \text{vorhandenes Fähigkeitsnieau} \\ h &: \text{Relevanz der Fähigkeit} \end{aligned} \tag{2.11}$$

Abbildung 2.22 zeigt die Abhängigkeit zwischen Produktivität und vorhandener Fähigkeit, für ein benötigtes Fähigkeitsniveau von 0.5 und unterschiedliche Relevanzgrößen.

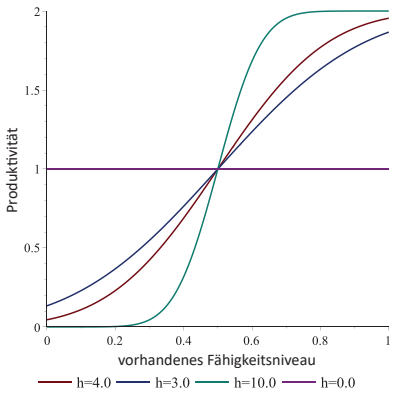


Abbildung 2.22.: Produktivität in Abhängigkeit der vorhandenen Fähigkeiten

Das Modell ist so gestaltet, dass ein durchschnittlicher Entwickler mit durchschnittlichen Fähigkeiten, der eine durchschnittlich schwierige Aufgabe bearbeitet, auch eine durchschnittliche Produktivität erreicht. Der Einfluss von Fähigkeiten ist beschränkt, so kann durch überdurchschnittliche Fähigkeiten die Produktivität höchstens verdoppelt werden. Damit wird verhindert, das eine Überqualifikation für eine Aufgabe einen unrealistisch großen Mehrwert für das Projekt schafft.

Erlernen von Fähigkeiten

Führt ein Mitarbeiter eine Aufgabe durch, verbessert er damit gegebenenfalls seine entsprechenden Fähigkeiten. Je länger sich ein Mitarbeiter mit einer Aufgabe beschäftigt, beziehungsweise je mehr Arbeitsaufwand in eine Aufgabe investiert wird, umso stärker sind im Allgemeinen die Verbesserungen.

Hanakawa et al. beschreiben in ihrer Arbeit ein Modell, das zeigt, wie sich die Arbeit an einer Aufgabe auf die Steigerung der Fähigkeiten auswirkt [Hanakawa, 2002]. Das Elementarmodell zum Erlernen von Fähigkeiten basiert auf den Zusammenhängen dieser Arbeit.

Abbildung 2.23 zeigt die Schnittstellen des Modells für den Fähigkeitserwerb durch Aufgabenbearbeitung. Die Eingangsparameter des Modells sind die in die betrachtete Aufgabe investierte Zeit, das aktuelle Fähigkeitsniveau des Mitarbeiters, das für die Aufgabe benötigte Fähigkeitsniveau und ein charakterspezifischer Lernfaktor der angibt, wie schnell der Mitarbeiter grundsätzlich lernt.

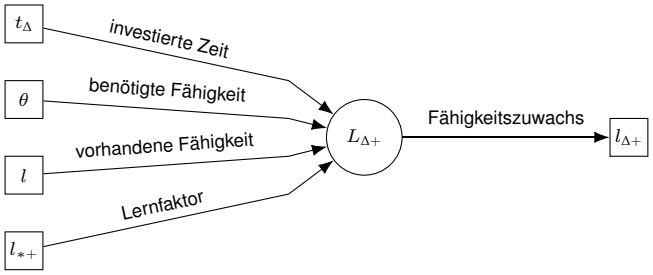


Abbildung 2.23.: Berechnung zum Fähigkeitsgewinn durch Aufgabenbearbeitung

Hanakawa et al. gehen von der Annahme aus, dass ein Fähigkeitszuwachs nur stattfindet, wenn die für die Aufgabe benötigten Fähigkeiten höher sind, als die vorhandenen Fähigkeiten des Bearbeiters. Das bedeutet, dass für einen Lernfortschritt immer eine Herausforderung benötigt wird, der sich der Bearbeiter stellen muss. Er wird so durch die Aufgabe gezwungen etwas für ihn noch Unbekanntes zu machen, und muss sich deshalb neue Fähigkeiten aneignen. Der bestmögliche Lerneffekt tritt ein, wenn die Aufgabe den Bearbeiter fordert, jedoch nicht überfordert.

Formel 2.12 zeigt die Berechnung des Wissenszuwachses durch Bearbeitung einer Aufgabe. Die genaue Form der daraus resultierenden Kurve und die damit verbundenen Faktoren sind so gewählt, dass der Mitarbeiter mit einer angemessenen Geschwindigkeit lernt. Ein optimaler Lerneffekt stellt sich dabei ein, wenn die Schwierigkeit ≈ 0.12 über den Fähigkeiten des Bearbeiters liegt. Ein durchschnittlicher Mitarbeiter kann damit bei optimalen Voraussetzungen innerhalb von ungefähr 7 Jahren von einem völligen Anfänger zu einem absoluten Experten werden. Typischerweise liegen jedoch keine optimalen Bedingungen vor, sodass der Lernfortschritt deutlich länger dauert.

$$L_{\Delta+}(\theta, l, l_{*+}, t_\Delta) = \begin{cases} t_\Delta \cdot l_{*+} \cdot \left(0.08 \cdot (\theta - l)^{2.5} \cdot e^{-40 \cdot (\theta - l)^{1.5}}\right) & , l \leq \theta \\ 0 & , l > \theta \end{cases}$$

$L_{\Delta+}$: Fähigkeitserwerb durch Bearbeitung der Aufgabe
θ : benötigtes Fähigkeitsniveau für die Bearbeitung der Aufgabe
l : vorhandenes Fähigkeitsniveau des Bearbeiters
t_Δ : Dauer der Aufgabenbearbeitung
l_{*+} : persönlicher Lernfaktor des Bearbeiters

(2.12)

Abbildung 2.24 zeigt den Verlauf der Fähigkeiten während der Bearbeitung einer Aktivität. Die betrachteten Mitarbeiter haben zu Beginn ein Fähigkeitsniveau von $b = 0.4$ und die zu bearbeitende Aufgabe eine Schwierigkeit von $\theta = 0.7$. Betrachtet werden Mitarbeiter mit unterschiedlichen Lernfaktoren $l_{*+} = 0.5, 1.0$ und 2.0. Das Schaubild zeigt die für komplexe Lernprozesse typische *S-Kurve* [Jaber, 2016]. Sie besteht aus drei Abschnitten: Zu Beginn reichen die Fähigkeiten des Mitarbeiters noch nicht aus, um effizient lernen zu können. Er muss zuerst die Grundlagen erlernen bevor er sich dem eigentlichen Stoff zuwenden kann. Im zweiten Abschnitt passt die Schwierigkeit des Lernstoffs zum Niveau des Lernenden, der so große Fortschritte machen kann. Im letzten Teil gibt es für den Lernenden nicht mehr viel Neues und die wenigen Aspekte, die er noch nicht beherrscht, sind entweder sehr schwierig oder treten nur selten auf, was zu einer deutlichen Abflachung des Lernfortschritts, bis hin zum Stillstand führt. Ein Mitarbeiter macht während seiner Karriere typischerweise eine Vielzahl solcher Lernkurven durch, indem er die Schwierigkeit der Aufgaben deren er sich annimmt immer weiter steigert, und so neue Wissensgebiete erschließt und damit neue Fähigkeiten erwirbt.

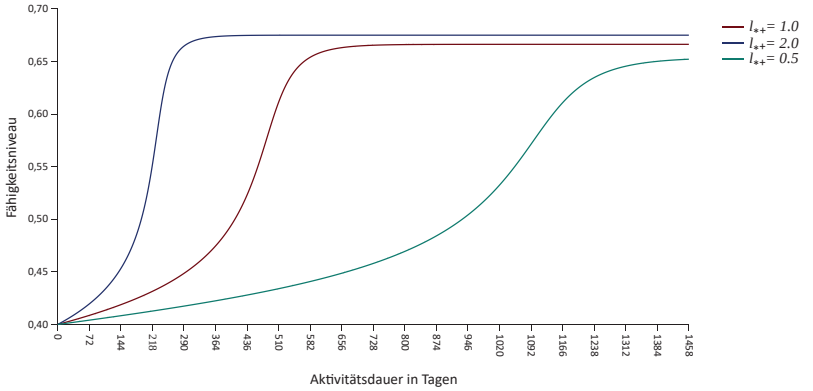


Abbildung 2.24.: Verlauf des Fähigkeitsniveaus bei Lernaktivitäten mit unterschiedlichen Lernfaktoren

Eine der Aufgaben des Spielers ist die Zuteilung der Mitarbeiter zu den offenen Aufgaben. Dabei soll er lernen, die Fähigkeit des Mitarbeiters und die Schwierigkeit der Aufgabe zu berücksichtigen. Neben der erbrachten Leistung des Mitarbeiters bei der Bearbeitung der Aufgabe, hat die Qualität der Zuteilung auch erhebliche Auswirkungen auf die Weiterentwicklung der Fähigkeiten des Mitarbeiters. Abbildung 2.25 zeigt den Vergleich zwischen zwei Konfigurationen, wobei bei dem Mitarbeiter bei der einen (rot) zunächst eine etwas leichtere Aufgaben gegeben wird bevor er die schwere Aufgabe bearbeiten muss, während dem Mitarbeiter bei der anderen (blau) gleich zu Beginn die schwere Aufgabe zugeteilt wird. An den Kurven lässt sich erkennen, dass der erste Mitarbeiter deutlich schnellere Lernfortschritte macht als der zweite und das für die schwere Aufgabe benötigte Fähigkeitsniveau früher erreicht.

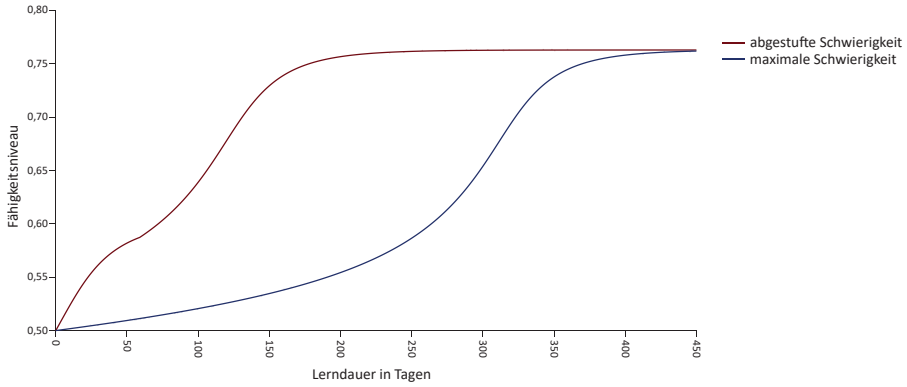


Abbildung 2.25.: Verlauf des Fähigkeitsniveaus bei Lernaktivitäten

Verlernen von Fähigkeiten

Fähigkeiten, die über gewisse Zeit nicht eingesetzt werden, gehen verloren. Das Elementarmodell für das *Verlernen von Fähigkeiten* stellt die dafür notwendigen Berechnungen bereit.

Als Grundlage für die Berechnung der Fähigkeitsverluste dient die Vergessenskurve nach Ebbinghaus [Ebbinghaus, 1992]. Ebbinghaus hat mittels empirischer Forschung gezeigt, dass das Vergessen »anfänglich sehr schnell und weiterhin langsamer geschehe« [Ebbinghaus, 1992, S. 103]. Um diese Eigenschaft umzusetzen, verwendet das Modell die Funktion des nach unten beschränkten Wachstums.

Laut [Michel, 1990] hängt die Vergessensrate maßgeblich von der Beschaffenheit der Informationen ab. So werden innerhalb der ersten 30 Tage nach Erlernen, bei Prinzipien und Gesetzen ca. 5%, bei Gedichten ca. 50%, bei Prosa ca. 60% und bei sinnlosen Silben sogar ca. 80% verlernt. Da Fähigkeiten im hier betrachteten Kontext größtenteils aus Prinzipien und Gesetzen bestehen, ist die Vergessensfunktion so gewählt, dass im Durchschnitt ein Verlust von 5% innerhalb der ersten 30 Tage auftritt. Diese Rate kann durch einen, für den jeweiligen Projektbeteiligten eigenen Parameter individualisiert werden.

Der von Ebbinghaus beschriebene Effekt bezieht sich auf neu erlerntes Wissen. Ebbinghaus unterscheidet zwischen diesem Wissen und dem schon gefestigten Wissen, das sich im *Langzeitgedächtnis* des Menschen befindet, und somit dem Prozess des Vergessens nicht mehr, oder nur noch in geringem Ausmaß unterworfen ist. Je häufiger Wissen wiederholt beziehungsweise angewandt wird, umso mehr geht es in das Langzeitgedächtnis über.

Abbildung 2.26 zeigt Aufbau des Modells zum Fähigkeitsverlust. Die Eingangsparameter sind die vergangene Zeitdauer, für die der Verlust ermittelt werden soll, das aktuelle Fähigkeitsniveau, die individuelle Vergessensrate des Mitarbeiters und ein Wert der beschreibt, wie hoch das Fähigkeitsniveau im Langzeitgedächtnis des Mitarbeiters ist. Formel 2.13 zeigt die dazugehörige Vergessensfunktion. Die Funktion ist so gewählt, dass der Fähigkeitsverlust bei neu erlerntem Wissen, das noch nicht im Langzeitgedächtnis vorhanden ist, über 30 Tage um die in [Michel, 1990] genannten 5% sinkt.

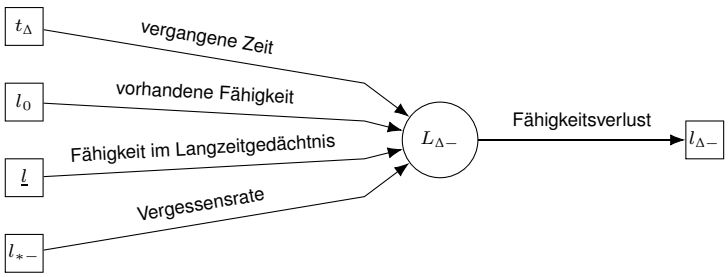


Abbildung 2.26.: Berechnung zum Fähigkeitsverlust

$$L_{\Delta-}(l_0, \underline{l}, l_{*-}, t_\Delta) = l_0 \cdot e^{-0.00007 \cdot t_\Delta \cdot l_{*-}} + \underline{l} \cdot \left(1 - e^{-0.00007 \cdot t_\Delta \cdot l_{*-}}\right) - l_0$$

$L_{\Delta-}$: Fähigkeitsverlust durch Vergessen über die Zeit
l_0 : vorhandenes Fähigkeitsniveau zu Beginn des betrachteten Zeitschritts
$\underline{l}$: Fähigkeitsniveau des Langzeitgedächtnisses
t_Δ : Dauer des betrachteten Zeitschritts in Stunden
l_{*-} : persönlicher Vergessensfaktor des betrachteten Mitarbeiters (2.13)

Um das Langzeitgedächtnis zu realisieren, wird das Modell zum Erlernen von Fähigkeiten um einen zusätzlichen Aspekt erweitert, der für die Übertragung eines Teils der Fähigkeiten in das Langzeitgedächtnis verantwortlich ist. Im Gegensatz zum oben beschriebenen Fähigkeitsniveau, das im Folgenden als *unmittelbares Fähigkeitsniveau* bezeichnet wird, unterliegt das Fähigkeitsniveau im Langzeitgedächtnis keinen Vergessensprozessen.

Formel 2.14 zeigt die Berechnung, die als Ergänzung zum Lernprozess ermittelt, wie sich das Fähigkeitsniveau im Langzeitgedächtnis verändert. Dazu wird mittels der Funktion des beschränkten

Wachstums das aktuelle Niveau im Langzeitgedächtnis in Richtung des unmittelbaren Fähigkeitsniveaus entwickelt, das die Schranke der Wachstumsfunktion bildet, die der Berechnung zugrunde liegt. Durch wiederholtes Anwenden der Fähigkeiten, unabhängig davon, ob dabei ein Lernfortschritt erzielt wird, festigen sich so die Fähigkeiten im Langzeitgedächtnis. Dieser Prozess verläuft langsamer als der eigentliche Lernprozess.

$$\underline{L}_{\Delta+}(\underline{l}_0, l, l_{*+}, t_\Delta) = \underline{l}_0 \cdot e^{-0.00005 \cdot t_\Delta \cdot l_{*+}} + l \cdot \left(1 - e^{-0.00005 \cdot t_\Delta \cdot l_{*+}}\right) - \underline{l}_0$$

$$\begin{aligned}
\underline{L}_{\Delta+} &: \text{Fähigkeitsgewinn im Langzeitgedächtnis durch Aktivitätsausführung} \\
\underline{l}_0 &: \text{Fähigkeitsniveau im Langzeitgedächtnis zu Beginn des betrachteten Zeitschritts} \\
l &: \text{aktuelles unmittelbares Fähigkeitsniveau} \\
t_\Delta &: \text{Dauer des betrachteten Zeitschritts in Stunden} \\
l_{*+} &: \text{persönlicher Lernfaktor des betrachteten Mitarbeiters}
\end{aligned} \tag{2.14}$$

Abbildung 2.27 zeigt den Fähigkeitsverlust über die Zeit unter der Annahme, dass während dieser Zeit kein Lernprozess stattfindet und das Fähigkeitsniveau im Langzeitgedächtnis bei 0.4 liegt.

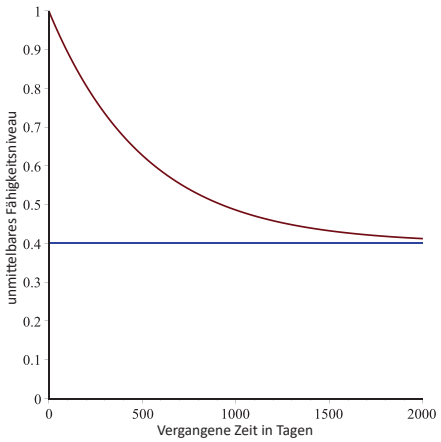


Abbildung 2.27.: Fähigkeitsverlust in Abhängigkeit der Zeit

Erlernen von Artefaktwissen

Der Erwerb von *Artefaktwissen*, unterscheidet sich vom Erlernen von Fähigkeiten. Es gibt drei grundsätzliche Möglichkeiten Artefaktwissen zu erwerben: Beim Erzeugen von neuem Wissen kann dieses direkt aufgenommen werden. Durch das Studium eines Artefakts und des dort dokumentierte Wissen, kann dieses erworben werden. Durch die Kommunikation mit anderen Projektbeteiligten, die das benötigte Wissen besitzen, kann diese ebenfalls erworben werden. Wissenserwerb durch Kommunikation wird in einem eigenen Modellteil behandelt, die anderen beiden Wege des Wissenserwerbs werden in diesem Abschnitt besprochen.

Bearbeitet ein Projektbeteiligter ein Artefakt und erzielt dabei einen Fortschritt, so erzeugt er neues Wissen. Dieses wird dem Bearbeiter ohne weiteren Aufwand gutgeschrieben. Wie Abbildung 2.28 zeigt, hängt das erworbene Wissen dabei nur vom am Artefakt erzielten Fortschritt ab. Kommunikationsaktivitäten und Qualitätssicherungsaktivitäten erzeugen keinen Artefaktfortschritt und schaffen entsprechend kein neues Artefaktwissen, das vom Bearbeiter direkt aufgenommen werden kann.

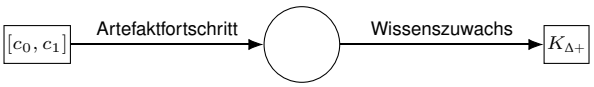


Abbildung 2.28.: Berechnung des Wissenszuwachses bei der Aufgabenbearbeitung

Das in einem vollständig erstellten Artefakt enthaltene Wissen wird auf das Intervall $[0, 1]$ abgebildet. Wird ein Artefaktteil bearbeitet, entsteht dabei das Wissen, das durch das entsprechende Teilintervall beschrieben wird. Die dazugehörige Abbildungsvorschrift wird derart gewählt, dass die Bearbeitungsreihenfolge der Artefaktteile der kontinuierlichen Reihenfolge der Intervalle, und deren Größe dem relativen Anteil des bearbeiteten Artefaktteils am Gesamtartefakt entspricht. Das in einem Artefakt vorhandene Wissen entspricht somit, abhängig vom Fortschritt des Artefakts, einem entsprechenden Teilintervall $[0, c] \subset [0, 1]$, wobei c den aktuellen relativen Artefaktfortschritt beschreibt.

Ein Bearbeiter, der einen Teil eines Artefakts erstellt, erlernt dadurch gleichzeitig das Wissen, das er erschaffen hat. Entsprechend ändert sich sein Artefaktwissen über das bearbeitete Artefakt. Formel 2.15 zeigt die Berechnung des neuen Wissenstands auf Basis der Änderung am Artefakt.

$$
\begin{aligned}
K_{\Delta+} &= \{[c_0, c_1]\} \\
K_{n+1} &= K_n \cup K_{\Delta+} \\
\\
K_{\Delta+} &: \text{Artefaktwissen, das durch die Aktivität entstanden ist} \\
c_0 &: \text{relativer Fortschritt des Artefakts vor der Bearbeitung} \\
c_1 &: \text{relativer Fortschritt des Artefakts nach der Bearbeitung} \\
K_n &: \text{Artefaktwissen vor der Bearbeitung des Artefakts} \\
K_{n+1} &: \text{Artefaktwissen nach der Bearbeitung des Artefakts}
\end{aligned}
\tag{2.15}
$$

Bearbeitet ein Projektbeteiligter ein Artefakt, über das er nur unzureichendes Wissen besitzt, muss er einen Teil der Bearbeitungszeit darauf verwenden, das Artefakt zu studieren und damit das fehlende Wissen zu ergänzen.

Wie viel Wissen er dabei erwirbt, hängt, wie in Abbildung 2.29 gezeigt, von der investierten Zeit, der Qualität des betrachteten Artefakts, und seinem persönlichen charakterspezifischen Wert für die Adaptionsfähigkeit ab. Eine Annahme für dieses Elementarmodell ist, dass Artefakte einer höheren Qualität besser zu verstehen sind, da sie gut strukturiert, in sich konsistent und ausreichend dokumentiert sind. Außerdem ist der Umfang des Artefakts relevant, da das Artefaktwissen relativ zum gesamten Artefakt angegeben wird, und das Studium größerer Artefakte entsprechend mehr Zeit in Anspruch nimmt.

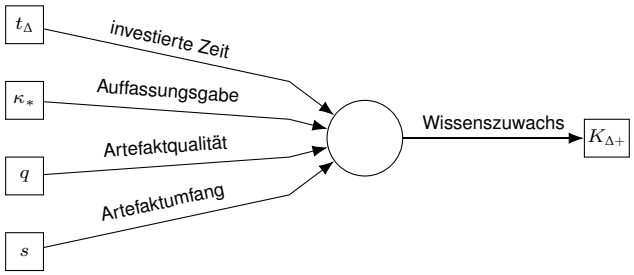


Abbildung 2.29.: Berechnung des Wissenszuwachses durch das Studium eines Artefakts

Formel 2.16 zeigt die Berechnung des Wissensgewinns durch das Studium eines Artefakts. Dazu wird zunächst der relative Wissenszuwachs berechnet. Anschließend wird das Artefaktwissen um diesen Wert erhöht, indem beliebig gewählte Teilintervalle des potentiell durch das Artefakt erlernbaren Artefaktwissens hinzugefügt werden. Das potentiell erlernbare Wissen wiederum entspricht dem im Artefakt dokumentierten Wissen, das, bezogen auf die Wissensdomäne des Artefakts, dem Intervall zwischen 0 und dem aktuellen relativen Fortschritt des Artefakts entspricht. Formel 2.16 beschreibt die Bedingungen, die durch einen entsprechenden Algorithmus erfüllt werden müssen. Das Simulationsmodell setzt diese Anforderungen um, indem zunächst der Umfang des Wissenszuwachses $|K_{n+1}| - |K_n|$ wie angegeben berechnet wird. Anschließend werden die Lücken zwischen den schon vorhandenen Teilintervallen im Artefaktwissen des Bearbeiters gefüllt. Dazu wird der Umfang des

Wissenszuwachses gleichmäßig auf die Lücken aufgeteilt.

$$\left(|K_{n+1}| - |K_n| = \frac{t_\Delta \cdot \kappa_* \cdot (0.3 + q) \cdot 5}{s}\right) \wedge \forall x \in K_{n+1} \colon x \subset [0, c] \tag{2.16}$$

K : Domänenwissen des Bearbeiters über das zu studierende Artefakt
t_Δ : investierte Zeit in Stunden
κ_* : charakterspezifischer Wert für die Adaptionsfähigkeit
q : Qualität des betrachteten Artefakts
s : Umfang des betrachteten Artefakts
c : relativer Fortschritt des betrachteten Artefakts

Vergessen von Artefaktwissen

Das Vergessen von Artefaktwissen folgt den gleichen Regeln wie das Verlernen von Fähigkeiten. Artefaktwissen liegt im Gegensatz zum Fähigkeitsniveau nicht als ein einzelner Wert, sondern in Form einer Menge von Teilintervallen vor. Der mittels Formel 2.13 berechnete Verlust muss dementsprechend diese Intervalle reduzieren. Wie Abbildung 2.30 zeigt, geht kein Aspekt des Langzeitgedächtnisses in die Berechnung ein. Diese Entscheidung wurde getroffen, da die Repräsentation von Artefaktwissen deutlich komplexer ist, als die von Fähigkeiten. Dadurch sind die damit verbundenen Berechnungen komplexer und deren negativen Auswirkung auf die Performance der Simulationsberechnung stärker. Da Artefaktwissen im Gegensatz zu den Fähigkeiten typischerweise nur innerhalb eines Projekts, und damit in einem relativ kurzen Zeitraum relevant ist, ist diese Vereinfachung akzeptabel.

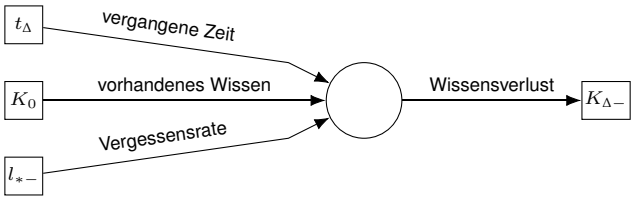


Abbildung 2.30.: Berechnung zum Wissensverlust

Da der Aspekt, welches Wissen wie und wann verloren geht, vom Simulationsmodell nicht betrachtet wird, wäre die beste Abstraktion, zufällige Bereiche innerhalb des Artefaktwissens zu entfernen. Das ist jedoch aus Implementierungssicht problematisch, da dadurch viele kleine Intervalle entstehen, die gespeichert und verwaltet werden müssen. Das Simulationsmodell reduziert daher lediglich die Größe der vorhandenen Teilintervalle, ohne dabei neue Intervalle entstehen zu lassen. Somit bleibt die Belegung der Wissensdomäne durch das Artefaktwissen erhalten, ohne die Performance der Berechnungen zu beeinflussen. Abbildung 2.31 illustriert den Unterschied beider Vorgehensweisen anhand eines einzelnen Teilintervalls. Der grüne Bereich symbolisiert dabei das Wissen nach dem Vergessen, der blaue Bereich zeigt das Wissen, das durch den Vergessensprozess verloren gegangen ist.

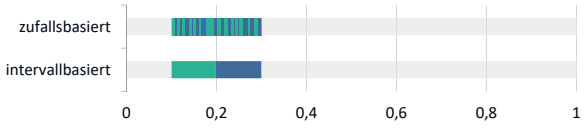


Abbildung 2.31.: Vergleich der zufallsbasierten und intervallbasierten Wissensreduktion

Die Optimierung führt dazu, dass bestimmte Abschnitte des Artefaktwissens bevorzugt verloren gehen. Dadurch können unerwünschte Anomalien entstehen. Bearbeiten beispielsweise zwei Mitarbeiter gemeinsam ein Artefakt, erwerben sie damit das gleiche Wissen. Vergessen beide anschließend einen Teil dieses Wissens, werden die verloren gegangenen Intervalle sehr ähnlich sein. Die beiden

Mitarbeiter können somit ihren Wissensverlust nur schlecht gegenseitig ausgleichen, was nicht der Realität entspricht. Derartige Anomalien sind während der Entwicklung des Simulationsmodells jedoch nie aufgefallen. Das weist darauf hin, dass solche Anomalien die Ergebnisse der Simulation allenfalls geringfügig verfälschen.

2.4.3. Kommunikationsmodelle

Arbeiten mehrere Mitarbeiter gemeinsam an einem Projekt, so ist die Kommunikation zwischen ihnen ein essentieller Bestandteil der Simulation. Kommunikation ist notwendig, wenn Wissen zwischen zwei oder mehr Mitarbeitern ausgetauscht werden muss. Hat zum Beispiel ein Mitarbeiter die erste Hälfte eines Artefakts alleine bearbeitet, und somit auch das alleinige Wissen über diesen Artefaktteil, und soll anschließend ein anderer Mitarbeiter die andere Hälfte des Artefakts erstellen, so ist es sinnvoll, das Wissen des ersten Mitarbeiters auf den zweiten zu übertragen, um einen möglichst guten Kontext für dessen Arbeit zu schaffen. Um diesen Transfer zu realisieren, werden die *Kommunikationsmodelle* benötigt.

In der Psychologie existiert eine Vielzahl unterschiedlicher Kommunikationsmodelle, die unterschiedliche Schwerpunkte haben und unterschiedliche Aspekte der Kommunikation berücksichtigen [Röhner, 2012]. Dazu gehören unter anderem die *Encoder-/Decoder-Modelle*, welche eine sehr einfache Sicht auf den Kommunikationsprozess liefern. Aspekte, wie die emotionale Verbindung der Kommunikationsteilnehmer, das Einnehmen der Perspektive des anderen, etc. werden hier nicht berücksichtigt.

Eines der bekanntesten Modelle dieser Kategorie ist das *Kommunikationsmodell nach Shannon und Weaver* [Shannon, 1963], das auch häufig als *Sender-Empfänger-Modell* bezeichnet wird. Das Simulationsmodell nutzt diesen Ansatz, da er einfach zu verstehen und leicht umzusetzen ist. Während der Evaluation hat sich gezeigt, dass dieses Modell ausreichend ist, um die gewünschten Effekte in der Simulation zu erzeugen.

Das Modell von Shannon und Weaver besteht aus sechs Komponenten: Sender, Kodierer, Übertragungskanal, Dekodierer und Adressat, sowie einer Störquelle, die auf den Kanal wirkt. Um eine Nachricht zu übermitteln, kodiert der Sender diese in ein Signal, das über den Kanal zum Adressat hin übermittelt wird, und dabei ggf. durch Störungen verändert wird. Der Adressat dekodiert dieses Signal, um die Nachricht zu empfangen.

Einfache Kommunikation

Der Basisfall einer Kommunikation ist, dass ein Mitarbeiter redet, während ihm ein anderer Mitarbeiter zuhört. Dabei adaptiert der hörenden Mitarbeiter einen Teil des ausgesprochenen Wissens. Eine Kommunikationsaktivität im Simulationsmodell bezieht sich immer auf genau ein Artefakt über das gesprochen wird. Werden mehrere Artefakte in einem Gespräch thematisiert, so wird das Gespräch in einzelnen Teilen zu je einem Artefakt betrachtet. Dieser Basisfall entspricht den Bedingungen, die im Modell von Shannon und Weaver gelten. Der redende Mitarbeiter ist der Sender, der hörende Mitarbeiter der Adressat. Der Kommunikationskanal ist die akustische Übertragung von Sprache. Auf eine Störquelle wird im Simulationsmodell verzichtet.

Abbildung 2.32 zeigt die Schnittstellen des Elementarmodells, welches den Basisfall umsetzt. Die Kommunikationsgeschwindigkeit geht als Basiswert für das zu übertragende Wissen ein. Im Rahmen der Evaluation hat sich herausgestellt, dass eine Kommunikationsgeschwindigkeit von $50^{U}/_{h}$ angemessen ist. Das bedeutet, durch eine Stunde Kommunikation kann so viel Wissen übertragen werden, wie typischerweise in 50h Arbeit entwickelt wird. Bei Werten, die deutlich über $50^{U}/_{h}$ liegen, verliert die Kommunikation zu sehr an Bedeutung, da sie zu wenig Zeit benötigt. Werte, die deutlich darunter liegen, führen zu einem zu großen Kommunikationsaufwand. Das wiederum führt dazu, dass Teams aus mehreren Mitarbeitern übermäßig an Produktivität durch Kommunikation verlieren.

Um zu berechnen, welches Wissen übertragen werden soll, wird der Wissensstand des Redners und des Hörers benötigt. Das Modell geht zunächst von einer möglichst optimalen Form der Kommunikation aus. Das bedeutet, der Redner erzählt genau das, was seinem Zuhörer an Wissen fehlt. Dieses Wissen lässt sich als die Differenz zwischen dem Wissen des Redners und dem Wissen des Hörers darstellen.

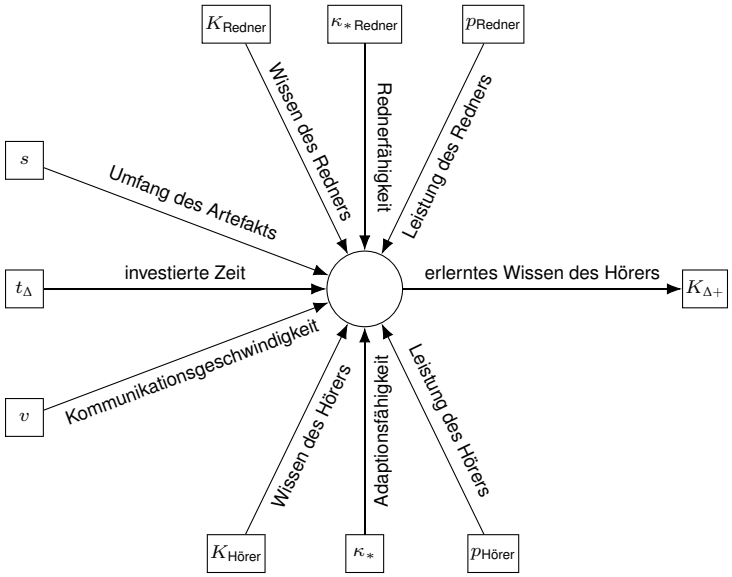


Abbildung 2.32.: Berechnung des direkten Wissenstransfers

Da eine optimale Kommunikation nicht der Realität entspricht, wird mittels zwei charakterspezifischen Faktoren eine entsprechende Abweichung berechnet. Für den Redner wird dazu die *Rednerfähigkeit* angegeben, die beschreibt, wie exakt der Redner über das redet, was der Hörer wissen möchte und wie gut er dies in Worte fassen kann. Bezogen auf das Modell von Shannon und Weaver entspricht dieser Vorgang der Kodierung auf der Senderseite, bei der Artefaktwissen in Sprache umgewandelt wird. Für den Hörer wird mit der *Adaptionsfähigkeit* ein charakterspezifischer Wert angegeben, der beschreibt, wie gut er in Worte gefasstes Wissen aufnehmen, verarbeiten und sich merken kann, was der Dekodierleistung nach Shannon und Weaver auf der Adressatseite entspricht. Neben den charakterspezifischen Werten wird außerdem die physische Grundleistung der Kommunikationsbeteiligten berücksichtigt. Ein Mitarbeiter der beispielsweise übermüdet ist, hat eine geringere Aufnahmefähigkeit als ein Mitarbeiter der fit ist.

Je mehr Zeit in die Kommunikation investiert wird, umso mehr Wissen wird übertragen. Um das zu realisieren, geht die Kommunikationszeit in die Berechnung ein.

Je größer ein Artefakt ist, umso länger dauert es, es vollständig zu beschreiben. Da das Artefaktwissen relativ zur Artefaktgröße angegeben wird, wird für die endgültige Berechnung der Umfang des Artefakts benötigt.

$$K_{\Delta+} \subseteq K_{\text{Redner}} - K_{\text{Hörer}} \ \wedge \ |K_{\Delta+}| = v \cdot p_{\text{Redner}} \cdot \kappa_{*\text{Redner}} \cdot p_{\text{Hörer}} \cdot \kappa_* \cdot t_\Delta \cdot s^{-1}$$

$K_{\Delta+}$: durch Kommunikation erworbenes Wissen des Hörers
$K_{\text{Hörer}}$: Wissen des Hörers über das Artefakt
K_{Redner} : Wissen des Redners über das Artefakt
$|K_{\Delta+}|$: relativer Anteil des erworbenen Wissens am Gesamtwissen
v : Kommunikationsgeschwindigkeit in $^{\text{U}}/_{\text{h}}$
t_Δ : die in die Unterhaltung investierte Zeit in Stunden
s : Umfang des Artefakts über das geredet wird
κ_* : Fähigkeit des Zuhörers Gesagtes zu verstehen
$\kappa_{*\text{Redner}}$: Fähigkeit des Redners Wissen in Worte zu fassen
$p_{\text{Hörer}}$: physische Grundleistung des Hörers
p_{Redner} : physische Grundleistung des Redners

(2.17)

Formel 2.17 zeigt die Berechnung des durch die Kommunikation übertragenen Wissens. Dazu wird zunächst die Quantität des Wissens als Produkt der einzelnen Faktoren berechnet und in Relation zum Umfang des Artefakts gesetzt, um den relativen Anteil zur gesamten Wissensdomäne zu bekommen. Im zweiten Teil wird dieser rein quantitative Wert in Artefaktwissen umgesetzt, um zu definieren, welches Wissen tatsächlich übertragen wird. Da das Modell vom konkreten Wissen abstrahiert, wird dabei lediglich berücksichtigt, dass der Umfang des übertragenen Wissens stimmt. Für das übertragene Wissen selbst gilt, dass es der Redner besitzt, der Hörer jedoch nicht. Dazu werden zufällige Teilintervalle im Gesamtumfang des zuvor berechneten Umfangs gewählt, die diese Voraussetzung erfüllen. Sie stammen also aus der Differenz zwischen Rednerwissen und Hörerwissen. Um diese Berechnung performant durchführen zu können, wird der gleiche Ansatz wie beim Vergessen von Artefaktwissen (vgl. Abbildung 2.31) angewandt, um die Wissensdifferenz zwischen Redner und Hörer auf den durch die Kommunikation zu übertragenden Umfang zu reduzieren.

Erweiterte einfache Kommunikation

Ein allgemeinerer Fall der Kommunikation ist, dass eine Person redet, jedoch mehrere Personen zuhören. Das oben beschriebene Modell wird dazu durch einige Änderungen erweitert. Anstelle des Wissens des einzelnen Hörers, wird die Schnittmenge des Wissens aller Hörer verwendet, die vom Wissen des Redners abgezogen wird, um das potentiell zu kommunizierende Wissen zu ermitteln. Das bedeutet, der Redner erzählt nur Aspekte, die mindestens einer der Zuhörer noch nicht kennt. Je besser das Vorwissen der Gruppe übereinstimmt, umso effizienter ist die so entstehende Kommunikation.

Um die individuellen Wissensgewinne zu ermitteln, wird die Berechnung, wie in Formel 2.18 gezeigt, in zwei Schritte unterteilt. Zuerst wird das vom Redner vermittelte Wissen berechnet. Dieses Wissen teilt er der Gruppe mit. Anschließend wird für jeden einzelnen Hörer individuell berechnet, welchen Teil er davon aufnimmt.

$$
\begin{aligned}
&K_{\Delta+}{}' \subseteq K_{\text{Redner}} - \bigcap K_{\text{Hörer}_i} \ \wedge \ |K_{\Delta+}{}'| = v \cdot p_{\text{Redner}} \cdot \kappa_{*\text{Redner}} \cdot t_\Delta \cdot s^{-1} \\
&K_{\Delta+_i} \subseteq K_{\Delta+}{}' \ \wedge \ |K_{\Delta+_i}| = |K_{\Delta+}{}'| \cdot p_{\text{Hörer}_i} \cdot \kappa_{*_i} \\
&K_{\Delta+}{}' : \text{vom Redner vermitteltes Wissen} \\
&K_{\Delta+_i} : \text{vom } i\text{ten Hörer aufgenommenes Wissen}
\end{aligned}
\tag{2.18}
$$

Konferenzkommunikation

Eine komplexere Form des Wissenstransfers erhält man, wenn die Rollen des Redners und des Zuhörers nicht klar getrennt sind. Im Rahmen einer Konferenz nehmen die Beteiligten zu unterschiedlichen Zeitpunkten unterschiedliche Rollen ein. Um das zu realisieren wird dafür ein eigenes Elementarmodell definiert.

Auch dieses Modell geht von einer möglichst optimalen Kommunikationsform aus. Das bedeutet, es redet immer genau eine Person, während alle anderen nur zuhören. Die Redezeiten der einzelnen Teilnehmer orientieren sich am Wissensvorsprung der einzelnen Personen in Bezug auf die Gruppe. So bekommt die Person am meisten Redezeit, die auch potentiell das meiste Wissen vermitteln kann.

Um den Wissenstransfer zu berechnen, wird die Konferenzkommunikation in einzelne Zeitabschnitte aufgeteilt, in denen jeweils einer der Teilnehmer die Rolle des Redners und die anderen die Rolle des Hörers einnehmen. Diese Abschnitte können einzeln mit dem Modell der erweiterten einfachen Kommunikation berechnet werden.

Um die Redezeit der einzelnen Teilnehmer zu berechnen, wird wie bei der *einfachen Kommunikation* zunächst das gemeinsame Wissen, das nicht kommuniziert werden muss, da alle darüber verfügen, berechnet. Vereinigt man das Wissen aller Teilnehmer und zieht davon das gemeinsame Wissen ab, so erhält man das Wissen K_{max}, das potentiell kommuniziert werden kann. Die Gesamtredezeit t_Δ wird entsprechend proportional zum vorhandenen Wissen K_k des Teilnehmers geschnitten mit K_{max}, wie in Formel 2.19 gezeigt, unter ihnen aufgeteilt.

$$
t_{\Delta k} = \frac{|K_k \cap K_{max}|}{|K_{max}|} \cdot t_\Delta \tag{2.19}
$$

2.4.4. Modelle der Artefaktbearbeitung

Führt ein Mitarbeiter eine Aufgabe aus, indem er an einem Artefakt arbeitet, verändert er damit im Allgemeinen dessen Zustand. Eine Aufgabe beschreibt dabei eine Aktivität, die ein Projektbeteiligter zu einem bestimmten Zeitpunkt oder über einen bestimmten Zeitraum auf einem Artefakt ausführt. Kommunikationsaktivitäten verändern den Zustand der Artefakte nicht, und werden daher ausschließlich mittels der Kommunikationsmodelle realisiert.

Die *Modelle der Artefaktbearbeitung* realisieren die Ausführung von artefaktverändernden Aufgaben. Dazu wird zunächst der Umfang der effektiv getanen Arbeit und die Qualität, mit der diese durchgeführt wird, berechnet. Die Ergebnisse werden anschließend auf den Zustand des Artefakts angewendet. Umfang und Qualität können dabei unabhängig vom Typ der konkret durchgeführten Aktivität berechnet werden, während die Anwendung auf das Artefakt davon abhängt, ob es sich um Qualitätssicherungsarbeit oder Fortschrittsarbeit handelt. Das bedeutet, ob das Ziel der Aktivität das Aufdecken von bestehenden Defekten im Artefakt ist, oder ob das Artefakt durch die Aktivität weiter vervollständigt werden soll.

Um die Änderung am Artefakt durch die vom Mitarbeiter ausgeführte Aufgabe zu berechnen, wird diese zunächst auf Basis der relevanten Disziplinen in einzelne unabhängige Teilaufgaben zerlegt. So kann beispielsweise die Erstellung des GUI-Artefakts in Teilaufgaben für das Grafikdesign, die Usabilitykonzepte und die Implementierung aufgeteilt werden. Für jede Teilaufgabe wird die Arbeitsleistung einzeln berechnet. Anschließend werden die einzelnen Teilleistungen zu einer Gesamtleistung kombiniert, die für das gesamte Artefakt gilt. Die ermittelte Gesamtleistung kann abschließend abhängig vom Typ der Aktivität auf das Artefakt angewendet werden, um dessen Zustand zu verändern.

Die Arbeitsleistung einer Teilaufgabe ist unter anderem von den Kontextartefakten der Aufgabe und dem Wissen des Bearbeiters über diese Artefakte abhängig. Je nach Artefaktkonfiguration kann es außerdem eine Rolle spielen, welches Wissen der Bearbeiter über den neu zu erstellenden Artefaktteil besitzt. So sind beispielsweise Mitarbeiter, die einen Artefaktteil mehrfach bearbeiten im Vorteil.

Zerlegung der Aufgabe und Kombination der Teilergebnisse

Für jedes Artefakt können eine oder mehrere Disziplinen festgelegt werden, die angeben, welche zusätzlichen Fähigkeiten für die Bearbeitung des Artefakts relevant sind. Für jede dieser Disziplinen wird dazu deren auf das Artefakt bezogene Schwierigkeit und ihr Gewicht angegeben. Das Gewicht legt fest, welchen Anteil die jeweilige Disziplin an der Gesamtaufgabe hat. Formel 2.20 zeigt die Partitionierung der Aufgabe in die einzelnen Teilaufgaben und die Kombination der Arbeitsergebnisse der Teilaufgaben zum Gesamtergebnis.

$$
\begin{aligned}
&D = \{d_1, \ldots, d_n\} \\
&\mathrm{T} = \{\tau_0, \tau_1, \ldots, \tau_n\} \\
&G = \{g_0, g_1, \ldots, g_n\} \text{ mit } \sum_{i=1}^{n} g_i \leq 1 \wedge g_0 = 1 - \sum_{i=1}^{n} g_i \\
&W(\mathrm{T}) = \sum_{i=0}^{n} W(\tau_i) \cdot g_i \cdot t_\Delta
\end{aligned}
\tag{2.20}
$$

D : Menge der für die Aufgabe relevanten Disziplinen
T : Partitionierung der Aufgabe in Teilaufgaben
τ_0 : disziplinlose Teilaufgabe
$g_1, \ldots, g_n$: Gewichte der einzelnen Disziplinen
$W(\tau)$: Arbeitsleistung der (Teil-)Aufgabe τ
t_Δ : Dauer der Aufgabe in Stunden

Eine Aufgabe wird so zerlegt, dass für jede relevante Disziplin eine eigene Teilaufgabe entsteht. Die Idee dazu stammt aus der Arbeit von Hanakawa et al. [Hanakawa, 2002]. Die Gewichtung der

Disziplinen, die über das Szenario festgelegt wird, lässt einen Gesamtwert < 1 zu. Die Differenz der Gewichtssumme zu 1 entspricht dem Anteil der Aufgabe, für den keine besondere fachliche Fähigkeit notwendig ist. Dieser Anteil wird als eine eigene Teilaufgabe aufgefasst. Werden für ein Artefakt keine zusätzlichen Disziplinen angegeben, entsteht so nur eine einzige Teilaufgabe, die der Gesamtaufgabe entspricht. Die Aufteilung der Bearbeitungszeit auf die einzelnen Teilaufgaben wird proportional zur Gewichtung der Disziplinen gewählt. Bei Gewichtssummen > 1 werden diese durch Normalisierung auf eine Gesamtsumme von 1 reduziert, indem die einzelnen Gewichte durch die Gesamtsumme geteilt werden.

Berechnung der Arbeitsleistung

Für jede der durch die Zerlegung entstandenen Teilaufgaben wird die *Arbeitsleistung* ermittelt, die aus zwei Werten besteht. Der erste Wert gibt den Umfang der Leistung an, der zweite Wert beschreibt die Qualität, mit der dieser Umfang erzeugt wurde. Das Wertepaar wird später auf das Artefakt angewendet, um dessen Zustandsänderung zu bewirken.

Kern der Berechnung ist das Modell zur Berechnung der Arbeitsleistung. Es kombiniert die einzelnen Aspekte zum oben beschriebenen Wertepaar. Die darin einfließenden Faktoren stammen entweder direkt aus dem Szenario, dem Zustand von Produkt und Bearbeitern, oder werden durch weitere Elementarmodelle erzeugt. Um das Modell zur Arbeitsleistung besser zu strukturieren und damit übersichtlicher zu halten, wurden der Einfluss des Aufgabenkontextes und der Einfluss des zusätzlichen Wissens über den zu erstellenden Artefaktteil in eigene Teilmodelle ausgelagert.

Abbildung 2.33 zeigt die Struktur der Berechnung der Arbeitsleistung einer einzelnen Teilaufgabe. Die Teile für den *Kontexteinfluss* und den *Einfluss zusätzlichen Wissens* sind von den einzelnen Teilaufgaben unabhängig und können somit einmal für alle Teilaufgaben gemeinsam vorberechnet werden. Die Parameter *Fähigkeit* und *Schwierigkeit* beziehen sich auf die konkrete Teilaufgabe und werden wie in Formel 2.21 gezeigt, jeweils mit Hilfe des geometrischen Mittels aus den Werten der entsprechenden Disziplin und den Werten der Aktivität der Aufgabe berechnet. So wird gewährleistet, dass die Rahmenbedingungen, welche die Aufgabe an sich definiert genauso einen Einfluss auf das Ergebnis haben, wie die speziellen Eigenschaften der Disziplin, die sich nur auf die jeweilige Teilaufgabe beziehen.

$$\theta_i = \sqrt{\theta_{\text{Aktivität}} \cdot \theta_{\text{Disziplin}_i}}\ ,\ \theta_0 = \theta_{\text{Aktivität}}$$
$$l_i = \sqrt{l_{\text{Aktivität}} \cdot l_{\text{Disziplin}_i}}\ ,\ l_0 = l_{\text{Aktivität}} \tag{2.21}$$

θ_i : Schwierigkeit der Teilaufgabe i
θ_0 : Schwierigkeit der disziplinlosen Teilaufgabe
l_i : Fähigkeit des Bearbeiters in Bezug auf die Teilaufgabe i
l_0 : Fähigkeit des Bearbeiters in Bezug auf die disziplinlosen Teilaufgabe

Der *Kontexteinfluss* gibt an, wie sich der aktuelle Zustand des gesamten Produkts auf die Bearbeitung des Artefakts auswirkt. Er lässt sich in zwei Faktoren aufteilen, von denen einer den Einfluss des zu bearbeitenden Artefakts ausdrückt, der andere den Einfluss der Artefakte, die den zusätzlichen Kontext des zu bearbeitenden Artefakts bilden. Diese Kontextartefakte werden im Szenario für die einzelnen Artefakte definiert. Während der Arbeitsleistungsberechnung werden sie anhand der zur Aufgabe gehörenden Aktivität, entsprechend den dort angegebene Kontextartefakttypen gefiltert.

Das Teilmodell zur Ermittlung des Kontexteinflusses basiert auf der Annahme, dass vorangegangene Arbeitsergebnisse eine Auswirkung auf die Arbeitsleistung haben. So lässt sich ein Dokument, das korrekt erstellt wurde, schneller und mit deutlich weniger Fehlern bearbeiten. Einen ebensolchen Einfluss haben Dokumente, die für den betrachteten Arbeitsschritt benötigt, selbst jedoch nicht verändert werden. So haben beispielsweise die Referenzdokumente bei einer Inspektion erheblichen Einfluss auf die Inspektionsergebnisse [Weller, 1993], oder die Spezifikation einer Schnittstelle auf die spätere Integration [Sommerville, 2007]. Es ist jedoch nicht ausreichend, dass eine benötigte oder hilfreiche Information in einem Dokument vorhanden ist, der Mitarbeiter, der die entsprechende

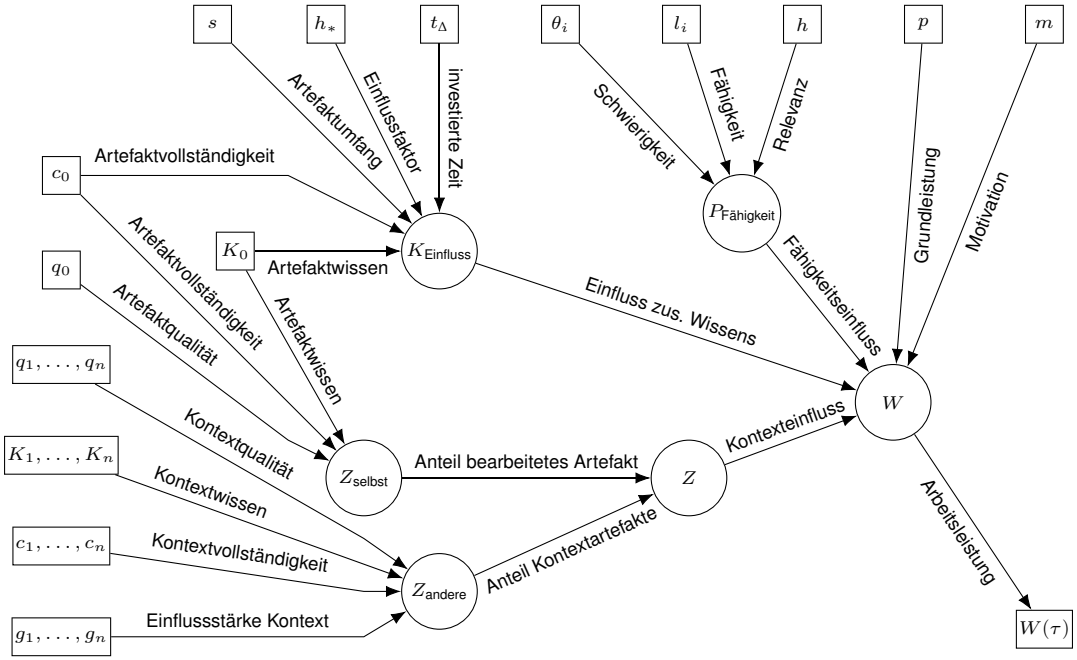


Abbildung 2.33.: Berechnung der Arbeitsleistung einer Teilaufgabe

Aufgabe durchführen soll, muss sie auch kennen. Das bedeutet, dass er Wissen über die relevanten Dokumente besitzen muss, welches er sich vor oder während der Aufgabenbearbeitung gegebenenfalls aneignen muss. Um diese Aspekte zu realisieren, werden Qualität und Vollständigkeit der Dokumente, sowie das Wissen des Bearbeiters darüber, wie in Formel 2.22 gezeigt, zu einem Einflussfaktor kombiniert.

$$
\begin{aligned}
&Z(q_0, q_1, \ldots, q_n, c_0, c_1, \ldots, c_n, g_1, \ldots, g_n, K_0, K_1, \ldots, K_n) = Z_{\text{selbst}} \cdot Z_{\text{andere}} \\
&k_{\Delta i} = \frac{|K_i \cap \{[0, c_i]\}|}{c_i} \\
&Z_{\text{selbst}} = \left(q_0 \cdot c_0 \cdot 1.42 + (1 - c_0)\right)^2 \cdot F_k\left(k_{\Delta 0}\right) \\
&Z_{\text{andere}} = \sqrt[n]{\prod_{i=1}^{n} \Big(\left(q_i \cdot c_i \cdot 1.42\right)^2 \cdot F_k\left(k_{\Delta i}\right) \cdot g_i + (1 - g_i) \Big)}
\end{aligned}
$$

$$
\begin{aligned}
Z &: \text{Kontexteinflussfaktor} \\
Z_{\text{selbst}} &: \text{Einfluss des zu bearbeitenden Artefakts} \\
Z_{\text{andere}} &: \text{Einfluss der Kontextartefakte} \\
k_{\Delta i} &: \text{Wissensabdeckung für das } i\text{te Artefakt} \\
q_0 &: \text{Qualität des zu bearbeitenden Artefakts} \\
q_1, \ldots, q_n &: \text{Qualität der Kontextartefakte} \\
c_0 &: \text{relative Vollständigkeit des zu bearbeitenden Artefakts} \\
c_1, \ldots, c_n &: \text{relative Vollständigkeit der Kontextartefakte} \\
g_1, \ldots, g_n &: \text{Stärke des Einflusses der Kontextartefakte} \\
K_0 &: \text{Wissen über das zu bearbeitenden Artefakts} \\
K_1, \ldots, K_n &: \text{Wissen über die Kontextartefakte} \\
F_k &: \text{aktivitätsspezifische Funktion zum Einfluss der Wissensabdeckung}
\end{aligned}
\tag{2.22}
$$

Der Einflussfaktor, der aus dem schon fertiggestellten Teil des zu bearbeitenden Artefakts resultiert, wird anhand dessen Qualität und des Wissens des Bearbeiters über diesen schon existierenden Teil berechnet. Dabei wird berücksichtigt, wie weit das Artefakt schon fortgeschritten ist. Wurde das Artefakt noch nicht bearbeitet, d. h. $c_0 = 0$, so hat die Qualität keinen Einfluss. Ist das Artefakt quasi vollständig, wird der größtmögliche Einfluss erreicht. Dadurch wird verhindert, dass ein vom Umfang her minimaler Teil zu Beginn des Bearbeitungsprozesses unverhältnismäßig hohe Auswirkungen auf den Rest der Artefaktbearbeitung hat, was schlussendlich zu einem negativen Wirkungszyklus führen würde, wenn dieser Teil eine niedrige Qualität aufweist.

Es wird angenommen, dass ein Qualitätsniveau von 0.7, das laut Definition in Abschnitt 2.3.1 der Qualität eines durchschnittlichen Dokuments entspricht, weder positive noch negative Auswirkungen hat. Entsprechend wird das Qualitätsniveau mit einem Faktor von 1.42 skaliert. Ein niedrigeres Qualitätsniveau hat negative Auswirkungen, wohingegen sich ein höheres Niveau positiv auf die Arbeitsleistung auswirkt. Um die negativen Wirkungen sehr niedriger und sehr hoher Qualität zu verstärken, geht der Faktor quadratisch in den Einfluss ein.

Um den Einfluss des Wissens über das zu bearbeitende Artefakte zu berücksichtigen, wird zunächst berechnet, über welchen Anteil des in diesem Artefakt dokumentierten Wissens der Bearbeiter verfügt. Ist ein Artefakt beispielsweise zu 50% fertiggestellt, muss das entsprechende Artefaktwissen des Bearbeiters das Teilintervall $[0, 0.5]$ enthalten, um eine vollständige Abdeckung zu erreichen. Diese direkte Art der Umrechnung ist möglich, da die Abbildungsvorschrift von Artefaktinhalten auf Artefaktwissen, wie in Abschnitt 2.4.2 beschrieben, so gewählt ist, dass sich eine streng sequentielle aufsteigende Ordnung der bearbeiteten Teile ergibt. Der Einfluss der Wissensabdeckung wird mittels einer frei wählbaren aktivitätsspezifischen Funktion in einen Einflussfaktor umgerechnet. Hier haben sich Potenzfunktionen als hilfreich gezeigt. Mit Hilfe des Exponenten lässt sich gut einstellen, ob Wissen unter- oder überproportional wichtig ist. Das wiederum hängt von der Art der Aktivität ab. Während beispielsweise für einen Black-Box-Test lediglich geringes Wissen über die Schnittstellen notwendig ist, muss für einen White-Box-Test das Artefakt vollständig bekannt sein.

Der Einfluss der Kontextartefakte berechnet sich im Wesentlichen gleich wie der des zu bearbeitenden Artefakts. Sind Kontextartefakte nicht vollständig, fehlen notwendige Informationen. Im Gegensatz zum zu bearbeitenden Artefakt hat fehlende Vollständigkeit bei Kontextartefakten somit negative Auswirkungen.

Weiterhin ist die Einflussstärke der Kontextartefakte zu berücksichtigen. Durch sie lässt sich im Szenario festlegen, wie sehr sich der Zustand eines Kontextartefakts auswirkt. Bei einer maximalen Stärke von $g_i = 1$ kann ein fehlendes Kontextartefakt die vollständige Aufgabebearbeitung verhindern, bei einer Stärke von 0 hingegen hat das Artefakt keinen Einfluss.

Um den Einfluss der einzelnen Artefakte zu einem gemeinsamen Faktor zu kombinieren wir das geometrische Mittel über alle Faktoren berechnet. Im Gegensatz zu einem einfachen Produkt über alle Teilfaktoren, ist dieses Mittel von der Anzahl der Kontextartefakte unabhängig.

Neben dem Wissen über die Kontextartefakte und den schon bearbeiteten Teil des zu bearbeitenden Artefakts, kann auch das Wissen über den neu zu erstellenden Teil einen Einfluss auf die Arbeitsleistung haben. Ob und wie, hängt dabei vom entsprechenden Szenario ab, und wird über dieses definiert. Dabei kann dieses *zusätzliche Wissen* über den zu erstellenden Teil notwendig für dessen Erstellung sein, oder zusätzliche Vorteile bringen.

Beispielsweise können Informationen zu Anforderungen, die gegebenenfalls nur der Kunde besitzt, notwendig sein, um eine Anforderungsdefinition zu erstellen. Fehlen diese Informationen, kann das Artefakt nicht, oder nur langsam und in schlechter Qualität erstellt werden. Alternativ kann es auch möglich sein, dass Informationen nicht direkt zur Erstellung des Artefakts notwendig, jedoch hilfreich sind. So ist der Autor eines Artefaktteils typischerweise schneller in der Überarbeitung dieses Teils, da er sich damit schon auskennt [Drappa, 2000a]. Ähnliche Effekte sind zu beobachten, wenn ein Autor einen Produktteil mehrfach erstellt, weil beispielsweise grundlegende Änderungen umzusetzen sind, die eine Neuerstellung notwendig machen, oder ein sehr ähnlicher Aspekt für ein anderes Produkt umgesetzt werden soll.

Um diesen Aspekt zu realisieren, wird mittels Formel 2.23 ein entsprechender Einflussfaktor ermittelt. Dieser basiert auf dem Artefaktwissen des Mitarbeiters und dessen Überdeckung mit dem neu zu erstellenden Artefaktteil.

$$K_{\text{Einfluss}}(t_\Delta, K_0, c_0, s, h_*) = \begin{cases} 1 + k_+ \cdot (h_* - 1) & , h_* \geq 1 \\ 1 - (1 - k_+) \cdot (1 - h_*) & , h_* < 1 \end{cases}$$

$$k_+ = \frac{|K_0 \cap \{[c_0, c_0 + \frac{t_\Delta}{s}]\}|}{\frac{t_\Delta}{s}}$$

K_{Einfluss} : Einfluss zusätzlichen Artefaktwissens auf die Arbeitsleistung
k_+ : relativer Anteil des Artefaktwissens am zu erstellenden Artefaktteil
t_Δ : investierte Zeit in die Entwicklung des neuen Artefaktteils
K_0 : Artefaktwissen des Bearbeiters
c_0 : Vollständigkeit des Artefakts vor der Bearbeitung
s : Umfang des Artefakts
h_* : Relevanzfaktor für zusätzliches Artefaktwissen aus dem Artefakt (2.23)

Dazu wird zunächst der erzielte relative Fortschritt anhand der investierten Zeit und des Artefaktumfangs geschätzt und auf die Wissensdomäne des Artefakts abgebildet, um das Artefaktwissen zu erhalten, das durch die Aufgabe entstehen wird. Diese Schätzung kann unter Umständen vom tatsächlich benötigten Wissen abweichen, da viele Faktoren dabei nicht berücksichtigt werden. Für klein genug gewählte Zeitschritte wirkt sich diese Ungenauigkeit jedoch nur minimal auf das Ergebnis aus. Unter Einbeziehung des vorhandenen Artefaktwissens des Mitarbeiters wird die relative Überdeckung berechnet, die angibt, welchen Anteil des potentiell möglichen Wissens, der Mitarbeiter über den zu erstellenden Artefaktteil besitzt.

Dieser Wert wird mittels dem im Szenario für das zu bearbeitende Artefakt festgelegten Relevanzfaktor für zusätzliches Wissen zum eigentlichen Einflussfaktor umgerechnet. Der Faktor bestimmt, ob vorhandenes zusätzliches Wissen positive, oder fehlendes zusätzliches Wissen negative Effekte hat. Relevanzfaktoren < 1 führen zu einer Reduktion der Arbeitsleistung im Falle fehlenden Wissens. Besitzt der Bearbeiter beispielsweise nur 50% des Wissens bei einem Relevanzfaktor von 0, reduziert sich die Leistung auf 50%. Faktoren > 1 wirken sich positiv auf die Leistung aus. So bewirkt beispielsweise ein Faktor von 2 eine Verdopplung der Leistung falls der Mitarbeiter das maximale Wissen für den betrachteten Schritt besitzt.

Die beiden Faktoren für *Kontexteinfluss* und *zusätzliches Wissen* gehen, wie in Formel 2.24 gezeigt, zusammen mit der physischen Basisleistung, der Motivation und der auf Basis von Fähigkeit und Schwierigkeit berechneten kognitiven Leistung in die Funktion zur Arbeitsleistung ein, mit der die Leistung zu einer einzelnen Teilaufgabe berechnet werden kann.

$$z = Z(\cdot)$$
$$k_* = K_{\text{Einfluss}}(\cdot)$$
$$W(p, k_*, l, \theta, h, m, z) = p \cdot m \cdot z \cdot k_* \cdot P_{\text{Fähigkeit}}(\theta, l, h)$$

W : Funktion zu Berechnung der Arbeitsleistung
z : Kontexteinflussfaktor aus Formel 2.22
k_* : Einflussfaktor des zusätzlichen Artefaktwissens aus Formel 2.23
p : physische Basisleistung des Bearbeiters
l : Fähigkeitsniveau des Bearbeiters bezogen auf die auszuführende Aktivität
θ : Artefaktschwierigkeit
h : Relevanz der Fähigkeit
m : Motivationswert des Bearbeiters (2.24)

Um auf das Wertepaar (w_c, w_q) für Arbeitsumfang und dessen Qualität der Gesamtaufgabe zu kommen, werden nach der Zerlegung der Aufgabe in die einzelnen Teilaufgaben, für jede Teilaufgabe mittels Formel 2.24 zunächst die jeweiligen Umfänge und deren Qualität berechnet. Hier wird von der Annahme ausgegangen, dass sich die Arbeitsleistung sowohl in der Geschwindigkeit mit der Ergebnisse produziert werden, als auch in deren Qualität niederschlägt. Um den Umfang zu berechnen wird eine Fähigkeitsrelevanz von $h = 2.0$ angenommen. Die Basisleistung entspricht der physischen Leistung des Bearbeiters $p = p_{\text{Physis}}(\cdot)$. Um die Qualität zu erhalten, wird dieselbe Formel verwendet, jedoch mit einer Fähigkeitsrelevanz von $h = 3.0$ und einer Basisleistung von $p = p' \cdot 0.7$, wobei $p' = p_{\text{Physis}}(\cdot)$ mit $\rho = 1$ entspricht, um zu verhindern, dass ein erhöhter Zeitdruck einen positiven Einfluss auf die Qualität hat. Da die durchschnittliche Qualität entsprechend ihrer Definition bei 0.7 liegt, wird die Basisleistung, deren Durchschnitt bei 1.0 liegt, entsprechend umgerechnet. Der Motivationswert wird nach oben auf 1.0 beschränkt, da mangelnde Motivation zwar zu mehr Fehlern führt, hohe Motivation jedoch kein Fähigkeitsdefizit ausgleichen kann.

Die einzelnen Werte haben sich bei der Abstimmung der Modelle aufeinander ergeben. Die Entscheidung, dass für die Qualitätsberechnung ein höherer Wert für die Fähigkeitsrelevanz verwendet wird, basiert auf der Annahme, dass Fähigkeiten die Qualität der Ergebnisse stärker beeinflussen als die Geschwindigkeit mit der sie erarbeitet werden.

Die einzelnen Teilqualitäten werden zusätzlich mit den jeweils zugehörigen zuvor berechneten Umfängen gewichtet. So wirken sich Teilaufgaben, die einen größeren Umfang erzeugen stärker auf die Gesamtqualität aus, als Teilaufgaben, die einen geringeren Anteil am Gesamtumfang haben.

Die Teilergebnisse werden anschließend, wie in Formel 2.20 gezeigt, zur Gesamtarbeitsleistung kombiniert. Zusätzlich werden noch der *Quantitätsmodifikator* auf die Fortschrittskomponente und der *Qualitätsmodifikator* auf die Qualitätskomponente des Ergebnisses angewendet. Dabei handelt es sich um einfache aktivitätsspezifische Faktoren, mit denen die Ergebnisse multipliziert werden bevor sie auf den Artefaktzustand angewendet werden. Dadurch lassen sich leicht Unterschiede zwischen den Aktivitäten erzeugen. So kann beispielsweise ein struktureller Test im Vergleich zu einem funktionsorientierten Test eine niedrigere Geschwindigkeit, jedoch eine höhere Qualität ermöglichen, was bedeutet, dass in gleicher Zeit zwar weniger, jedoch schwieriger zu findende Defekte aufgedeckt werden. Dazu werden die Modifikatoren beispielsweise auf 0.5 für die Quantität, und auf 1.2 für die Qualität gesetzt.

Anwendung der Arbeitsleistung auf den Zustand des Artefakts

Um die tatsächliche Zustandsänderung am Artefakt vorzunehmen, wird die zuvor ermittelte Arbeitsleistung (w_c, w_q) auf das Artefakt angewendet. Dabei muss zwischen Fortschrittsaktivität und Qualitätsaktivität unterschieden werden.

Wird eine Fortschrittsaktivität auf ein Artefakt angewendet, erhöht sich dadurch dessen Vollständigkeit. Gleichzeitig kommen neue Defekte hinzu, die wiederum die Qualität des Artefakts verändern. Abbildung 2.34 zeigt den Berechnungsablauf, der die Arbeitsleistung, die mittels einer Fortschrittsaktivität erzeugt wurde, auf ein Artefakt anwendet.

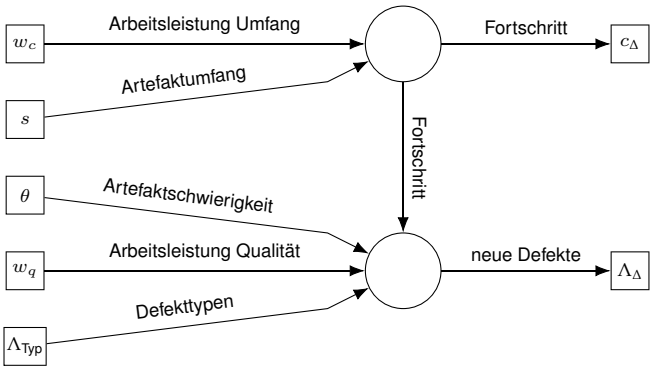


Abbildung 2.34.: Anwendung der Arbeitsleistung aus Fortschrittsaktivitäten auf ein Artefakt

Dazu wird zunächst mittels der in Formel 2.25 angegebenen Vorschrift der Fortschritt berechnet, der angibt, wie stark die Vollständigkeit des Artefakts zunimmt. Dabei wird sichergestellt, dass der Umfang des Artefakts nicht überschritten wird.

$$
\begin{aligned}
c_\Delta &= \min(c_n + w_c, s) - c_n \\
c_{n+1} &= c_n + c_\Delta
\end{aligned}
$$

c_Δ : Umfang des hinzugekommenen Artefaktteils
c_n : Vollständigkeit des Artefakts vor der Anwendung
c_{n+1} : Vollständigkeit des Artefakts nach der Anwendung
w_c : Umfang der Arbeitsleistung
s : Umfang des Artefakts

(2.25)

Anschließend werden die Defekte, die durch den Fortschrittszuwachs entstanden sind, mittels der in Formel 2.26 dargestellten Regeln ermittelt. Im ersten Schritt wird dazu der Gesamtumfang der entstandenen Defekte anhand des Fortschrittzuwachses und der Qualität der Arbeitsleistung ermittelt. Die Funktion dazu entspricht der Umkehrfunktion zur Berechnung der Qualität aus der Menge der Defekte (Formel 2.1). Damit wird erreicht, dass die Qualität des gesamten Artefakts, würde es in einem einzigen Schritt erstellt werden, genau der der Arbeitsleistung entspricht. Anschließend werden mittels einer Zufallsfunktion neue Defekte erzeugt, die in ihrer Summe dem zuvor berechneten Umfang entsprechen. Für jeden dieser Defekte wird mit einer weiteren, auf der gaußschen Normalverteilung basierenden Zufallsfunktion, dessen Schwierigkeitsgrad festgelegt. Der Erwartungswert hängt dabei von der Schwierigkeit des Artefakts ab, liegt jedoch deutlich darunter, da typischerweise mehr leicht zu findende Fehler gemacht werden, als Fehler, die sehr komplex und damit schwer zu finden sind. Der jeweilige Typ der Defekte wird ebenfalls zufällig aus der durch die Aktivität vorgegebenen Menge der Defekttypen gewählt, die angibt, welche Arten von Fehlern durch die Aktivität entstehen können. Dabei wird die zu jedem Typ angegebene Wahrscheinlichkeit bei der zufälligen Wahl berücksichtigt.

$$
\begin{aligned}
\Lambda_{\Delta+} &= \{\lambda_1, \ldots, \lambda_n\} \\
|\Lambda_{\Delta+}| &= Q^{-1}(w_q, c_\Delta) = c_\Delta \cdot (1 - w_q) \cdot 0.5 \\
\theta_{\lambda_i} &= |R_N(\theta - 0.4, 0.15)|
\end{aligned}
$$

$\Lambda_{\Delta+}$: Menge der neu entstandenen Defekte
$|\Lambda_{\Delta+}|$: Umfang der Defekte, die zum Artefakt hinzugekommen sind
c_Δ : Umfang des hinzugekommenen Artefaktteils
w_q : Qualität der Arbeitsleistung
θ_{λ_i} : Schwierigkeit des $i$ten neuen Defekts
θ : Artefaktschwierigkeit
$R_N(\mu, \sigma^2)$: Zufallsfunktion, die der gaußschen Normalverteilung folgt, mit Erwartungswert μ und Standardabweichung σ

(2.26)

Wird eine Qualitätsaktivität auf ein Artefakt angewendet, wird anhand der Arbeitsleistung berechnet, welche Defekte durch die Aktivität aufgedeckt werden. Diese Defekte werden aus dem Artefakt entfernt. Da die Defekte nicht nur gefunden, sondern auch behoben werden müssen, wird der dafür notwendige Aufwand im Artefakt hinterlegt, indem der Fortschritt des Artefakts um den Umfang der entfernten Defekte reduziert wird. Die Korrektur der Defekte kann durch anschließende Fortschrittsaktivitäten vorgenommen werden.

Die Berechnung, welche Defekte aus dem Artefakt entfernt werden, erfolgt in drei Schritten, die in Abbildung 2.35 dargestellt sind. Zunächst werden dazu alle möglichen Kandidaten aus der Menge der im Artefakt enthaltenen Defekte auf Basis der Arbeitsqualität und des Defekttyps, wie in Formel 2.27 gezeigt, ermittelt. Dabei werden alle Defekte ausgewählt, deren Typ in der Menge der durch die

Aktivität auffindbaren Defekttypen vorhanden ist, und deren Schwierigkeit höchsten der Qualität der Arbeitsleistung entspricht. Der Mitarbeiter kann Defekte somit nur finden, wenn er die dafür notwendige Leistung erbringt; das bedeutet, über die notwendigen Fähigkeiten, Motivation, physische Leistung und entsprechendes Wissen verfügt.

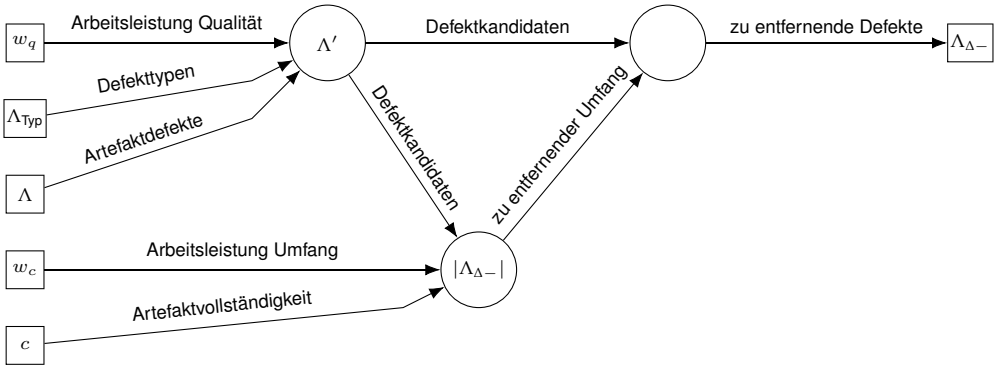


Abbildung 2.35.: Berechnung der aus einem Artefakt zu entfernenden Defekte

$$\Lambda' = \{\lambda \in \Lambda \mid \lambda_{\text{Typ}} \in \Lambda_{\text{Typ}} \wedge \lambda_\theta \leq w_q\}$$

$$\begin{aligned}
\Lambda' &: \text{Defektkandidaten} \\
\Lambda &: \text{Defekte des Artefakts} \\
\Lambda_{\text{Typ}} &: \text{Defekttypen der Aktivität} \\
w_q &: \text{Arbeitsqualität} \\
\lambda_{\text{Typ}} &: \text{Typ eines Defekts} \\
\lambda_\theta &: \text{Schwierigkeit eines Defekts}
\end{aligned} \tag{2.27}$$

Anhand der Kandidaten, des Umfangs der Arbeitsleistung und des Artefaktumfangs, wird anschließend der Umfang der zu entfernenden Defekte mittels Formel 2.28 ermittelt. Dazu wird die Defektdichte des Artefakts mit dem Umfang der Arbeitsleistung multipliziert. Je mehr Defekte in einem Artefakt vorhanden sind, umso größer ist damit auch die Wahrscheinlichkeit einen dieser Defekte zu entdecken. Mittels der Defektdichte, die als Quotient aus Kandidatenumfang und absoluter Vollständigkeit des Artefakts ermittelt wird, wird dieser Aspekt berücksichtigt. Um die Geschwindigkeit, mit der Defekte gefunden werden, in ein gutes Verhältnis zur Geschwindigkeit, mit der Fortschritt erzielt wird, zu setzen, wird das Ergebnis um Faktor 3 erhöht. Um gleichzeitig zu verhindern, dass aufgrund einer sehr hohen Defektdichte zu viele Defekte aufgedeckt werden, wird das Gesamtergebnis auf den Umfang der Arbeitsleistung beschränkt.

$$|\Lambda_{\Delta-}| = \min\left(w_c \cdot \frac{|\Lambda'|}{c} \cdot 3, w_c\right)$$

$$\begin{aligned}
|\Lambda_{\Delta-}| &: \text{Umfang der aufgedeckten Defekte} \\
|\Lambda'| &: \text{Umfang der Defektkandidaten} \\
w_c &: \text{Umfang der Arbeitsleistung} \\
c &: \text{absolute Vollständigkeit des Artefakts}
\end{aligned} \tag{2.28}$$

Im letzten Schritt werden zufällig Defekte aus der Menge der Kandidaten im Gesamtumfang des zuvor ermittelten Umfangs entfernt.

2.4.5. Motivationsmodell

Motivation beschreibt den »Zustand einer Person, der sie dazu veranlasst, eine bestimmte Handlungsalternative auszuwählen, um ein bestimmtes Ergebnis zu erreichen und der dafür sorgt, dass diese Person ihr Verhalten hinsichtlich Richtung und Intensität beibehält« [Winter, 2013]. Im Rahmen des Simulationsmodells wird Motivation als ein Maß für die Arbeitsbereitschaft eines Mitarbeiters hinsichtlich der Erledigung der ihm übertragenen Aufgaben angesehen.

In der Psychologie wird zwischen *intrinsischer* und *extrinsischer Motivation* unterschieden. Intrinsische Motivation beschreibt den Willen, sich einer Tätigkeit um ihres Selbstwillens zu widmen, zum Beispiel ein Spiel zu spielen, weil es Spaß macht. Extrinsische Motivation hingegen begründet sich im Willen ein bestimmtes Ziel durch eine bestimmte Handlungsweise zu erreichen [Myers, 2015].

Die Motivation der Mitarbeiter spiegelt sich in *Arbeitszufriedenheit* bzw. *Arbeitsunzufriedenheit* wider. Gemäß der Arbeit von Bruggemann, Groskurt und Ulrich [Bruggemann, 1975] treten diese beiden Zustände in verschiedenen Ausprägungsformen auf. Je nach Form sind die Auswirkungen auf die *Arbeitsleistung*, *Personalfluktuation* und *Fehlzeiten* unterschiedlich.

- Die **Stabilisierende Arbeitszufriedenheit** ist ein instabiler Zustand in dem sich eine Person typischerweise befindet, bevor sie in einen Zustand der stabilen Zufriedenheit wechselt. Befindet sich eine Person im Zustand der stabilisierenden Arbeitszufriedenheit, ist sie mit ihren Arbeitsbedingungen zufrieden, nimmt diese jedoch noch nicht als auf Dauer gesichert an. Eine Schwankung zwischen stabilisierender Arbeitszufriedenheit und Diffuser Arbeitsunzufriedenheit ist für eine Übergangsphase typisch.

- Die **Progressive Arbeitszufriedenheit** ist ein stabiler Zustand in dem sich eine Person befinden kann, die mit ihren Arbeitsbedingungen zufrieden ist. Eine Person, die sich in diesem Zustand befindet, nutzt den Vorteil ihrer Arbeitssituation aus, um diese noch weiter zu verbessern. Dadurch hat die Person typischerweise eine deutlich höhere Leistungsbereitschaft. Der Zustand der progressiven Arbeitszufriedenheit ist typisch für junge, engagierte und karriereorientierte Mitarbeiter.

- Die **Stabilisierte Arbeitszufriedenheit** ist ebenfalls ein stabiler Zustand in dem sich eine Person befindet, die mit ihren Arbeitsbedingungen zufrieden ist. Im Gegensatz zur progressiven Arbeitszufriedenheit, versucht die Person jedoch nicht ihre Situation weiter zu verbessern, sondern lediglich die aktuellen Bedingungen aufrecht zu erhalten. Dieser Zustand ist typisch für ältere Mitarbeiter, die mit ihrem Status zufrieden sind. Eine erhöhte Leistungsbereitschaft ist in diesem Zustand nicht zu erwarten, dafür jedoch ein sehr niedriges Kündigungsrisiko.

- Die **Diffuse Arbeitsunzufriedenheit** ist das negative Pendant zur stabilisierenden Arbeitszufriedenheit. In diesem Zustand befindet sich eine Person typischerweise bevor sie in einen stabilen negativen Zustand wechselt.

- Befindet sich eine Person im Zustand der **Resignativen Arbeitszufriedenheit**, ist sie sich ihrer negativen Arbeitsbedingungen bewusst und akzeptiert diese. Dadurch kommt es zu einer niedrigeren Leistungsbereitschaft, höheren Fehlzeiten, typischerweise jedoch zu keinem erhöhten Kündigungsrisiko.

- In den Zustand der **Pseudoarbeitszufriedenheit** kommt eine Person, die ihre schlechte Arbeitssituation verdrängt. Das subjektive Empfinden der Arbeitssituation ist demnach positiv. Dadurch kommt es zu keiner Verschlechterung der Leistungsbereitschaft, jedoch durch die psychische Belastung häufig zu erhöhten Fehlzeiten.

- Befindet sich eine Person im Zustand der **fixierten Arbeitsunzufriedenheit**, ist sie sich ihrer schlechten Arbeitssituation bewusst, unternimmt jedoch keine Anstrengung daran etwas zu ändern. Im Gegensatz zur resignativen Arbeitsunzufriedenheit, akzeptiert sie diesen Zustand jedoch nicht. Es kommt zu erhöhten Fehlzeiten, einer niedrigen Leistungsbereitschaft und einem hohen Risiko, dass diese Person bei nächster Gelegenheit die Arbeitsstelle wechselt.

- Im Gegensatz zur fixierten Arbeitsunzufriedenheit, unternehmen Personen im Zustand der **Konstruktiven Arbeitsunzufriedenheit** alles Mögliche, um ihre schlechte Arbeitssituation zu verbessern. Dadurch kommt es zu einer erhöhten Leistungsbereitschaft, weniger Fehlzeiten, und keinem erhöhten Kündigungsrisiko. Personen in diesem Zustand haben eine Produktivität, die mit Personen im Zustand der progressiven Arbeitszufriedenheit vergleichbar ist. Im Gegensatz zur Progressiven Arbeitszufriedenheit, hält der Zustand der Konstruktiven Arbeitszufriedenheit jedoch zeitlich nur beschränkt an, und es besteht ein erhöhtes Risiko, dass die betroffene Person direkt in den Zustand der fixierten Unzufriedenheit wechselt, wenn sich ihre Arbeitssituation durch die Bemühungen nicht verbessern lässt.

Jeder Projektbeteiligte befindet sich zu jedem Zeitpunkt in einem dieser acht Zustände. Ein Wechsel zwischen den Zuständen erfolgt durch Veränderungen der Bedingungen oder über die Zeit. Abbildung 2.36 zeigt eine Übersicht über die Zustände und die möglichen Übergänge, die dafür im Simulationsmodell vorgesehen sind. Zu jedem Zustand ist dessen gegebenenfalls positiver (grün) oder negativer (rot) Einfluss auf Kündigungstendenzen (F), Fehlzeiten (K) und Arbeitsleistung (L) des Mitarbeiters angegeben.

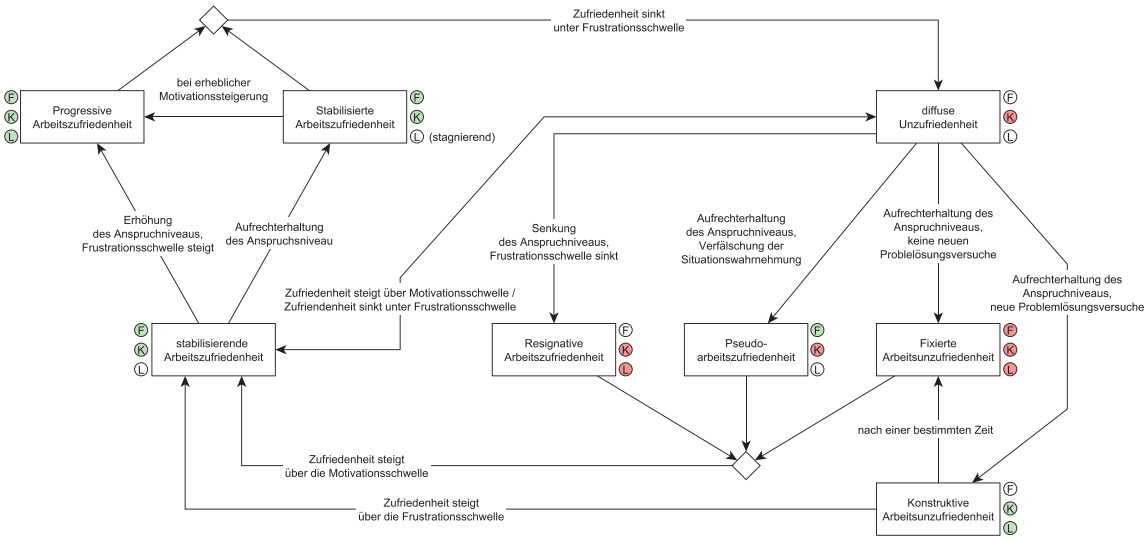


Abbildung 2.36.: Wechsel zwischen den verschiedenen Zuständen der Arbeitszufriedenheit

Um die Übergänge zwischen den einzelnen Zuständen zu steuern, muss die Zufriedenheit des betrachteten Mitarbeiters gemessen werden. Diese hängt von einer Vielzahl einzelner Faktoren ab. Die Pittsburgh-Studie [Herzberg, 1959], [Neuberger, 1974] aus den 50er Jahren identifizierte durch eine umfangreiche empirische Untersuchung eine Liste der wichtigsten Faktoren, die von Mitarbeitern zur Bewertung ihrer Arbeitssituation herangezogen werden. Diese Faktoren wurden von Herzberg et al. in der *Zwei-Faktoren-Theorie* in die zwei Gruppen *Hygienefaktoren* und *Motivatoren* aufgeteilt. Motivatoren beziehen sich auf die Motivationsschwelle, Hygienefaktoren auf die Frustrationsschwelle des Zustandsmodells der Motivation.

- Hygienefaktoren:
  - Firmenpolitik und Verwaltung
  - technische Kompetenz der Vorgesetzten
  - Arbeitsbedingungen, einschließlich Autonomie, Unterstützung und Sicherheit
  - Entlohnung und Gehalt
  - zwischenmenschliche Beziehungen zu anderen Mitarbeitern und Vorgesetzten
  - Einfluss auf das Privatleben
  - Sicherheit der Arbeitsstelle

- Motivatoren:
  - Leistung und Erfolgserlebnisse
  - Anerkennung
  - Arbeitsinhalte
  - Verantwortungsgefühl
  - Fortschritt, Aufstieg und Beförderung
  - Wachstum und persönliche Weiterentwicklung

Hygienefaktoren verhindern, wenn sie erfüllt sind, die Entstehung von Unzufriedenheit. Sie tragen jedoch nur wenig zur tatsächlich wahrgenommenen Zufriedenheit bei. Die Situation, in der alle Hygienefaktoren erfüllt sind, wird meist als selbstverständlich betrachtet. Die Faktoren treten erst in das Bewusstsein der Mitarbeiter, wenn ein entsprechender Mangel entsteht, der zu Unzufriedenheit führt.

Motivatoren haben laut Herzberg direkten Einfluss auf die Motivation der Mitarbeiter, Leistung zu erbringen. Werden Motivatoren erfüllt, führt das zu Zufriedenheit, die von den Mitarbeitern auch wahrgenommen wird. Fehlende Motivatoren führe hingegen nicht direkt zu Unzufriedenheit.

Für das Simulationsmodell werden, wie von Herzberg vorgeschlagen, zwei Maße für jeden Mitarbeiter eingeführt: der Hygienefaktor als Maß für die Unzufriedenheit, und der Motivator als Maß für die Zufriedenheit eines Mitarbeiters. In beide Maße fließen die jeweiligen Faktoren ein. Diese können durch das Charakterprofil des Mitarbeiters unterschiedlich gewichtet werden, indem für jeden Faktor der Anspruch des Mitarbeiters an den entsprechenden Faktor angegeben wird. Ein guter Ausgangspunkt für die Anspruchsverteilung bieten die Ergebnisse der Pittsburgh-Studie.

Nicht alle oben aufgeführten Faktoren lassen sich im Rahmen des Simulationsmodells sinnvoll handhaben. Daher werden einige der Faktoren in der Simulation durch das Simulationsszenario als konstant vorgegeben beziehungsweise ignoriert. Ein Faktor, der in die Simulation einfließen soll, muss zwei Kriterien erfüllen:

- Der Faktor muss aus der restlichen Simulation als Wert ableitbar sein. Die Einführung eines Faktors im Rahmen einer Planspielsimulation ergibt nur Sinn, wenn der Spieler die Simulation so steuern kann, dass sich dieser Wert vorhersagbar ändert.

- Zum Faktor muss für jeden Projektbeteiligten eine entsprechende Anspruchshaltung vorgegeben, oder während der Simulation entwickelt werden können, die mit dem abgeleiteten Wert vergleichbar ist.

Die Hygienefaktoren Firmenpolitik, Verwaltung und der Einfluss auf das Privatleben liegen außerhalb des in dieser Arbeit betrachteten Rahmens, sind somit weder Teil des Spiels, noch der Simulation, und könne daher nicht durch den Spieler beeinflusst werden. Die technische Kompetenz des Vorgesetzten – im Planspiel ist das der Spieler – ist schwer messbar. Eine entsprechende Umsetzung erfordert eine Analyse und Bewertung der Spielerhandlungen durch das Simulationsmodell. Obwohl es wünschenswert ist, dass sich die Motivation der Projektbeteiligten durch die Qualität der Spielerhandlungen beeinflussen lässt, wird der Aspekt aufgrund seiner Komplexität im Rahmen dieser Arbeit nicht behandelt.

Zu den im Rahmen des Simulationsmodells schwer zu simulierenden Motivatoren gehören Anerkennung und Verantwortungsgefühl.

Anerkennung entsteht typischerweise auf einer persönlichen Ebene. Der einzige Aspekt, den das Simulationsmodell in dieser Hinsicht enthält, ist der Sympathie/Antipathie-Bezug zwischen den Projektbeteiligten. Weitere Aspekte sind nicht umgesetzt. Das Simulationsmodell enthält keine Aspekte, die eine Interaktion auf persönlicher Ebene abdecken und für den Motivator Anerkennung ausgewertet werden können. Ein möglicher Ansatz dafür ist es dem Spieler explizit Möglichkeiten zur Verfügung zu stellen Mitarbeiter zu Loben oder zu ermahnen, um den Grad der Anerkennung des Vorgesetzten umzusetzen.

Um ein Verantwortungsgefühl wahrnehmen zu können muss man in der Lage sein, die eigene Verantwortung einzuschätzen. Dazu muss der Mitarbeiter entsprechendes zusätzliches Wissen über die Bedeutung seiner Rolle im Projekt haben. Da das nicht Teil des Simulationsmodells ist, lässt sich

Verantwortungsgefühl nicht simulieren und der dazugehörige Motivator ist nicht Teil des Motivationsmodells.

Alle weiteren Faktoren erfüllen die oben erläuterten Kriterien. Sie sind im Motivationsmodell enthalten und werden im Folgenden im Einzelnen beschriebenen. Alle Faktoren werden durch die Simulation auf den Wertebereich $[0, 1]$ abgebildet, um den Erfüllungsgrad bezogen auf die einzelnen Projektbeteiligten darzustellen.

Entlohnung und Gehalt Das Gehalt eines Mitarbeiters ist eine direkt messbare Größe, welche durch den Spieler in einem gewissen Rahmen direkt manipuliert werden kann. Als Vergleichswert für die Bewertung des eigenen Gehalts, dienen die Gehälter der anderen Mitarbeiter, und die Angebote auf dem allgemeinen Arbeitsmarkt.

Der Faktor für Entlohnung und Gehalt wird kontinuierlich aktualisiert. Dazu wird zu Beginn jeden Zeitschritts der aktuelle Wert mit der in Formel 2.29 angegebenen Funktion berechnet.

$$
\begin{aligned}
M_{\text{Lohn}}(\omega, \overline{\omega}, \overline{l}) &= \min\left(\frac{\omega}{\overline{\omega}} \cdot (\overline{l} + 0.5) \cdot 0.5, 1\right) \\
\overline{\omega} &= \sum_{i=1}^{n} \omega_i \cdot 0.5 + \bar{\omega}_{\text{Arbeitsmarkt}} \cdot 0.5
\end{aligned}
$$

$$
\begin{aligned}
M_{\text{Lohn}} &: \text{Hygienefaktor für Lohn und Gehalt} \\
\omega &: \text{Gehalt des betrachteten Mitarbeiters} \\
\omega_i &: \text{Gehalt des } i\text{ten Kollegen} \\
\overline{\omega} &: \text{durchschnittliches Gehalt} \\
\overline{\omega}_{\text{Arbeitsmarkt}} &: \text{durchschnittliches Gehalt auf dem Arbeitsmarkt} \\
\overline{l} &: \text{durchschnittliches Niveau der Fähigkeiten des betrachteten Mitarbeiters}
\end{aligned}
\tag{2.29}
$$

Der Faktor setzt das aktuelle Gehalt des Mitarbeiters in ein Verhältnis zum durchschnittlichen Gehalt seiner Umwelt. Dieses setzt sich jeweils zur Hälfte aus dem durchschnittlichen Gehalt seiner Projektkollegen und dem durchschnittlichen Gehalt auf dem gesamten Arbeitsmarkt zusammen. Der Mitarbeiter betrachtet so bei der Bewertung seines eigenen Gehalts sein direktes Umfeld, und damit, ob er im Vergleich zu seinen Kollegen fair bezahlt wird, und den größeren Kontext, um abzuschätzen, ob ihn sein Arbeitgeber insgesamt fair bezahlt, oder ob er bei einem anderen Arbeitgeber bessere Konditionen erwarten kann.

Der Arbeitsmarkt selbst ist nicht Teil des Simulationsmodells. Das durchschnittliche Gehalt, das für vergleichbare Stellen bezahlt wird, kann jedoch über eine entsprechende Schnittstelle in das Modell einfließen. Das ist insbesondere für Mehrspielerkonzepte interessant, bei denen die einzelnen Projekte gegenseitig konkurrieren und Mitarbeiter zwischen den Projekten wechseln können.

Da Mitarbeiter mit besseren Fähigkeiten auch eine bessere Bezahlung erwarten können, wird auch dieser Faktor bei der Bewertung des eigenen Gehalts berücksichtigt. Liegen die Fähigkeiten über dem Durchschnitt (> 0.5) darf ein überdurchschnittlicher Lohn erwartet werden. Liegen die Fähigkeiten unter dem Durchschnitt, muss mit einem niedrigeren Lohn gerechnet werden. Das Simulationsmodell sieht dafür einen Spielraum von 50% bis 200% bezogen auf den Durchschnittslohn vor. Bei einem durchschnittlichen Beispielgehalt Gehalt von 50.000€ eröffnet das eine Gehaltsspanne zwischen 25.000€ und 100.000€.

Sicherheit der Arbeitsstelle Die Sicherheit der Arbeitsstelle sagt aus, wie hoch die Wahrscheinlichkeit ist, dass ein Mitarbeiter entlassen wird. Die Sicherheit der Arbeitsstelle lässt sich im beschränkten Maße anhand der Zahl der entlassenen Mitarbeiter ableiten. Mitarbeiter die von sich aus gekündigt haben dürfen dabei nicht berücksichtigt werden. Werden Kollegen entlassen, verursacht das auch Angst unter den verbleibenden Mitarbeitern, da Entlassungen oft bedeuten, dass Personal abgebaut werden muss. Die subjektive Einschätzung der Sicherheit des eigenen Arbeitsplatzes verschlechtert sich dadurch. Die Entlassung eines Kollegen ist jedoch nur ein sehr schwaches Maß für die Sicherheit des eigenen Arbeitsplatzes, da zur Bewertung noch viele andere Faktoren, wie beispielsweise die

Unternehmenssituation und -politik herangezogen werden müssen. Da diese Aspekte nicht Teil des Simulationsmodells sind, beschränkt sich die Bewertung der Sicherheit der Arbeitsstelle auf die Entlassungen.

Die Sicherheit wird durch einen Hygienefaktor repräsentiert. Dessen Wert steigt kontinuierlich um 0.0015 pro Tag, und sinkt mit jeder Entlassung um Teamgröße^{-1}, wie in Formel 2.30 gezeigt.

$$
\begin{aligned}
m_{\text{Sicherheit}_{n+1}} &= \min\left(m_{\text{Sicherheit}_n} + 0.0015 \cdot t_\Delta, 1\right) \\
m_{\text{Sicherheit}} &= \max\left(m'_{\text{Sicherheit}} - \frac{1}{|B|}, 0\right)
\end{aligned}
$$

$$
\begin{aligned}
m_{\text{Sicherheit}_n} &: \text{Faktorwert nach Zeitschritt } n \\
t_\Delta &: \text{Dauer des Zeitschritts in Tagen} \\
m_{\text{Sicherheit}} &: \text{Faktorwert nach der Entlassung} \\
m'_{\text{Sicherheit}} &: \text{Faktorwert vor der Entlassung} \\
B &: \text{Menge der Projektbeteiligten vor der Entlassung}
\end{aligned}
\tag{2.30}
$$

Damit fühlen sich die Mitarbeiter umso sicherer, je länger niemand entlassen wurde, und je weniger Kollegen gehen mussten. Je kleiner die Teamgröße ist, umso größer ist auch die Wahrscheinlichkeit, dass eine Entlassung einen bestimmten Mitarbeiter betrifft. Entsprechend sinkt der Wert bei Entlassungen in kleinen Teams schneller, während bei größeren Teams eine entsprechende Fluktuation im Personal als normal angesehen, und damit nicht so negativ bewertet wird.

Der Spieler, in seiner Rolle als Vorgesetzter muss deshalb darauf achten, keine zu große Fluktuation bei seinen Mitarbeitern zu erzeugen, und nicht zu viele Mitarbeiter auf einmal zu entlassen, da dies einen negativen Einfluss auf die Motivation der verbleibenden Mitarbeiter haben kann.

Zwischenmenschliche Beziehungen Der Hygienefaktor für zwischenmenschliche Beziehungen gibt an, wie gut ein Projektbeteiligter mit den anderen Projektbeteiligten auskommt, und wie wohl er sich dabei fühlt. Das Simulationsmodell abstrahiert die Beziehung zwischen zwei Projektbeteiligten mittels eines Werts zwischen 0 und 1, der die Sympathie/Antipathie angibt. Jeder Projektbeteiligte verfügt für jeden seiner Kollegen über einen solchen Wert.

Der Hygienefaktor eines Mitarbeiters ergibt sich aus der durchschnittlichen Sympathie/Antipathie zu allen anderen Kollegen, und gibt damit an, wie wohl der Mitarbeiter sich in seinem kollegialen Umfeld fühlt. Mittels der Funktion des beschränkten Wachstums, wird der Hygienefaktor, wie in Formel 2.31 gezeigt, über die Zeit in Richtung des durchschnittlichen Sympathie/Antipathie-Wertes entwickelt. Das bedeutet, solange er nicht anderweitig beeinflusst wird nähert er sich immer weiter diesem Wert an. Diese Änderung wird in jedem Zeitschritt der Simulation durchgeführt.

$$
m_{\text{Beziehung}_{n+1}} = m_{\text{Beziehung}_n} + (\overline{\psi_{\text{m}}} - m_{\text{Beziehung}_n}) \cdot t_\Delta \cdot 0.1
$$

$$
\begin{aligned}
m_{\text{Beziehung}_n} &: \text{Hygienefaktor für zwischenmenschliche Beziehung nach Zeitschritt } n \\
\overline{\psi_{\text{m}}} &: \text{durchschnittliche Sympathie/Antipathie} \\
t_\Delta &: \text{Dauer des Zeitschritts in Tagen}
\end{aligned}
\tag{2.31}
$$

Neben dem allgemeinen kollegialen Arbeitsumfeld haben vor allem die Kollegen, mit denen der Mitarbeiter eng zusammenarbeitet, einen Einfluss auf den Hygienefaktor. Beispielsweise kann, ausgehend von einer guten Situation innerhalb des Entwicklungsteams, auch ein unliebsamer Kunde, mit dem der betrachtete Mitarbeiter eng zusammenarbeiten muss, den Faktor erheblich senken. Um solche Einflüsse zu berücksichtigen, werden die Aufgaben, welche die Mitarbeiter bearbeiten, betrachtet. Bearbeitet ein Mitarbeiter eine Aufgabe gemeinsam mit einem anderen Mitarbeiter, wird dessen Sympathie/Antipathie-Wert verwendet um den Hygienefaktor zu verändern. Dazu wird die Funktion des beschränkten Wachstums, wie in Formel 2.32 gezeigt, für jeweils jeden an einer Aufgabe mitbeteiligten Kollegen angewendet. Diese Änderung wird für jede bearbeitete Aufgabe durchgeführt.

$$m_{\text{Beziehung}_{n+1}} = m_{\text{Beziehung}_n} + (\psi_{\text{m}} - m_{\text{Beziehung}_n}) \cdot 0.04 \cdot t_\Delta$$

$$\begin{aligned} \psi_{\text{m}} &: \text{Sympathie/Antipathie für den Kollegen} \\ t_\Delta &: \text{Dauer der Aufgabe in Stunden} \end{aligned} \tag{2.32}$$

Neben den beiden beschriebenen Aspekten ist auch die zwischenmenschliche Beziehung zum Vorgesetzten relevant. Da der Vorgesetzte durch den Spieler repräsentiert wird, und der Wert Sympathie/Antipathie-Wert für diesen nicht bekannt ist und es auch nicht sinnvoll scheint, ihn über das Szenario zu definieren, wird dieser Aspekt im Simulationsmodell nicht berücksichtigt.

Arbeitsbedingungen Die meisten Aspekte welche die Arbeitsbedingungen beeinflussen, wie beispielsweise die Ausstattung des Arbeitsplatzes, sind nicht Teil der Simulation. Zu den Arbeitsbedingungen gehört jedoch auch die Gestaltung der Arbeitszeit, die wiederum Teil des Simulationsmodells ist, das Arbeitszeit, Überstunden und Zeitdruck erfasst und beeinflusst. Näheres dazu findet sich im Abschnitt 2.6.2 zur Handlungsplanung der Agenten.

Aus dem Vergleich der tatsächlichen Arbeitszeit mit der vertraglich vereinbarten Arbeitszeit kann ein Wert für die Arbeitsbedingungen in Bezug auf die Arbeitszeit abgeleitet werden. Ebenso kann der Zeitdruck, der auf einem Mitarbeiter lastet gemessen werden, und somit in die Bewertung der Arbeitsbedingungen einfließen. Damit deckt der Hygienefaktor im Simulationsmodells zwei wichtige Aspekte der Arbeitsbedingungen ab.

Der Faktor für die Arbeitsbedingungen wird kontinuierlich anhand des *Arbeitszeitkontos* und des aktuellen, aus dem Projektplan zu ermittelnden *Zeitdruck* aktualisiert.

Formel 2.33 zeigt die Berechnung der Veränderung durch Überstunden. Da Überstunden nur für einen vollständigen Arbeitstag sinnvoll betrachtet werden können, wird diese Aktualisierung nach jedem Arbeitstag durchgeführt.

$$m_{\text{Arbeitsbedingung}_{n+1}} = m_{\text{Arbeitsbedingung}_n} + (1 - m_{\text{Arbeitsbedingung}_n} - t_{\Delta\,\text{Überstunden}}) \cdot 0.001$$

$$\begin{aligned} m_{\text{Arbeitsbedingung}_n} &: \text{Hygienefaktor der Arbeitsbedingung nach Tag } n \\ t_{\Delta\,\text{Überstunden}} &: \text{Über- bzw. Minusstunden die am Tag } n+1 \text{ gemacht wurden} \end{aligned} \tag{2.33}$$

Die Berechnung besteht aus einer Komponente, die den Faktor mittels einer Funktion des beschränkten Wachstums über die Zeit erhöht. Da die Schrittlänge hier fest auf 24 Stunden festgelegt ist, wird kein Parameter für die Zeit benötigt. Die zweite Komponente reduziert den Faktor entsprechend der Überstunden.

Formel 2.34 zeigt den Einfluss des Zeitdrucks auf den Hygienefaktor. Diese Berechnung wird zu Beginn jeden Zeitschritts für den gesamten Zeitschritt durchgeführt.

$$m_{\text{Arbeitsbedingung}_{n+1}} = m_{\text{Arbeitsbedingung}_n} + (1 - \min(\rho, 3)) \cdot 0.0001$$

$$\begin{aligned} m_{\text{Arbeitsbedingung}_n} &: \text{Hygienefaktor der Arbeitsbedingung nach Zeitschritt } n \\ \rho &: \text{aktueller Zeitdruck} \\ t_\Delta &: \text{Länge des Zeitschritts } n+1 \text{ in Stunden} \end{aligned} \tag{2.34}$$

Je höher der Zeitdruck ist, und je länger er anhält, umso stärker wird der Hygienefaktor verringert. Um übermäßige Einflüsse zu verhindern, wird der Zeitdruck auf maximal 3 beschränkt. Da der Zeitdruck nicht unter 0 fallen kann, ist auch die Verbesserung des Hygienefaktors durch fehlenden Zeitdruck entsprechend beschränkt.

Persönliches Wachstum Der Motivator zum persönlichen Wachstum beschreibt die berufliche Entwicklung des Mitarbeiters. Diese Entwicklung findet typischerweise über einen längeren Zeitraum statt. Da das Simulationsmodell nur auf ein einzelnes Projekt fokussiert ist, wird nur ein vergleichsweise kurzer Zeitabschnitt behandelt. Ein Mitarbeiter kann sich zwar in diesem Zeitraum entwickeln und

persönliche Fortschritte machen, um diese jedoch sinnvoll bewerten zu können ist ein eine Betrachtung über einen längeren Zeitraum notwendig.

Zum persönlichen Wachstum gehört auch die Weiterentwicklung von Wissen und Fähigkeiten. Dieser Aspekt wird vom Simulationsmodell abgedeckt und kann somit für einen entsprechenden Motivator ausgewertet werden. Dazu wird der Motivator nach jeder ausgeführten Aufgabe anhand der durch sie gewonnenen Fähigkeiten, wie in Formel 2.35 gezeigt, aktualisiert.

$$
\begin{aligned}
&m_{\text{Wachstum}_{n+1}} = m_{\text{Wachstum}_n} + (l_\Delta - t_\Delta \cdot 0.0001) \cdot 2 \\
&m_{\text{Wachstum}_n} : \text{Motivator Wachstum vor Ausführung der Aufgabe} \\
&\qquad l_\Delta : \text{durch die Aufgabe gewonnene Fähigkeiten} \\
&\qquad t_\Delta : \text{Dauer der Aufgabe}
\end{aligned}
\tag{2.35}
$$

Dabei wird die tatsächliche Steigerung des Fähigkeitsniveaus mit einer Erwartung von 0.0001 pro Stunde verglichen. Liegt der Zuwachs über der Erwartung, steigt der Motivator, ansonsten sinkt er. Die Skalierung der Änderung mit dem Faktor 2 ist notwendig, da sich der Motivator ohne ihn innerhalb einer typischen Projektlaufzeit nicht stark genug verändert, um einen sichtbaren Einfluss auf die Motivation zu haben.

Leistung und Erfolgserlebnisse Leistung und Erfolgserlebnisse wurde in der Pittsburgh-Studie als der Motivator mit dem allergrößten Einfluss auf die Motivation der Mitarbeiter identifiziert. Dieser Faktor beschreibt die Ergebnisse der durchgeführten Aufgaben im Vergleich zu den Erwartungen, die der Mitarbeiter an diese stellt.

Die Leistung eines einzelnen Mitarbeiters oder einer Gruppe von Mitarbeitern, die gemeinsam an einer Aufgabe arbeiten, kann direkt über die Veränderung der Artefakte gemessen werden. Dazu wird der Vollständigkeitsgrad eines Artefakts und dessen Qualität bzw. der Umfang der Defekte vor und nach der Durchführung einer Aufgabe gemessen. Die Differenz kann anschließend mit den Erwartungen des Mitarbeiters verglichen werden. Der Erwartungswert basiert auf den eigenen Erfahrungen des Mitarbeiters und dem allgemeinen Anspruch an die Aufgabe.

Formel 2.36 zeigt die Berechnung zur Aktualisierung des Motivators anhand der Ergebnisse einer Aufgabendurchführung. Dazu werden drei einzelne Aspekte berechnet, die zu einem neuen Zielwert kombiniert werden, in dessen Richtung sich der Motivator mittels der Funktion des beschränkten Wachstums entwickelt.

$$
\begin{aligned}
&F_v = \frac{v}{v'} \\
&F_\theta = v \cdot (\theta + 0.5) \\
&F_q = q - 0.7 \\
&m_{\text{Leistung}_{n+1}} = m_{\text{Leistung}_n} + (F_v \cdot 0.3 + F_\theta \cdot 0.7 + F_q - 0.5 - m_{\text{Leistung}_n}) \cdot 0.001 \cdot t_\Delta \\
&m_{\text{Leistung}_{n+1}} : \text{Motivator Leistung und Erfolg nach Ausführung der Aufgabe} \\
&\qquad F_v : \text{Einfluss anhand der erwarteten Arbeitsgeschwindigkeit} \\
&\qquad F_\theta : \text{Einfluss anhand der Schwierigkeit} \\
&\qquad F_q : \text{Einfluss anhand der Qualität} \\
&\qquad v : \text{bei der Bearbeitung wahrgenommene Arbeitsgeschwindigkeit} \\
&\qquad v' : \text{auf Basis der Erfahrung erwartete Arbeitsgeschwindigkeit} \\
&\qquad \theta : \text{Schwierigkeit der Aufgabe} \\
&\qquad q : \text{Qualität des Aufgabenergebnisses} \\
&\qquad t_\Delta : \text{Dauer der Aufgabe in Stunden}
\end{aligned}
\tag{2.36}
$$

Um die Aspekte unabhängig der Aufgabendauer zu berechnen, wird anstelle des Fortschritts, der durch die Aufgabe erzeugt wurde, die dazugehörige Arbeitsgeschwindigkeit verwendet.

Der erste Aspekt setzt die Arbeitsgeschwindigkeit ins Verhältnis zur Erwartung des betrachteten Mitarbeiters, die auf seinen Erfahrungen basiert.

Der zweite Aspekt setzt die Arbeitsgeschwindigkeit in das Verhältnis zum allgemeinen Durchschnitt, berücksichtigt dabei jedoch die Schwierigkeit der Aufgabe. Dazu wird angenommen, dass schwerere Aufgaben länger dauern, und dass das Lösen schwerere Aufgaben als größeren Erfolg angesehen wird. Der Schwierigkeitsgrad wird dazu als Korrekturfaktor auf die Arbeitsgeschwindigkeit angewendet.

Der dritte Aspekt berücksichtigt die Qualität der Arbeitsleistung. Dabei wird von einem Durchschnittswert von 0.7 ausgegangen. Durch eine höhere Qualität lässt sich der Motivator verbessern, eine niedrigere Qualität verschlechtert ihn.

Die Faktoren werden so zusammengesetzt, dass ihre Summe für einen neutralen Rahmen 1 entspricht. Ein neutraler Rahmen bedeutet eine Aufgabe mit der Schwierigkeit von 0.5, eine Ergebnisqualität von 0.7 und eine Arbeitsgeschwindigkeit von 1, bei einer Erwartung der Arbeitsgeschwindigkeit von 1. Da der Durchschnittswert eines Motivators bei 0.5 liegt, wird der kombinierte Faktor zusätzlich um diesen Offset verschoben.

Die getrennte Berechnung von Bezug zur persönlichen Erwartung und zur allgemeinen Arbeitsgeschwindigkeit von 1 ist dabei notwendig. Würden sich beide Faktoren auf die persönliche Erwartung beziehen, wäre die Bewertung zu sehr subjektiv und der Mitarbeiter würde sich nur an sich selbst orientieren. Sein Umfeld und deren Leistungen würden dabei nicht in die Bewertung seiner eigenen Leistung einfließen. Würde der Mitarbeiter sich hingegen nur an seiner Umwelt orientieren, besteht die Gefahr, dass persönliche Erfolge untergehen, wenn die Umwelt grundsätzlich besser ist. Ebenso würden persönliche Misserfolge trotzdem als Erfolg gewertet, wenn der Mitarbeiter immer noch besser als der Durchschnitt ist.

Kommunikationsaktivitäten bewirken keine Änderungen an Artefakten und können deshalb nicht anhand dieser Änderungen bewertet werden. Um die Leistungsbewertung einer Kommunikationsaktivität mit Formel 2.36 zu berechnen, wird anstelle der Arbeitsgeschwindigkeit die Kommunikationsgeschwindigkeit verwendet, für die Qualität wird der neutrale Wert von $q = 0.7$, für die Schwierigkeit der neutrale Wert von $\theta = 0.5$ angenommen.

Außerdem muss noch berücksichtigt werden, dass es möglich ist, dass ein Mitarbeiter am Arbeitsplatz ist, jedoch keine Aufgabe durchführt. In dieser Zeit erbringt er keine Leistung und der Motivator sinkt deswegen um 0.0004 pro Stunde, wie in Formel 2.37 gezeigt. Dieser Wert hat sich als passend herausgestellt, da kurzfristige Unterbeschäftigung so keinen nennenswerten Auswirkungen haben, ein Mitarbeiter, der jedoch über ganze Projektphasen keine Aufgaben zugeteilt bekommt, entsprechend darauf reagiert.

$$
\begin{aligned}
&m_{\text{Leistung}_n+1} = m_{\text{Leistung}_n} - 0.0004 \cdot t_\Delta \\
&t_\Delta : \text{Dauer der unproduktiven Zeit in Stunden}
\end{aligned}
\tag{2.37}
$$

Arbeitsinhalte Jeder Projektbeteiligte hat typischerweise Vorlieben und Abneigungen für, beziehungsweise gegen bestimmte Aufgaben. Diese Einstellung korreliert oft mit den Fähigkeiten der Person, kann jedoch auch davon abweichen. Die Einstellung eines Mitarbeiters ist über seinen Charakter definiert, und wird mittels eines Werts für Sympathie/Antipathie bezüglich der einzelnen Aktivitäten festgelegt. Dieser Wert stellt das Pendant zum Sympathie/Antipathie-Wert für Kollegen dar. Der Motivator wird ähnlich wie der Hygienefaktor für die zwischenmenschliche Beziehung berechnet. Formel 2.38 zeigt, wie sich die Durchführung eine konkrete Aufgabe auf den Motivator für Arbeitsinhalte auswirkt.

$$
\begin{aligned}
&m_{\text{Arbeitsinhalte}_{n+1}} = m_{\text{Arbeitsinhalte}_n} + (\overline{\psi_a} - m_{\text{Arbeitsinhalte}_n}) \cdot 0.0015 \cdot t_\Delta \\
&m_{\text{Arbeitsinhalte}_n} : \text{Motivator für Arbeitsinhalte nach Zeitschritt } n \\
&\overline{\psi_a} : \text{durchschnittliche Sympathie/Antipathie} \\
&t_\Delta : \text{Dauer der betrachteten Aufgabe}
\end{aligned}
\tag{2.38}
$$

Während alle anderen Faktoren die extrinsische Motivation ansprechend, bezieht sich der Motivator Arbeitsinhalte vor allem auf die intrinsische Motivation.

Eine Besonderheit bei der Bewertung der Arbeitsinhalte betrifft die Zeit, die ein Mitarbeiter im Büro sitzt, jedoch nichts zu tun hat. Unterforderung im Beruf, auch als *Boreout* bezeichnet, wird erst

seit kurzen wissenschaftlich genauer untersucht. Aktuellen Erkenntnissen zur Folge, sind die negativen Auswirkungen von Unterforderung auf Motivation und Wohlbefinden nicht zu unterschätzen [Prammer, 2012]. Dieser Aspekt ist im Simulationsmodell als Verringerung des Motivators für Arbeitsinhalte, in Abhängigkeit der Zeitabschnitte, in denen Mitarbeiter unterfordert sind, realisiert. Dazu wird der Motivator, wie in Formel 2.39 gezeigt, solange um 0.001 pro Stunde reduziert, wie der Mitarbeiter nichts zu tun hat. Der Spieler muss daher auch darauf achten, seine Mitarbeiter genügend auszulasten, um deren Motivation aufrecht zu erhalten.

$$m_{\text{Arbeitsinhalte}_n+1} = m_{\text{Arbeitsinhalte}_n} - 0.001 \cdot t_\Delta$$

t_Δ : Dauer der unproduktiven Zeit in Stunden (2.39)

Zustandsänderungen

Das Simulationsmodell realisiert die verschiedenen Arbeitszustände und deren Übergänge mit einem Zustandsautomaten, der die oben beschriebenen Zustände der Arbeitszufriedenheit nach Groskurt et al. enthält. Im Gegensatz zu den anderen Elementarmodellen ist das Motivationsmodell somit zustandsbehaftet, und muss für jeden einzelnen Projektbeteiligten instanziiert werden. Übergänge werden durch die Motivatoren und Hygienefaktoren gesteuert, die mittels der charakterspezifischen Anspruchsfaktoren, wie in Formel 2.40 gezeigt, zu jeweils einem gemeinsamen Wert für Motivation und Hygiene kombiniert werden. Dazu wird der Anspruch des Mitarbeiters an den jeweiligen Faktor als Grenze für die vollständige Erfüllung des Faktors angenommen. Der Erfüllungsgrad ergibt sich aus dem Verhältnis zwischen Faktor und Anspruch.

$$m_{M/H} = \sqrt[n]{\prod_{i=1}^{n} \frac{m_i}{g_i}}$$

$m_{M/H}$: gemeinsamer Wert der Hygienefaktoren bzw. Motivatoren
m_i : $i$ter Hygienefaktor bzw. Motivator
g_i : Anspruch an den $i$ten Faktor
n : Anzahl der Faktoren (2.40)

Motivatoren und Hygiene spannen einen zweidimensionalen Zustandsraum auf. Der aktuelle Hygiene- und Motivatorwert des Mitarbeiters beschreibt darin einen Punkt. Verändern sich die einzelnen Faktoren, verschiebt sich dieser Punkt im Zustandsraum. Das Simulationsmodell bildet die Zustände des Zustandsautomaten als Polygone in diesem Zustandsraum ab, und unterteilt ihn damit in die einzelnen Zustände, die sich auch überlappen können. Dadurch wird jedem Zustand eine Menge von Punkten aus dem Gesamtraum, und ein konkreter Punkt im Zustandsraum, einem oder mehreren Zuständen zugeordnet.

Abbildung 2.37 zeigt eine Visualisierung des Zustandsraums. Die beiden Achsen entsprechen dem gemeinsamen Wert von Hygienefaktoren und Motivatoren. Die farbigen Flächen zeigen die von den Zuständen eingenommenen Teile des Zustandsraums, die sich teilweise erheblich überlappen. Der rote Punkt symbolisiert beispielhaft die aktuellen Werte von Motivatoren und Hygiene. Des Weiteren ist angegeben, welche Bereiche auf welche Zustände abgebildet werden. In Anhang A.3 findet sich die genaue Definition der einzelnen Zustände.

Anhand der Werte für Hygiene und Motivation können dem betrachteten Mitarbeiter so ein oder mehreren Zuständen zugeordnet werden. Ein Mitarbeiter hat zu jedem Zeitpunkt jedoch nur genau einen Motivationszustand. Sein Anfangszustand wird durch das Anfangsszenario vorgegebenen. Verschiebt sich der Punkt durch die Einflüsse des Projektverlaufs im Zustandsraum, so ändern sich der Zustand gemäß folgenden Regeln:

1. Der aktuelle Zustand bleibt so lange wie möglich erhalten. Ein Zustandswechsel tritt daher erst ein, wenn der Punkt den Teilbereich, der den aktuellen Zustand definiert, verlässt.

2. Ein Zustandswechsel kann nur über die im Zustandsautomaten definierten Übergänge stattfinden. Verlässt der Punkt den Teilraum, der den aktuellen Zustand definiert, und liegt seine

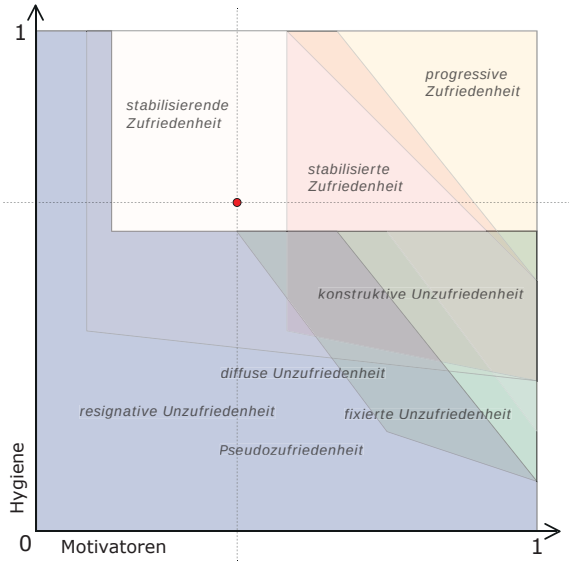


Abbildung 2.37.: Visualisierung des Zustandsraums im Motivationsmodell

neue Position nur in Teilräumen, die aufgrund des Übergangsmodells vom aktuellen Zustand aus nicht erreicht werden können, so bleibt der aktuelle Zustand erhalten.

3. Wechselt der Punkt in einen neuen Zustand der vom aktuellen Zustand aus erreicht werden kann, und trifft 1. nicht zu, wird dieser Zustand zum neuen aktuellen Zustand.

4. Liegt der Punkt in Teilbereichen mehrerer Zustände, und gelten die Regeln 1-3 für diese Zustände, so ist der Übergang nicht mehr deterministisch. In diesem Fall wird zufällig, entsprechend einer gegebenen Wahrscheinlichkeitsverteilung, ein neuer Zustand ausgewählt.

Für den Fall eines nichtdeterministischen Übergangs, wird der neue Zustand unter den möglichen Folgezuständen zufällig ausgewählt. Dabei ist die Wahrscheinlichkeit, dass ein bestimmter Zustand gewählt wird, im Allgemeinen nicht über alle Zustände hinweg gleich, sondern folgt einer bestimmten Verteilung, die auf Basis der für jeden Übergang zusätzlich festgelegten Wahrscheinlichkeit berechnet wird. So kann über die Definition der Wahrscheinlichkeiten beeinflusst werden, ob ein Charakter beispielsweise eher dazu neigt, bei Problemen in eine resignative oder konstruktive Arbeitsunzufriedenheit überzugehen. Zusätzlich wird noch die Tendenz der Zufriedenheit bei der Auswahl der Zustände berücksichtigt. So ist es beispielsweise unwahrscheinlich, dass eine Person in einen negativeren Zustand wechselt, solange sich deren Zufriedenheitsindikatoren stetig verbessern. Zustände, die der Tendenz entsprechen, werden bevorzugt.

Um zu verhindern, dass bei kleinen Simulationsschritten und Fluktuationen der Hygienefaktoren und Motivatoren ein zu häufiger unrealistischer Zustandswechsel passiert, ist das Modell auf einen Wechsel pro Woche beschränkt.

Berechnung von Kenngrößen

Anhand des Zustands und der Werte von Hygiene und Motivation können Kenngrößen für *Leistung*, *krankheitsbedingte Ausfälle* und *Kündigung* berechnet werden. Leistung gibt dabei einen Faktor an, der festlegt, wie sich die Grundleistung des Mitarbeiters durch den Einfluss der Motivation verändert. Die anderen beiden Werte können als Faktoren eingesetzt werden, um eine Grundwahrscheinlichkeit für krankheitsbedingte Ausfälle, beziehungsweise eine Kündigung von Seiten des Mitarbeiters zu beeinflussen.

Der Leistungseinfluss berechnet sich, wie in Formel 2.41 gezeigt, anhand des Zustands, der Hygiene und der Motivatoren. Dieser Wert fließt in die Berechnung des Arbeitsfortschritts ein. Dabei fließt

bei den positiven Zuständen sowohl Motivator als auch Hygienefaktor ein, während in die negativen Zustände nur der Hygienefaktor berücksichtigt wird; es müssen zuerst die grundlegenden Bedürfnisse erfüllt werden, bevor zusätzliche Motivationsmaßnahmen Wirkung zeigen können.

$$m_{\text{Leistung}} = \begin{cases} \max(m_H + m_M, 0.1) & \text{Stabilisierte Arbeitszufriedenheit} \\ \max(m_H + m_M, 0.1) & \text{Stabilisierende Arbeitszufriedenheit} \\ \max(m_H + m_M \cdot 1.25, 0.1) & \text{Progressive Arbeitszufriedenheit} \\ \max(m_H + m_M \cdot 1.25, 0.1) & \text{Konstruktive Arbeitsunzufriedenheit} \\ \max(m_H \cdot 0.75, 0.1) & \text{Diffuse Arbeitsunzufriedenheit} \\ \max(m_H \cdot 0.75, 0.1) & \text{Fixierte Arbeitsunzufriedenheit} \\ \max(m_H \cdot 0.75, 0.1) & \text{Resignative Arbeitsunzufriedenheit} \\ \max(m_H, 0.1) & \text{Pseudoarbeitsunzufriedenheit} \end{cases} \tag{2.41}$$

m_H : Gesamtwert der Hygienefaktoren

m_M : Gesamtwert der Motivatoren

Formel 2.42 zeigt eine mögliche Interpretation der Zustände in Bezug auf Krankheits- und Kündigungsrisiko. Das Simulationsmodell selbst verwendet diese Werte nicht, da es Krankheitsausfälle und Kündigungen nicht direkt realisiert. Zwar existieren entsprechende Schnittstellen, über die solche Aktionen ausgelöst werden können, eine auf dem Simulationsverlauf basierende automatische Auslösung ist jedoch nicht Teil des Simulationsmodells. In einer Planspielimplementierung können die Werte eingesetzt werden, um Ereignisse wie Krankheitsausfälle und Kündigungen von Mitarbeitern umzusetzen. Dazu muss das Simulationsmodell lediglich um einfache Regeln erweitert werden, die diese Werte regelmäßig prüfen und bei Unterschreitung eines bestimmten Grenzwerts geeignete Aktionen auslösen.

$$m_{\text{Krankheit}} = \begin{cases} 0.5 & \text{Stabilisierte Arbeitszufriedenheit,} \\ & \text{Stabilisierende Arbeitszufriedenheit,} \\ & \text{Progressive Arbeitszufriedenheit,} \\ & \text{Konstruktive Arbeitsunzufriedenheit} \\ 1.5 & \text{Diffuse Arbeitsunzufriedenheit,} \\ & \text{Fixierte Arbeitsunzufriedenheit,} \\ & \text{Resignative Arbeitsunzufriedenheit,} \\ & \text{Pseudoarbeitsunzufriedenheit,} \end{cases}$$

$$m_{\text{Kündigung}} = \begin{cases} 0.2 & \text{Stabilisierte Arbeitszufriedenheit,} \\ & \text{Stabilisierende Arbeitszufriedenheit,} \\ & \text{Progressive Arbeitszufriedenheit,} \\ & \text{Konstruktive Arbeitsunzufriedenheit,} \\ & \text{Pseudoarbeitsunzufriedenheit} \\ 1.0 & \text{Diffuse Arbeitsunzufriedenheit,} \\ & \text{Resignative Arbeitsunzufriedenheit,} \\ 2.0 & \text{Fixierte Arbeitsunzufriedenheit} \end{cases} \tag{2.42}$$

Alle drei Werte werden nochmals zusätzlich mittels der Funktion des beschränkten Wachstums über die Zeit geglättet, um große Sprünge und starke Schwankungen zu vermeiden.

2.4.6. Modell zum Aufgabenwechsel

Wechselt ein Mitarbeiter von einer Aufgabe zu einer anderen, fällt ein gewisses Maß an Zusatzaufwand an. Ein häufiger Wechsel zwischen mehreren Aufgaben tritt besonders häufig auf, wenn ein Mitarbeiter mehrere Aufgaben gleichzeitig bearbeiten soll, oder in mehrere Projekte eingebunden ist.

Dieser Zusatzaufwand besteht laut DeMarco [DeMarco, 2002] aus folgenden Aspekten:

- **Routinearbeiten**, wie beispielsweise alte Unterlagen wegräumen, neue Unterlagen herholen oder Programme und geöffnete Dokumente wechseln.

- **Eintauchzeit** in die neue Aufgabe, die für das mentale Umstellen von der alten Aufgabe auf die neue Aufgabe und das Einfinden in den neuen Arbeitsfluss benötigt wird.
- **Emotionales Überwinden** von Frustration. Bei unbeliebten Aufgaben muss zuerst eine mentale Hemmschwelle überwunden werden, bevor mit der Aufgabe begonnen werden kann.
- **Gedankliche Doppelarbeit**. Gedanken die im Kontext dieser Aufgabe schon gemacht, jedoch noch nicht ausreichend in das Arbeitsergebnis eingepflegt wurden, müssen zum Teil nochmal gemacht werden. Das gilt vor allem für unstrukturiertes und kreatives Arbeiten.

DeMarco veranschlagt für einen durchschnittlichen Aufgabenwechsel einen Zusatzaufwand von 20 Minuten bzw. für einen Mitarbeiter der mit 2 oder 3 Aufgaben gleichzeitig betraut ist einen Zusatzaufwand von 15%.

Das Simulationsmodell setzt den von DeMarco beschriebenen zeitlichen Mehraufwand wie in Formel 2.43 gezeigt um.

$$t_\Delta = 10 + 7 \cdot j_* + 6 \cdot (1 - \psi_a)$$

t_Δ : zeitlicher Mehraufwand in Minuten
j_* : charakterspezifische Fähigkeit Aufgaben zu wechseln
ψ_a : Sympathie für die Aktivität der neuen Aufgabe

(2.43)

Dazu wird für das Wechseln der Unterlagen und die gedankliche Doppelarbeit ein Wert von 10 Minuten veranschlagt. Die restlichen 10 Minuten unterteilt das Simulationsmodell in durchschnittlich 7 Minuten für den gedanklichen Wechsel, und 3 Minuten für das emotionale Überwinden bei einer neutralen Aktivitätssympathie von $\psi_a = 0.5$. Während die Zeit für die Routinearbeit und gedankliche Doppelarbeit fix ist, ist die Zeit für den gedanklichen Wechsel von einer entsprechenden Fähigkeit des Mitarbeiters abhängig, die Zeit für das emotionale Überwinden von der Sympathie des Mitarbeiters für die neue Aufgabe.

Dazu enthält der Charakter jedes Projektbeteiligten einen Wert zwischen 0 und ∞, der angibt, wie gut der Projektbeteiligte mental zwischen Aufgaben wechseln kann. Bei einem Wert von 0 benötigt der Mitarbeiter keine zusätzliche Zeit, bei einem Wert von 1 benötigt er die durchschnittlichen 7 Minuten, bei Werten größer 1 benötigt er überdurchschnittlich viel zusätzliche Zeit.

Um die Zeit für die Bewältigung der emotionalen Hemmschwelle zu berechnen, wird die Aktivitätssympathie des Projektbeteiligten verwendet, die auch im Motivationsmodell zum Einsatz kommt.

Ein Aufgabewechsel findet nicht nur statt, wenn ein Mitarbeiter eine Aufgabe beendet und beginnt eine andere Aufgabe zu bearbeiten, sondern auch bei Unterbrechungen, zum Beispiel durch Kollegen, Feierabend, Wochenende oder Urlaub. Die Berechnung des Mehraufwands entspricht dabei im Wesentlichen dem des Aufgabenwechsels. Da der Mitarbeiter seinen Arbeitsplatz nicht verändern muss, entfallen jedoch die dafür vorgesehenen 10 Minuten. Außerdem muss berücksichtigt werden, wie lange die Unterbrechung der Arbeit ist, um sehr kurze Unterbrechungen nicht überzubewerten. Formel 2.44 zeigt die dafür notwendigen Berechnungen. Die Unterbrechungsdauer wirkt sich linear bis zu einer maximalen Unterbrechungsdauer von 30 Minuten für die gedankliche Umstellung, und 15 Minuten für das emotionale Eintauchen auf den Mehraufwand aus.

$$t_\Delta = 7 \cdot j_* \cdot \min(t_{\Delta\mathrm{U}} \cdot 0.033, 1) + 6 \cdot (1 - \psi_a) \cdot \min(t_{\Delta\mathrm{U}} \cdot 0.066, 1)$$

t_Δ : zeitlicher Mehraufwand in Minuten
j_* : charakterspezifische Fähigkeit Aufgaben zu wechseln
ψ_a : Sympathie für die Aktivität der neuen Aufgabe
$t_{\Delta\mathrm{U}}$: Dauer der Unterbrechung in Minuten

(2.44)

Durch das Modell der Aufgabenwechsel kann der Spieler lernen, wie wichtig es ist, dass sich die Mitarbeiter auf eine Aufgabe konzentrieren können, anstatt mehrere Arbeiten gleichzeitig erledigen zu müssen. Auch der negative Einfluss ständiger Unterbrechungen eines Arbeitsprozesses, wird durch das Modell realisiert.

2.5. Tätigkeitsmodelle

Der gesamte Simulationsverlauf wird durch die *Tätigkeiten*, welche die einzelnen Projektbeteiligten durchführen, bestimmt. Es gibt drei wesentliche Typen von Tätigkeiten: *Erholung*, *Arbeitsplatz betreten* bzw. *Arbeitsplatz verlassen*, und *Durchführung einer Aufgabe*, um am Projekt zu arbeiten. Für jeden Typ existiert ein eigenes Tätigkeitsmodell, das vorgibt, wie die entsprechende Tätigkeit abläuft, welche Elementarmodelle daran beteiligt sind, und wie diese kombiniert werden. Diese Modelle werden im Folgenden beschrieben. Zuvor wird die zeitliche Struktur eingeführt, die der Simulation zugrunde liegt

2.5.1. Zeitliche Struktur

Das Simulationsmodell betrachtete ausschließlich den Projektzeitraum. Der Startzeitpunkt des Projekts wird über das Szenario festgelegt. Der Endzeitpunkt des Projekts wird entweder auch über das Szenario definiert, alternativ kann es auch dem Spieler überlassen werden, wann er sein Projekt beenden will. Projektbeginn und Projektende definieren den durch die Simulation abgedeckten Zeitraum. Die Zeit vor dem Projektbeginn wird nicht simuliert, sondern spiegelt sich allenfalls im Ausgangsszenario wider. Die Zeit nach dem Projektende kann durch eine anschließende Simulation abgedeckt werden, indem das Simulationsergebnis als neues Ausgangsszenario verwendet wird. So können beispielsweise Anschlussprojekte realisiert werden.

Der Projektzeitraum lässt sich grob in *Arbeitszeit* und *Freizeit* unterteilen. Der Fokus des Simulationsmodells liegt auf den Vorgängen, die während der Arbeitszeit stattfinden. Da die Leistung der Mitarbeiter jedoch nicht nur von den Vorgängen während der Arbeitszeit abhängen, sondern auch maßgeblich durch die Erholung, die während der Freizeit stattfindet, beeinflusst wird, reicht es nicht, nur die Arbeitszeit zu simulieren, es muss auch die Freizeit betrachtet werden. Die Tätigkeiten, die der Mitarbeiter während der Freizeit durchführt, sind in einer einzigen Tätigkeit *Erholung* abstrahiert, und werden durch das Simulationsmodell nicht detaillierter betrachtet. Während seiner Freizeit kann der Projektbeteiligte daher lediglich diese Erholungstätigkeit durchführen.

Die Arbeitszeit der Projektbeteiligten orientiert sich an einer 5-Tage-Woche mit 40 Wochenstunden Arbeitszeit, Arbeitsbeginn um 8 Uhr morgens, und Arbeitsende um 16 Uhr nachmittags. Diese Standardkonfiguration kann durch das Szenario verändert werden, um Zeitraum und Länge der Arbeitszeit zu verändern. So kann beispielsweise Zeit für eine Mittagspause geschaffen werden, indem das Arbeitszeitende nach hinten verschoben wird. Außerdem können Zeitbereiche als Urlaubszeit festgelegt werden. Alle Einstellungen können individuell für jeden Projektbeteiligten gemacht werden.

Die tägliche Arbeitszeit jedes Mitarbeiters beginnt mit dem Betreten, und endet mit dem Verlassen seines Arbeitsplatzes. Das dafür verantwortliche Tätigkeitsmodell beinhaltet lediglich die Änderung des Status, der angibt, ob sich der Mitarbeiter momentan an seinem Arbeitsplatz befindet. Weitere Modelle sind daran nicht beteiligt. Sowohl das Betreten als auch das Verlassen des Arbeitsplatzes ist zeitlich punktuell. Der Weg zur Arbeit oder von der Arbeit nach Hause gehört nicht zur Arbeitszeit des Mitarbeiters, sondern zu seiner Freizeit.

Das Szenario legt die *vertragliche Arbeitszeit* des Mitarbeiters fest. Diese stimmt jedoch nicht zwangsläufig mit der tatsächlichen Arbeitszeit überein. Die Summe der täglichen Abweichungen zwischen tatsächlicher Arbeitszeit und vertraglicher Arbeitszeit ergibt die Überstunden oder Minusstunden des Mitarbeiters. Jeder Mitarbeiter bestimmt über sein Verhaltensmodell selbst, wann er tatsächlich kommt und geht. Um die Simulation besser steuern zu können, ist pro Tag maximal ein Arbeitsintervall vorgesehen. Pausen, die der Mitarbeiter macht, werden als eigene Tätigkeit während der Arbeitszeit umgesetzt.

Jeder Mitarbeiter muss die gesamte Projektzeit vollständig durch Tätigkeiten abdecken. Ist es einem Mitarbeiter nicht möglich eine Aufgabe zu bearbeiten, wird er dazu gezwungen, die Erholungstätigkeit durchzuführen. Dass die einzelnen Agenten diese Rahmenbedingung einhalten, wird durch die Ausführungsumgebung gewährleistet. So wird sichergestellt, dass fehlerhafte Implementierungen einzelner Agent den Rest der Simulation nicht zerstören.

Befindet sich ein Projektbeteiligter an seinem Arbeitsplatz, stehen im Aufgabentätigkeiten zur Verfügung, um aktiv am Projekt zu arbeiten. Er hat auch die Möglichkeit die Erholungstätigkeit zu nutzen, um Pausen zu machen, oder mit Arbeitsplatz verlassen in die Freizeitphase zu wechseln.

Während seiner Freizeit stehen ihm ausschließlich die Erholungstätigkeit und die Tätigkeit Arbeitsplatz betreten zur Verfügung, mit der er zurück in die Arbeitszeitphase wechseln kann.

2.5.2. Erholung

Führt ein Mitarbeiter in einem Zeitabschnitt keine Aufgabe durch, so wird diese Zeit als Erholungszeit simuliert. Erholungszeiten nehmen typischerweise den größten Teil des Tages ein und realisieren unter anderem Freizeit, Schlaf und Pausen des Mitarbeiters. Das Simulationsmodell abstrahiert die konkreten Aktivitäten als *Erholung*, eine detailliertere Betrachtung findet nicht statt, da es für das Planspiel nicht relevant ist, wie genau sich die Mitarbeiter erholen.

Während durch die Bearbeitung von Aufgaben das Energieniveau des Mitarbeiters abnimmt, steigt es durch Erholung wieder an. Ein ausgewogenes Verhältnis zwischen Arbeit und Erholung ist notwendig, um den Energiewert, und damit die Arbeitskraft der Mitarbeiter, auf einem produktiven Niveau zu halten.

Erholung ist als eine eigene Tätigkeit definiert. Außerhalb der Arbeitszeit ist sie die einzige mögliche Tätigkeit und wird daher implizit angenommen. Während der Arbeitszeit kann jeder Mitarbeiter anstelle einer Aufgabentätigkeit auch die Erholungstätigkeit durchführen, und somit eine Pause machen. Um Erholung zu simulieren, wird das Elementarmodell der Erholung, wie in Formel 2.10 beschrieben verwendet, das als einzigen Parameter die Erholungsdauer nutzt, um den Energiegewinn durch die Erholung für diesen Zeitraum zu berechnen.

Befindet sich der betrachtete Mitarbeiter während der Erholungstätigkeit am Arbeitsplatz, hat das Auswirkungen auf seine Motivation. Zu häufige, oder zu lange Leerphasen während der Arbeitszeit führen zu Langeweile und Unterforderung, die wiederum, wie im Motivationsmodell beschrieben, Auswirkungen auf die Bewertung der Arbeitsinhalte und die eigene Leistung hat. Entsprechend muss neben der Simulation der Erholung selbst, hier auch das Motivationsmodell, wie in Formel 2.37 und Formel 2.39 beschrieben, aktualisiert werden.

2.5.3. Aufgabentätigkeiten

Um am Produkt einen Fortschritt zu erzielen, müssen die Projektmitglieder Aktivitäten auf dessen Artefakten durchführen. Welche Aktivitäten auf welchen Artefakten, und unter welchen Bedingungen durchgeführt werden können, wird durch das Szenario über das Produkt und durch die Definition der Aktivitäten und Rollen, sowie deren Abhängigkeiten festgelegt (vgl. Abschnitt 2.3).

Dabei kann eine Aktivität von mehreren Mitarbeitern gemeinsam durchgeführt werden. Das bedeutet nicht, dass die an der Tätigkeit beteiligten Mitarbeiter parallel arbeiten, sondern dass diese sich zusammensetzen, um gemeinsam ein Problem zu lösen oder ein Ergebnis zu erzielen. Insbesondere Kommunikationsaufgaben können nicht von einem Mitarbeiter alleine durchgeführt werden. Arbeiten mehrere Mitarbeiter parallel an einem Artefakt, so wird dies durch mehrere zeitgleiche, jedoch unabhängige einzelne Aufgaben realisiert. Wird eine Aufgabe von mehreren Mitarbeitern gemeinsam ausgeführt, hat das diverse Auswirkungen auf den Aufgabenverlauf und das Arbeitsergebnis.

Die Ausführung einer Projektaufgabe läuft unter Verwendung der zuvor beschriebenen Elementarmodell nach einem festen Schema ab. Die dazu notwendigen Schritte folgen sequentiell aufeinander:

1. Überprüfung der Voraussetzungen
2. falls nötig, Aktivitätswechsel durchführen
3. Arbeitszeit verbuchen
4. fehlendes Wissen kompensieren
5. Produktfortschritt berechnen
6. spezielle Einflüsse umsetzen
7. Wissenszuwachs berechnen
8. Fähigkeitszuwachs berechnen
9. Motivationsmodell aktualisieren

Bei Kommunikationsaktivitäten entfällt die Kompensation von fehlendem Wissen und die Berechnung des Produktfortschritts, da das Artefakt nicht bearbeitet, sondern lediglich Wissen darüber ausgetauscht wird. Da die Kommunikationsfähigkeiten Teil des Charakters sind und sich nicht durch die Simulation verändern, entfällt auch die Berechnung des Fähigkeitszuwachses.

Die einzelnen Berechnungsschritte verwenden als Eingangsgrößen entweder den unveränderten Zustand des vorangegangenen Zeitschritts oder entsprechende Zwischenergebnisse, die durch vorherigen Berechnungsschritte temporär erzeugt wurden.

Initiale Schritte

Bevor eine Aktivität auf ein Artefakt angewendet werden kann, muss überprüft werden, ob die Voraussetzungen dafür erfüllt sind. Dabei werden zum einen die generellen Voraussetzungen für die Aufgabe, zum anderen die persönlichen Voraussetzungen der einzelnen Mitarbeiter geprüft.

An der Aufgabe nehmen von den dafür vorgesehenen Mitarbeitern nur diejenigen teil, die folgende Voraussetzungen erfüllen:

- Der Mitarbeiter ist an seinem Arbeitsplatz anwesend. Mitarbeiter, die nicht zur Arbeit erschienen sind, können keine Aufgaben bearbeiten.
- Der Mitarbeiter nimmt mindestens eine, der von der Aktivität geforderten Rollen ein.
- Das Zeitkonto des Mitarbeiters ist für den von der Aufgabe vorgegebenen Zeitschritt noch nicht erschöpft. Das heißt, der Mitarbeiter hat durch vorangegangene Aufgaben das dafür vorgesehene Zeitintervall nicht so stark überzogen, dass er für die aktuelle Aufgabe keine Zeit mehr hat (vgl. Abschnitt 2.1.3).

Folgenden generelle Voraussetzungen müssen erfüllt sein, damit die Aufgabe bearbeitet werden kann:

- Die Aktivität muss zum Artefakt passen. Das heißt, die Menge der Zielartefakttypen der Aktivität muss den Artefakttyp des zu bearbeitenden Artefakts enthalten. Damit wird sichergestellt, dass Aktivitäten nicht auf falsche Artefakte angewendet werden können, und so beispielsweise Codefragmente durch eine Anforderungsanalyse entstehen.
- Alle für das zu bearbeitende Artefakt angegebenen relevanten Vorbedingungen müssen erfüllt sein. Eine Vorbedingung ist relevant, wenn sie sich auf ein Artefakt bezieht, dessen Typ in der Menge der Kontextartefakttypen der betrachteten Aktivität vorhanden ist. Ob eine Vorbedingung erfüllt ist, hängt von deren Definition und dem Zustand des Artefakts ab, auf das sie sich bezieht. Typischerweise wird dabei ein gewisser Fortschritt des Artefakts oder ein bestimmtes Qualitätsniveau gefordert.
- Es existiert mindestens ein Mitarbeiter, der die notwendigen persönlichen Voraussetzungen erfüllt.

Mitarbeiter, deren Zeitkonto nicht ausreicht oder die nicht am Arbeitsplatz sind, werden bei den folgenden Schritten ignoriert, da sie an der Aufgabe physisch nicht teilnehmen. Die restlichen Mitarbeiter werden im Folgenden als *aktive* und *passive Mitarbeiter* unterschieden. Aktive Mitarbeiter erfüllen alle Voraussetzungen um an der Aktivität teilzunehmen, passive Mitarbeitern fehlt mindestens eine der Voraussetzungen.

Ist mindestens eine der generellen Voraussetzungen nicht erfüllt, wird die eigentliche Aktivität nicht durchgeführt. Dadurch werden alle Mitarbeiter zu passiven Mitarbeitern.

Während nur die aktiven Mitarbeiter einen Einfluss auf die Veränderung des Produkts haben, und sich durch die Durchführung der Aktivität weiterentwickeln, wird die Arbeitszeit auch für die passiven Mitarbeiter verbucht, und für sie auch ein Aufgabenwechsel durchgeführt, da sie formal an der Aufgabe teilnehmen. Ein Projektleiter muss dementsprechend darauf achten, dass er seinen Mitarbeitern nur Aufgaben zuordnet, für die sie die entsprechenden Voraussetzungen mitbringen, um nicht unnötig Kapazitäten zu verschwenden. Je nach Verhaltensmodell sind Mitarbeiter gegebenenfalls auch selbst in der Lage solche Probleme zu erkennen und Alternative zu finden. Ein entsprechendes Verhaltensmodell wird weiter unten beschrieben.

Sowohl für aktive, als auch für passive Mitarbeiter wird, falls notwendig, ein Aufgabenwechsel gemäß dem Modell in Abschnitt 2.4.6 durchgeführt. Dazu wird die zuletzt vom Mitarbeiter bearbeitete

Aufgabe mit der aktuellen Aufgabe verglichen, und mit Hilfe des Modells zum Aufgabenwechsel ermittelt, wie viel zusätzliche Zeit für den Aufgabenwechsel oder die Unterbrechung der Aufgabe veranschlagt werden muss.

Wird die Aufgabe von mehr als einem Mitarbeiter bearbeitet, warten alle Beteiligten, bis der letzte Mitarbeiter den Aufgabenwechsel abgeschlossen hat, um gemeinsam mit der eigentlichen Aufgabe beginnen zu können. Darunter fallen auch passive Mitarbeiter, die, obwohl sie keinerlei Produktivität liefern, die Arbeit der anderen verzögern können. Außerdem wird hier die eventuelle Überziehung der Zeitkonten aus vorangegangenen Aufgaben aufgelöst, indem diese Zeiten zu den Aufgabenwechselzeiten der jeweiligen Mitarbeiter hinzugerechnet werden.

Sowohl die für den Aufgabenwechsel benötigte Zeit, als auch die für die eigentliche Bearbeitung noch übrige Zeit, wird sowohl für die aktiven, als auch für die passiven Mitarbeiter verbucht. Somit wird der gesamte für die Aufgabe veranschlagte Zeitschritt verbucht, und damit für andere Aufgaben blockiert.

Abbildung 2.38 zeigt einen möglichen zeitlichen Ablauf einer Aufgabe, für die eine Bearbeitungszeit von 60 Minuten vorgesehen ist. Mitarbeiter 2 ist zu Beginn des Zeitintervalls noch durch eine vorangegangene Aufgabe blockiert (rot) und kann mit dem Aufgabenwechsel (blau) deswegen erst später beginnen. Mitarbeiter 1 hat die Aufgabe schon in einem früheren Zeitschritt begonnen und muss daher nur auf seine Kollegen warten, während Mitarbeiter 3 den Aufgabenwechsel sofort vollziehen kann.

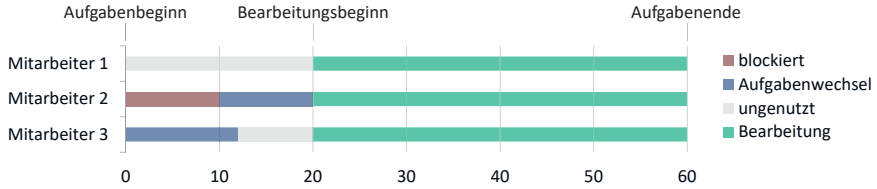


Abbildung 2.38.: Beispiel zu einem zeitlichen Ablauf einer Aufgabe

Produktfortschritt berechnen

Die konkreten Änderungen am Artefakt, welche mittels der Aufgabe erzielt werden, berechnen sich mit Hilfe des Elementarmodells zur Artefaktbearbeitung. Dabei werden zuerst die notwendigen Eingangswerte ermittelt, anschließend wird die Arbeitsleistung berechnet, die am Ende, entsprechend des jeweiligen Aktivitätstyps auf das Artefakt angewendet wird. Da Kommunikationsaktivitäten keine Änderungen am Produkt vornehmen, entfällt für sie dieser Schritt, der durch einen anderen ersetzt wird.

Die Berechnung der Arbeitsleistung fordert als Eingangsgrößen Zustand und Eigenschaften des zu bearbeitenden Artefakts, seiner Kontextartefakte, und des Bearbeiters. Das Modell zur Artefaktbearbeitung berücksichtigt nicht, dass eine Aufgabe auch von mehreren Mitarbeitern gemeinsam bearbeitet werden kann. Deshalb müssen bei mehr als einem Mitarbeiter, die relevanten Eigenschaften der einzelnen Mitarbeit zu jeweils einer gemeinsamen Größe zusammengefasst werden, die das Modell verarbeiten kann.

An dieser Stelle ist es wichtig, das gemeinsame Bearbeiten nicht mit dem parallelen Bearbeiten zu verwechseln. Ein gemeinsames Bearbeiten findet beispielsweise bei der *Paarprogrammierung* statt. Dabei sitzen zwei Mitarbeiter gemeinsam an einem Computer, wobei einer der beiden Code schreibt, den der andere nach dem Zweiaugenprinzip direkt überprüft. Gleichzeitig hat der Mitarbeiter, der nicht mit dem Eingeben des Codes beschäftigt ist, die Möglichkeit, über das zu lösende Problem nachzudenken und den Überblick zu behalten. Um solche Ansätze zu realisieren, muss das Simulationsmodell eine Aufgabenbearbeitung durch Gruppen von Projektbeteiligten unterstützen. Dabei wird die Bearbeitung nicht zwangsläufig beschleunigt, sondern kann je nach Aktivität und Beschaffenheit der Gruppe auch andere Vorteile haben.

Für die Arbeitsleistungsberechnung wird die Gruppe als ein einzelner Mitarbeiter aufgefasst. Dazu wird das Wissen über das Produkt, die physische Basisleistung, die Motivation und die Fähigkeiten aller aktiven Mitarbeiter, wie in Formel 2.45 gezeigt, zusammengefasst.

$$
\begin{aligned}
&\hat{K_0}, \ldots, \hat{K_j} = \bigcup_{i=1}^{n} K_{0_i}, \ldots, \bigcup_{i=1}^{n} K_{j_i} \\
&\hat{l} = \max_{i=1\ldots n} (l_i) \\
&\overline{m} = \frac{\sum_{i=1}^{n} \left(m_{\text{Leistung}_i} \cdot (\psi_{ai} + 0.5) \right)}{n} \\
&\overline{p} = \frac{\sum_{i=1}^{n} P_{\text{Physis}}(t_{\text{Start}}, t_{\text{Ende}}, \mathrm{e}_i, \rho_i, \hat{\rho}_i, P_{T_i})}{n} \\
&\overline{p'} = \overline{p} \cdot F_a(n)
\end{aligned}
\tag{2.45}
$$

n : Anzahl der Mitarbeiter

K_{i_j} : Wissen des $j$ten Mitarbeiters über das $i$te Kontextartefakt. $i = 0$ entspricht dem zu bearbeitende Artefakt.

l_i : Fähigkeitsniveau des $i$ten Mitarbeiters für die durchzuführende Aktivität bzw. relevante Disziplin

m_{Leistung_i} : Leistungsauswirkung der Motivation des $i$ten Mitarbeiters

ψ_{ai} : Sympathie des $i$ten Mitarbeiters für die durchzuführende Aktivität

P_{Physis} : Elementarmodell zur Ermittlung des physischen Leistungsniveaus des $i$ten Mitarbeiters.

F_a : aktivitätsspezifische Funktion zur Berechnung des Leistungseinflusses durch die Gruppengröße

Da die Mitarbeiter die Aufgabe gemeinsam bearbeiten, können sie auf das Wissen der anderen Gruppenmitglieder zugreifen. Daher entspricht das Gesamtwissen, das der Gruppe bei der Aufgabenbearbeitung zur Verfügung steht, der Vereinigung des Artefaktwissens aller Gruppenmitglieder. Das gilt sowohl für das zu bearbeitende Artefakt, als auch für alle relevanten Kontextartefakte. Dadurch ergibt sich für die Gruppe ein erheblicher Vorteil, wenn das Wissen, das für die Aufgabe benötigt wird, stark unter den Mitarbeitern verteilt ist. Ein einzelner Mitarbeiter müsste dieses Wissen zuerst erlangen, bevor er die Aufgabe sinnvoll bearbeiten kann.

Bei der für die Aufgabe relevanten Fähigkeit, wird der beste Wert der Gruppe gewählt. Dieser Ansatz basiert auf der Annahme, dass eine Gruppe niemals schlechter wird, wenn Mitarbeiter mit niedrigeren Fähigkeiten hinzukommen, solange die Gruppe immer gemeinsam arbeitet, anstatt die Aufgaben aufzuteilen. Die besseren Mitarbeiter verhindern die Fehler, welche die schlechteren Mitarbeiter machen würden, und finden die Lösungen, die den schlechteren Mitarbeitern nicht einfallen. Im Extremfall kann ein schlechterer Mitarbeiter ausschließlich die Rolle des Beobachters einnehmen, um der Gruppe nicht zu schaden. So liefert er zwar keinen produktiven Anteil an der Aufgabe, kann jedoch durch die Teilnahme an der Aktivität seine Fähigkeiten verbessern und Wissen erwerben.

Der Motivationswert, der in das Arbeitsleistungsmodell eingeht, berechnet sich als Mittelwert über alle Teilnehmer. Im Gegensatz zu Wissen und Fähigkeiten, gibt es hier keine klaren Annahmen, wie sich die Motivation der einzelnen Teilnehmer gegenseitig beeinflusst, und ob sich die Gruppe eher an den motivierten oder den unmotivierten Mitarbeitern orientiert. Das Maximum zu nehmen kann an dieser Stelle problematisch sein. Wenn beispielsweise dem motivierten Mitarbeiter die Fähigkeiten, und dem fähigen Mitarbeiter die Motivation fehlt, wird die Aufgabe im Original vermutlich nicht mit dem Maximum an Fähigkeit und Motivation bearbeitet. Der Mittelwert ist ein neutraler Ansatz mit dem Problem umzugehen, bis dafür ein besseres Modell gefunden wird. In die Berechnung geht der Leistungseinfluss aus dem Motivationsmodell der einzelnen Mitarbeiter ein. Da sich dieser Wert allgemein auf das gesamte Projekt bezieht, wird er zusätzlich anhand der Sympathie des jeweiligen Mitarbeiters, welche die intrinsische Motivation für die durchzuführende Aktivität angibt, um bis auf die Hälfte reduziert bzw. bis auf das Doppelte erhöht. So wird nicht nur die allgemeine, sondern auch die aufgabenspezifische Motivation der Mitarbeiter berücksichtigt.

Auch die Basisleistung der Gruppe wird anhand des Mittelwerts über die Basisleistung der einzelnen Mitarbeiter berechnet. Die Argumentation dafür ist die gleiche wie bei der Motivation. Dazu wird

für jeden einzelnen Mitarbeiter anhand des Modells für die physische Leistung ein individueller Wert berechnet. Diese Werte werden anschließend gemittelt. Das veranschlagte Zeitintervall entspricht dem Zeitraum, der nach Aufgabenwechsel und Wartezeiten im aktuellen Zeitschritt für die Aufgabenbearbeitung noch übrig ist und für die tatsächliche Bearbeitung der Aufgabe zur Verfügung steht.

Um den Einfluss der Gruppengröße auf die Gruppenleistung zu berücksichtigen, wird dieser Wert anschließend mit einem Faktor korrigiert, der mittels einer aktivitätsspezifischen Funktion auf Basis der Gruppengröße berechnet wird. Häufig ist diese Funktion konstant 1, das bedeutet, die Gruppengröße hat keinen direkten Einfluss auf die Leistung. Die positiven Effekte kommen dann ausschließlich durch die oben beschriebenen Zusammenhänge zustande. Für einige Aktivitäten, wie beispielsweise der Inspektion, ist jedoch bekannt, wie sich die Gruppengröße auf die Produktivität auswirkt, und wo ihr Optimum und ihre Sättigung in Bezug auf die Effizienz liegen. In diesen Fällen kann die Funktion für die betroffene Aktivität individuell gewählt werden.

Im nächsten Schritt wird das für die Bearbeitung notwendige, jedoch in der Gruppe der Bearbeiter noch fehlende Wissen behandelt. Wurde dieses Wissen nicht explizit vor Beginn der Aufgabe durch entsprechende Tätigkeiten erworben, muss ein Teil der für die Aufgabe vorgesehenen Zeit investiert werden, um das zu bearbeitenden Artefakt und die relevanten Kontextartefakte zu studieren.

Wie viel Zeit dafür investiert wird, hängt davon ab, wie stark sich das fehlende Wissen auf die eigentliche Bearbeitung auswirkt. Das wiederum hängt von der konkreten Aktivität ab, die eine Funktion liefert, welche den Einfluss der Wissensüberdeckung auf die Arbeitsleistung bestimmt (vgl. 2.4.4). Die in die Kompensation des fehlenden Wissens investierte Zeit, ist für alle Bearbeiter der Gruppe identisch, da von einer gemeinsamen Wissensbasis ausgegangen wird. Sie wird mittels der in Formel 2.46 angegebene Regel für jedes relevante Artefakt einzeln berechnet. Dabei wird berücksichtigt, dass zwangsläufig immer ein gewisser Zeitanteil in den Wissenserwerb fließt. Das geschieht auch, wenn der Bearbeiter sich nicht bewusst Informationen über das Artefakt beschafft. Das Simulationsmodell legt diesen Anteil auf 1% der für die Aufgabe zur Verfügung stehenden Zeit fest, da sich dieser Wert als der kleinstmögliche Wert herausgestellt hat, bei dem der Effekt noch sichtbar ist.

$$
\begin{aligned}
&t_{\Delta\text{Kompensation}} = t_\Delta \cdot \max(1 - F(k_\Delta), 0.01) \\
&t_\Delta : \text{Dauer für die Bearbeitung der Aufgabe} \\
&F : \text{aktivitätsspezifische Funktion zum Einfluss der Wissensabdeckung} \\
&k_\Delta : \text{relative Wissensabdeckung der Gruppe für das betrachtete Artefakt}
\end{aligned}
\tag{2.46}
$$

Anschließend nutzt jeder Mitarbeiter diese Zeit für das Studium des jeweiligen Artefakts. Im Original passiert diese Kompensation während der eigentlichen Bearbeitung. Da es für das Endergebnis keinen Unterschied macht, ob das Artefaktstudium vorangestellt oder parallel zur eigentlichen Aufgabenbearbeitung durchgeführt wird, führt das Simulationsmodell die Berechnungen der Einfachheit halber vor der eigentlichen Artefaktbearbeitung durch. Dazu wird das Elementarmodell für das Erlernen von Artefaktwissen (vgl. 2.4.2, Formel 2.16) mit den jeweiligen Eigenschaften des Mitarbeiters und dem zuvor ermittelten Zeitintervall angewendet.

Die verbleibende Zeit kann nun in die eigentliche Aufgabe investiert werden. Anhand des Modells zur Ermittlung der Arbeitsleistung, wird auf Basis der zuvor ermittelten Parameter zunächst, wie in Abschnitt 2.4.4 beschrieben, die Arbeitsleistung ermittelt, und dies anschließend entsprechend des Artefakttyps auf das Artefakt angewendet, um den Artefaktfortschritt zu realisieren.

Nachdem die Änderungen am zu bearbeitenden Artefakt vollzogen worden sind, müssen, falls entsprechende Beziehungen vorhanden sind, besondere Einflüsse umgesetzt werden. Dazu werden die in den jeweiligen Einflussbeziehungen hinterlegten Funktionalitäten ausgeführt, welche die Änderungen am eben bearbeiteten Artefakt verwenden, um abhängige Artefakte anzupassen.

Sekundäreffekte realisieren

Die primäre Absicht bei der Bearbeitung einer Aufgabe ist es in der Regel, die Artefakte des Produkts weiterzuentwickeln oder bestehende Teile davon zu verbessern. Die Ausführung einer Aufgabe hat im Allgemeinen jedoch noch weitere Effekte, die nicht das Artefakt, sondern die bearbeitenden Mitarbeiter betreffen. Diese Sekundäreffekte werden im letzten Schritt der Aufgabenausführung berechnet und angewendet. Dazu gehören die durch die Aufgabe entstandene Erschöpfung, das erworbene Wissen und die entstandenen Verbesserungen der Fähigkeiten, und die Auswirkungen auf die Motivation der Mitarbeiter.

Sowohl für die aktiven, als auch für die passiven Mitarbeiter der Aufgaben, wird das gesamte Zeitintervall als investierte Arbeitszeit verbucht. Dabei wird der durch die Bearbeitung der Aufgabe verursachte Energieverlust berechnet, und vom Energieniveau der einzelnen Bearbeiter abgezogen. Dazu wird das in Abschnitt 2.4.1 beschriebene Elementarmodell verwendet, das die notwendigen Charaktereigenschaften und die Zeitdauer des Arbeitsvorgangs übergeben bekommt, um damit den Energieverlust zu berechnen.

Für alle aktiven Mitarbeiter wird ermittelt, welches Wissen sie durch die Bearbeitung der Aufgabe erworben, und wie sich ihre Fähigkeiten verbessert haben. Da die passiven Mitarbeiter nicht die Voraussetzungen dafür mitbringen, effektiv an der Aufgabenbearbeitung teilzunehmen, findet bei ihnen keine derartige Verbesserung statt. Um den Wissenszuwachs zu berechnen, wird die im Elementarmodell zum Erwerb von Artefaktwissen für diesen Fall angegebene Regel (vgl. Formel 2.15) für jeden aktiven Mitarbeiter einzeln angewendet. Als Eingangswert benötigte das Modell den Artefaktfortschritt vor der Aufgabenbearbeitung, der dazu vor Beginn der Aufgabe gesichert werden muss, und den Zustand nach der Bearbeitung.

Während der Wissenserwerb unabhängig für alle aktiven Mitarbeiter berechnet werden kann, muss beim Fähigkeitserwerb der Gruppenaspekt berücksichtigt werden. Ein einzelner Mitarbeiter kann von stärkeren Gruppenmitgliedern profitieren, und so einen höheren Lerneffekt erzielen.

Zur Ermittlung des Fähigkeitsgewinns wird auf die Partitionierung der Gesamtaufgabe in ihre Teilaufgaben, wie im Modell zur Artefaktbearbeitung beschrieben (vgl. Abschnitt 2.4.4, Formel 2.20), zurückgegriffen. Wie auch die Berechnung des Fähigkeitseinflusses auf das Arbeitsergebnis, wird auch der Fähigkeitszuwachs anhand der Teilaufgaben einzeln für jede relevante Disziplin, und die Aktivität der Aufgabe berechnet. Die einzelnen Berechnungen sind dabei jeweils bis auf die Aufgabendauer und die Fähigkeit, auf die sie sich beziehen, identisch. Der Fähigkeitszuwachs wird für jeden aktiven Mitarbeiter und jede relevante Fähigkeit einzeln berechnet.

$$
\begin{aligned}
l_{*+}{}' &= \min(\max(\hat{l} - l_n, 0) + l_{*+}, 1) \\
l_{\text{einzeln}} &= L_{\Delta+}(\theta, l_n, l_{*+}{}', t_\Delta) \\
l_{\text{Gruppe}} &= L_{\Delta+}(\theta, \hat{l}, l_{*+}{}', t_\Delta) \\
\underline{l}_{\text{einzeln}} &= \underline{L}_{\Delta+}(\underline{l}_n, l_{\text{einzeln}} + l_n, l_{*+}, t_\Delta) \\
l_{n+1} &= l_n + \min(l_{\text{einzeln}}, l_{\text{Gruppe}}) \cdot \min(1, c_\Delta \cdot 4) \\
\underline{l}_\Delta &= \underline{l}_{\text{einzeln}} \cdot \max(1, c_\Delta \cdot 4)
\end{aligned}
\tag{2.47}
$$

l_n : Fähigkeitsniveau des betrachteten Bearbeiters vor der Aufgabe

$\hat{l}$: maximales Fähigkeitsniveau in der Gruppe

t_Δ : Zeit, die in die betrachtete Teilaktivität investiert wurde

$\underline{l}_n$: Fähigkeitsniveau im Langzeitgedächtnis zu Beginn der Aufgabe

l_{*+} : persönlicher Lernfaktor des betrachteten Bearbeiters

c_Δ : tatsächlicher am Artefakt durch die Aufgabe erzielter Fortschritt, bzw. Gesamtumfang der durch die Aufgabe aufgedeckten Defekte

Dazu wird mittels der Modelle zum Erwerb von Fähigkeiten (vgl. Abschnitt 2.4.2, Formel 2.12, Formel 2.14) der Zuwachs der unmittelbaren Fähigkeiten, sowie der Zuwachs im Langzeitgedächtnis

ermittelt. Formel 2.47 zeigt, wie die Eingangswerte der Berechnung ermittelt, und wie die Ergebnisse der Berechnung zusammengesetzt werden, um den tatsächlichen Fähigkeitszuwachs der einzelnen Gruppenmitglieder zu ermitteln.

Dabei wird der Zuwachs für das unmittelbare Fähigkeitsniveau, unabhängig voneinander, einmal auf Basis des persönlichen Niveaus des betrachteten Mitarbeiters, und einmal auf Basis des Niveaus der Gruppe berechnet. Anschließend wird der größere Wert als tatsächlicher Zuwachs ausgewählt. So profitieren alle Gruppenmitglieder, deren Fähigkeiten unterhalb des Gruppendurchschnitts liegen von der Gruppe, während die besseren Mitglieder nicht durch die Gruppe beim Lernen behindert werden.

Das Simulationsmodell geht von der Annahme aus, dass das Lernen in der Gruppe Vorteile für den Lerneffekt bringt. Um das zu realisieren, wird der Lernfaktor, der individuell angibt, wie schnell ein Projektbeteiligter lernt, verändert. Dazu wird die Differenz der eigenen Fähigkeit zur Gruppenfähigkeit ermittelt und auf den Faktor addiert, falls sich dieser dadurch verbessern lässt.

Der Fähigkeitszuwachs im Langzeitgedächtnis wird nur auf Basis der Eigenschaften des betrachteten Mitarbeiters bestimmt. Da der Übergang von Fähigkeiten in das Langzeitgedächtnis hauptsächlich durch die eigene Anwendung der Fähigkeiten geschieht, der zusätzliche Lerneffekt durch eine bessere Gruppe jedoch auch durch reine Beobachtung derer Aktivität zustande kommen kann, wird hier nur der persönliche Lerneffekt betrachtet. Um den Lerneffekt, der durch die Gruppe zustande gekommen ist, in das Langzeitgedächtnis zu bringen, muss dieser in weiteren Aufgaben selbst angewendet werden.

Um zu verhindern, dass Aufgaben, die keinen Fortschritt erzielen, zu entsprechenden Lerneffekten führen, ist der Zuwachs, sowohl des unmittelbaren Niveaus, als auch der im Langzeitgedächtnis, durch den Fortschritt, den die Aufgabe am Artefakt erzielt hat beschränkt. Bei Fortschrittsaktivitäten wird dazu der absolute Fortschritt in Arbeitseinheiten, bei Qualitätssicherungsaktivitäten, die Reduktion des Defektumfangs in Arbeitseinheiten verwendet.

Im letzten Schritt werden die Auswirkungen der Aufgabenbearbeitung auf die Motivation der Beteiligten ermittelt, und damit das Motivationsmodell aktualisiert. Die Berechnung findet für jeden Mitarbeiter der Gruppe individuell und unabhängig von den anderen statt.

Die Bewertung der *Arbeitsinhalte* beruht hauptsächlich auf den Aufgaben, die der Projektbeteiligte durchführen muss. Um die Auswirkung der betrachteten Aufgabe auf diese Bewertung zu ermitteln, wird die im Motivationsmodell angegeben Regel (Formel 2.38) angewendet, wobei die Dauer des Zeitschritts der Aufgabendauer und die Sympathie/Antipathie der der Aufgabe entspricht. Die Bewertung der Arbeitsinhalte steigt, wenn der Mitarbeiter eine Aufgabe bearbeitet hat, die ihm Spaß macht, und sinkt bei Aufgaben, die er nicht gerne macht.

Die Bewertung der *zwischenmenschlichen Beziehung* wird durch die an der Aufgabe beteiligten Kollegen beeinflusst. Wird ein Mitarbeiter dazu gezwungen, mit unbeliebten Kollegen zusammen zu arbeiten, hat das negative Auswirkungen auf die Bewertung, da er sich in dieser Situation nicht wohl fühlt. Darf er hingegen mit Kollegen zusammenarbeiten, mit denen er sich gut versteht, wirkt sich das positiv auf die Bewertung seines menschlichen Umfelds am Arbeitsplatz aus. Um diesen Effekt zu simulieren, wird die Bewertung entsprechend der im Motivationsmodell vorgesehenen Regel (Formel 2.32) für jeden an der Aufgabe beteiligten Kollegen aktualisiert.

Neben den Bedingungen, unter denen die Aufgabe ausgeführt wird, sind auch deren Ergebnisse für die Änderungen der Motivation wichtig. Der Motivator zum *persönlichen Wachstum* wird im Simulationsmodell ausschließlich durch die Lernfortschritte der aktiven Mitarbeiter beeinflusst. Diese finden wiederum nur im Kontext der Aufgabenbearbeitung statt. Um die Auswirkungen zu ermitteln, wird der zuvor mittels Formel 2.47 berechnete Fähigkeitszuwachs $l_{\Delta+}$ in der im Motivationsmodell angegebenen Regel (Formel 2.35) zusammen mit der Aufgabendauer verwendet. Da sich das durch die Aufgabe erworbene Artefaktwissen nur im Kontext des konkreten zu entwickelnden Produkts verwenden lässt, hat es keine Auswirkungen auf die Bewertung des allgemeinen persönlichen Wachstums, und wird daher bei der Aktualisierung des Motivationsmodells ignoriert.

Der letzte Faktor, der aktualisiert werden muss, ist die Bewertung der *Leistung und Erfolgserlebnisse*. Dazu wird für die aktiven Mitarbeiter die Arbeitsleistung verwendet, die bei der Aufgabenbearbeitung erzielt wurde. Diese liegt als Ergebnis der Arbeitsleistungsberechnung vorangegangener Schritte vor. Dabei ist zu beachten, dass der tatsächliche Artefaktfortschritt davon abweichen kann, wenn mehr Zeit in ein Artefakt investiert wird, als für dessen Fertigstellung notwendig ist. In diesem Fall muss die Fortschrittskomponente der Arbeitsleistung entsprechend reduziert werden, um zu verhindern,

dass Arbeit, die zwar potentiell geleistet werden hätte können, jedoch nicht tatsächlich geleistet wurde, als Erfolg gewertet wird. Für passive Mitarbeiter wird eine Arbeitsleistung von 0 angenommen. Die Berechnung der Änderung am Motivator wird entsprechend der Regel im Motivationsmodell (Formel 2.36) vorgenommen.

Anpassungen für Kommunikationsaufgaben

Kommunikationsaufgaben verändern den Zustand der Produktartefakte nicht. Daher weicht ihr Ablauf von dem der anderen Aufgaben ab. Die initialen Schritte entsprechend denen der anderen Aufgaben, mit der Ausnahme, dass die Aufgabe nicht begonnen wird, wenn nicht mindestens zwei Teilnehmer anwesend sind.

Anstelle des Produktfortschritts wird bei Kommunikationsaufgaben das übertragene Wissen mittels dem Modell zur Konferenzkommunikation (vgl. 2.4.3) ermittelt. Die Kommunikation zwischen zwei einzelnen Teilnehmern ist ein Spezialfall dieses Modells und muss nicht gesondert behandelt werden. Das Modell liefert für jeden aktiven Mitarbeiter das Wissen, das er durch die Kommunikationsaufgabe erworben hat. Für passive Mitarbeiter wird ein Wissen von 0 angenommen. Auf Basis dieses Werts und der in die Aufgabe investierte Zeit kann die Kommunikationsgeschwindigkeit ermittelt werden, die für die Sekundäreffekte benötigt wird.

Die Sekundäreffekte werden wie bei den anderen Aufgaben angewendet. Da bei Kommunikationsaufgaben kein Wert für die Qualität der Kommunikationsleistung ermittelt wird, und das Simulationsmodell keine Schwierigkeit von Kommunikationsaufgaben kennt, wird von einer durchschnittlichen Qualität und Aufgabenschwierigkeit ausgegangen. Das Motivationsmodell verhält sich gegenüber diesen durchschnittlichen Werten neutral, Qualität und Schwierigkeit haben so keinen Einfluss auf das Ergebnis. Die Arbeitsgeschwindigkeit wird mit 2% der durch die Kommunikation übertragenen Arbeitseinheiten angenommen, um sie mit der der anderen Aufgaben vergleichbar zu machen.

Eine besondere Form der Kommunikation ist die *spontane Kommunikation*. Während die Kommunikation im Rahmen der Aufgaben geplant ist, findet spontane Kommunikation kurzfristig und ungeplant statt. Sie steht den Mitarbeitern zur Verfügung, um Kollegen nach bestimmten Wissen zu fragen, ohne dass diese Aktivität vom Projektleiter explizit vorgegeben wird. Ihr ist somit auch keine explizite Aktivität aus dem Szenario zugeordnet. Für spontane Kommunikationsaktivitäten finden keine Überprüfungen bezüglich der notwendigen Rollen statt, das bedeutet, jeder Mitarbeiter hat die Möglichkeit, jeden für ihn erreichbaren anderen Mitarbeiter zu befragen.

Wann und wen ein Mitarbeiter fragt, wird über sein Verhaltensmodell geregelt. Im Gegensatz zur Kommunikation im Rahmen einer Aufgabe, findet bei spontaner Kommunikation kein vollständiger Aufgabenwechsel statt, die aktuelle Aufgabe wird jedoch für sowohl beim Fragenden, als auf beim Gefragten für die benötigte Zeit unterbrochen, was wiederum Auswirkungen auf deren Leistung hat. Wird ein Mitarbeiter häufig von anderen gestört, beeinträchtigt das seine Arbeitsleistung mitunter erheblich.

2.6. Verhaltensmodelle

Vielen Ansätzen für Planspiele im Projektmanagementbereich, betrachten den Projektbeteiligten als reine Ressource. Er stellt eine Arbeitskraft dar, die mit einer gewissen Leistung Arbeiten erledigen kann. Dabei wird der Mitarbeiter häufig nicht als Individuum gesehen. Zwar werden grundlegende Werte wie Energie und Motivation mitberücksichtigt, diese fließen in der Regel jedoch nur als weitere Faktoren in die Berechnung der Leistungsfähigkeit des Mitarbeiters ein.

Diese Arbeit liefert einen neuen Ansatz. Sie beschreibt die Projektbeteiligten als eigenständige individuelle Elemente, die selbstständig innerhalb der ihnen gegebenen Möglichkeiten agieren. Das betrifft zum einen die Arbeit an sich, das heißt die Auswahl der konkreten Aufgaben die sie ausführen, als auch die Interaktion zwischen den einzelnen Mitarbeitern, beispielsweise zum Austausch von Wissen. Der Spieler gibt in seiner Rolle als Projektleiter den Projektplan vor, und definiert damit seine Anweisung an das Team. Die Mitarbeiter führen die im Projektplan definierten Aufgaben jedoch nicht unreflektiert aus, sondern handeln entsprechend ihres Charakters und ihrer Fähigkeiten.

Ziel dieses Ansatzes ist es, das Verhalten der Mitarbeiter möglichst realitätsnah nachzubilden. Dadurch können unter anderem Aspekte der Mitarbeiterführung in Planspielen besser umgesetzt werden, als mit Lösungen, bei denen die Mitarbeiter sich immer genau an die gegebenen Anweisungen halten.

Durch die Realisierung der Projektbeteiligten als eigenständige Agenten wird ein individuelles Verhalten unterstützt. Die Berechnung dieses Verhaltens ist in einem vollständig eigenständigen Simulationsteil gekapselt, der als *Verhaltensmodelle* bezeichnet wird. Verhaltensmodelle entscheiden, welche Handlung ein Projektbeteiligter zu einem konkreten Zeitpunkt anstrebt durchzuführen. Die eigentliche Durchführung der gewünschten Handlung kann anschließend durch die Tätigkeitsmodelle realisiert werden. Ein solcher Berechnungszyklus besteht prinzipiell aus den drei Schritten:

1. Wahrnehmung der aktuellen Situation in der sich der Agent befindet.
2. Bewertung dieser Situation und Auswahl der passenden Handlung.
3. Ausführung der geplanten Handlung.

Diese Schritte werden in jedem Zeitschritt der Simulation ausgeführt. Schritt 1 und 2 werden dabei durch das Verhaltensmodell des jeweiligen Mitarbeiters realisiert. Als Ergebnis liefert es die Tätigkeit, die der Mitarbeiter im aktuellen Zeitschritt ausführen möchte. Diese Tätigkeit wird in Schritt 3 vom entsprechenden Tätigkeitsmodell umgesetzt.

Die Tätigkeiten, die dem Projektbeteiligten zur Auswahl stehen, sind durch die Tätigkeitsmodelle und das jeweilige Szenario, in dem sich der Projektbeteiligte befindet, definiert. Er kann:

- nichts tun und sich erholen,
- den Arbeitsplatz betreten oder verlassen,
- eine Aufgabe bearbeiten, und so eine Änderung des Projektzustands erzeugen.

Die Menge der möglichen Aufgaben, die ihm zur Auswahl stehen, ist durch das Szenario vorgegeben. Die Beschreibung einer Aufgabe umfasst die Aktivität und das Artefakt auf das diese sich bezieht, außerdem die Gruppe der Bearbeiter, die diese Aufgabe gemeinsam durchführen. Die Grundmenge aller möglichen Aufgaben lässt sich als das kartesische Produkt aus der Menge der im Prozessmodell festgelegten Aktivitäten, den im Produkt vorhandenen Artefakten, und der Potenzmenge der am Projekt beteiligten Mitarbeiter definieren. Welche dieser Aufgaben auf Grund der benötigten Rollen, diverser Vorbedingungen der Artefakte oder weiterer Einschränkungen schlussendlich nicht durchgeführt werden können, ist für die Bildung der Grundmenge irrelevant. Es ist Aufgabe der jeweiligen Agenten diese Einschränkungen zu erkennen und entsprechenden Aufgaben nicht als Handlung auszuwählen.

Für jeden Projektbeteiligten kann eine individuelle Handlungsweise implementiert werden. Die Verhaltensmodelle des Simulationsmodells geben den Rahmen und eine Referenzimplementierung dafür vor. Diese kann entweder direkt verwendet, angepasst und erweitert, oder auch vollständig ausgetauscht werden, um sie an das jeweilige Planspiel und dessen Lernziele anzupassen.

2.6.1. Die subjektive Wahrnehmung der Agenten

Um eine möglichst realitätsnahe Simulation der Projektbeteiligten zu schaffen, basieren die Handlungen eines Agenten ausschließlich auf dem Wissen, welches der jeweilige Agent besitzt. Es werden somit keine Informationen direkt aus der Simulation, der Simulationsumgebung oder dem Simulationsverlauf entnommen und verwendet, die einem Mitarbeiter im Original nicht in gleicher Weise zur Verfügung stehen.

Um das Wissen zu erlangen, das der Agent benötigt, um seine Entscheidungen zu treffen, muss er eine Möglichkeit besitzen, die benötigten Informationen zu erhalten. In der Realität geschieht dies über die Wahrnehmung der Mitarbeiter. Um das nachzubilden enthält das Simulationsmodell zwei Teilmodelle, die diese Aufgabe übernehmen: die *Wahrnehmung der Umgebung* und die *Selbstwahrnehmung*.

Die beiden Wahrnehmungsmodelle basieren auf der Annahme, dass der Mensch die Realität nur unvollständig und verzerrt wahrnehmen kann [Urban, 2005]. Die Prozesse, die ablaufen, um die durch die Sinnesorgane wahrgenommenen Informationen so zu verarbeiten, dass das menschliche Gehirn damit umgehen kann, sind so vielfältig und komplex, dass es mit der Kognitionspsychologie einen eigenen wissenschaftlichen Fachbereich gibt, der sich damit beschäftigt. Das Simulationsmodell abstrahiert von diesen Prozessen und beschränkt sich darauf, die Wahrnehmung als einen Filter zu betrachten, der die tatsächlichen Informationen verzerrt.

Zusätzlich zur Filterung der Informationen, passt das menschliche Gehirn die wahrgenommenen Informationen an, um sie in sein Gesamtmodell der Umwelt zu integrieren. Dieses Modell wird *mentales Modell* genannt [Johnson-Laird, 1983] und stellt eine abstrakte Repräsentation der Realität aus Sicht des betrachteten Individuums dar. Es basiert auf den bisher gemachten Erfahrungen und des in der Vergangenheit erworbenen Wissens.

Das Simulationsmodell setzt die Filter und das mentale Modell in einer stark vereinfachten Form um. Ziel ist es, damit eine Datengrundlage zu schaffen, auf Basis derer die Agenten ihre Handlungsentscheidungen treffen können. Um Vorhersagen und Abschätzungen zu treffen, die für diverse Entscheidungen benötigt werden, setzt das Simulationsmodell teilweise die gleichen Tätigkeits- und Elementarmodelle ein, die auch für die Berechnung des tatsächlichen Arbeitsergebnisses verwendet werden. Die dabei verwendeten Parameter werden jedoch nicht direkt aus der Simulation entnommen, sondern durch Umgebungswahrnehmung und Selbstwahrnehmung zuvor verzerrt. Das Simulationsmodell beschränkt sich dabei auf eine Wahrnehmung des Projektzustands. Komplexere Zusammenhänge werden nicht abgedeckt.

Außerhalb der Handlungsplanung werden die Wahrnehmungsmodule auch verwendet, um die Eingangswerte für die Aktualisierung des Motivationsmodells zu erhalten, da hier nicht die tatsächlichen, sondern die wahrgenommenen Werte relevant sind.

Mentales Modell

Das *mentale Modell* [Johnson-Laird, 1983] eines Agenten dient der Einordnung und Speicherung von Informationen, die der Agent wahrgenommen hat. Dabei wird über die Zeit – bewusst oder unterbewusst – ein System aus Informationen und Regeln geschaffen, das die erworbenen Informationen miteinander verknüpfen. Das menschliche Gehirn versucht die neuen Informationen in ein konsistentes, logisches und zeitlich kontinuierliches Modell einzuordnen. Dadurch werden Teile der Wahrnehmung teils erheblich verzerrt, um in dieses Modell zu passen.

Mentale Modelle helfen ihrem Besitzer, neue Informationen auf Plausibilität und Relevanz zu prüfen, und auf dessen Basis entsprechende Vorhersagen zu treffen. Ein mentales Modell kann dabei beliebig komplex werden. Für das Simulationsmodell wurde ein sehr einfaches Modell umgesetzt, das lediglich die über die Zeit gesammelten Informationen auf einen für die Zukunft zu erwartenden Wert abbildet. Dabei werden die einzelnen in der Simulation vorkommenden Eigenschaften unabhängig voneinander betrachtet. Zu jeder relevanten Eigenschaft, beispielsweise der Arbeitsgeschwindigkeit für eine bestimmte Aufgabe, existiert eine eigenständige Instanz des mentalen Modells. Komplexe Strukturen, die über einen einfachen Skalar hinausgeht, um beispielsweise mehrere Eigenschaften miteinander in Zusammenhang zu bringen, werden durch das Simulationsmodell nicht abgedeckt.

Das Simulationsmodell verwendet mentale Modelle im Rahmen der Umgebungs- und Selbstwahrnehmung, um eine konsistente Einschätzung der einzelnen Eigenschaften über die Zeit zu gewährleisten. Dadurch entstehen stabile Erfahrungswerte, die nicht von einzelnen Ausreißern, sondern durch langfristig erworbenen Informationen bestimmt werden. Die meisten Schätzungen im Simulationsmodell basieren auf Zufallsfunktionen, welche die Filterung und Verzerrung der tatsächlichen Informationen realisieren, um damit die Wahrnehmung der Mitarbeiter im Original zu abstrahieren. Wird eine zufallsbasierte Schätzung der verschiedenen Eigenschaften mehrfach hintereinander durchgeführt, springt das Schätzergebnis entsprechend der Zufallsverteilung. Solche Sprünge sind meist unrealistisch, da ein Mitarbeiter bei mehreren Schätzungen in sehr kurzer Zeit zu sehr ähnlichen Ergebnissen kommen müsste.

Um Sprüngen entgegenzuwirken wird im Simulationsmodell ein Verfahren angewendet, bei dem die Schätzwerte über die Zeit gewichtet werden, indem, wie in Formel 2.48 für eine allgemeine Eigenschaft gezeigt, die Funktion des beschränkten Wachstums angewendet wird. So haben die jüngsten Erfahrungen den größten Einfluss auf den Erwartungswert, je älter eine Erfahrung ist, umso geringer ist ihr Einfluss. Die Adaptionsrate, die angibt, wie stark sich neue Informationen auf die aktuelle Einschätzung auswirken, ist dabei von der jeweils zu beobachtenden Eigenschaft abhängig. In den folgenden Abschnitten ist mit *mentalem Modell* dieses Verfahren gemeint.

$$m_{n+1} = m' - e^{-k\cdot(t_{n+1}-t_n)} \cdot m' + e^{-k\cdot(t_{n+1}-t_n)} \cdot m_n$$

$$\begin{aligned}
t_n &: \text{Zeitpunkt, an dem das Modell zuletzt aktualisiert wurde} \\
t_{n+1} &: \text{aktueller Zeitpunkt, } t_n \leq t_{n+1}. \\
m_n &: \text{Wert des mentalen Modells zum Zeitpunkt } t_n \\
m' &: \text{Schätzwert der abgebildeten Eigenschaft zum Zeitpunkt } t_{n+1} \\
m_{n+1} &: \text{neuer Wert des mentalen Modells} \\
k &: \text{Adaptionsrate des mentalen Modells}
\end{aligned} \tag{2.48}$$

Wahrnehmung der Umgebung

Um sein Handeln planen zu können, benötigt der Projektbeteiligte entsprechende Informationen, auf die er seine Handlung aufbauen kann. Unter anderem betrachtet er dazu das Projekt, an dem er arbeitet. Insbesondere der Zustand der Artefakte des Produkts, das Wissen über die anderen Projektbeteiligten und der Projektplan sind für die Entscheidungsfindung wichtig. Der Projektbeteiligte erlangt die benötigten Informationen mittels seiner persönlichen *Wahrnehmung der Umgebung*.

Diese ist in ein eigenes Modell gekapselt, welches die Filterung und Verzerrung der Informationen realisiert. Es kann für jeden Agenten individuell angepasst werden, um festzulegen, wie dieser seine Umgebung wahrnimmt. Die Berechnung der Werte wird zum Bedarfszeitpunkt vorgenommen, das bedeutet, zu dem Zeitpunkt, an dem der Mitarbeiter sich Gedanken darübermacht.

Das Modell zur Wahrnehmung der Umgebung liefert seinem Besitzer folgende Informationen:

- Für jedes Artefakt des zu entwickelnden Produkts:
  - den aktuellen Fortschritt,
  - die aktuelle Qualität,
  - den Gesamtumfang,
  - den Schwierigkeitsgrad.
- Für jeden am Projekt beteiligten Kollegen:
  - die Qualität, die er durch eine bestimmte Aktivität erzeugen kann,
  - seine Fähigkeiten zu jeder Aktivität bzw. Disziplin,
  - sein Artefaktwissen über jedes Artefakt im Produkt.
- Den aktuellen Zeitdruck, den der betrachtete Mitarbeiter aktuell empfindet.

Da ein Projektplan normalerweise klar und eindeutig formuliert ist, wird davon ausgegangen, dass jeder Projektbeteiligte ihn korrekt versteht. Daher ist es nicht notwendig, ihn zusätzlich zu filtern oder zu verzerren. Der Projektplan wird somit durch die Wahrnehmung nicht verändert.

Jeder Projektbeteiligte ist in der Lage, die wesentlichen Eigenschaften der einzelnen Produktartefakte einzuschätzen. Dazu gehören der zu erwartende Umfang, der aktuelle Fortschritt und die aktuelle Qualität des Artefakts. Es ist unrealistisch, dass ein Mitarbeiter den tatsächlichen Zustand eines Artefakts genau bewerten kann, da er dazu wissen müsste, wie das Artefakt im optimalen Fall in Gestalt und Umfang aussieht. Sowohl Umfänge als auch Qualität können in der Realität lediglich geschätzt werden. Um diesen Aspekt korrekt umzusetzen, wird bei der Schätzung durch einen Projektbeteiligten ein zufälliger Wert, basierend auf dem tatsächlichen Wert gewählt. Wie weit dieser vom tatsächlichen Wert entfernt liegt, das bedeutet, wie genau die Schätzung des Mitarbeiters ist, wird über die Charaktereigenschaften des Mitarbeiters festgelegt, die angeben, wie gut er die entsprechenden Aspekte einschätzen kann. Es wird dabei zwischen den Fähigkeiten Aufwand und Qualität einzuschätzen unterschieden.

Es wird von der Annahme ausgegangen, dass der betrachtete Mitarbeiter die zu schätzende Eigenschaft im Allgemeinen einschätzen kann. Es ist damit zu erwarten, dass die von ihm geschätzten Werte in der Nähe des tatsächlichen Werts liegen, und die Wahrscheinlichkeit einen bestimmten Schätzwert zu wählen abnimmt, je weiter dieser vom tatsächlichen Wert entfernt liegt. Geht man weiterhin davon aus, dass ein über- und unterschätzen der Eigenschaft gleich wahrscheinlich ist, kann ein entsprechender Wahrnehmungsfilter basierend auf der gaußschen Normalverteilung erstellt werden. Die Schätzfähigkeit wird dabei in der Größe der Standardabweichung σ dieser Normalverteilung angegeben. Der eigentliche Schätzprozess entspricht einer Zufallswahl, basierend auf der Verteilungsfunktion der gaußschen Normalverteilung, die durch diese Standardabweichung parametrisiert ist.

Abbildung 2.39 zeigt eine solche Kurve. Der Erwartungswert μ entspricht dabei dem tatsächlichen Wert dessen, was geschätzt wird.

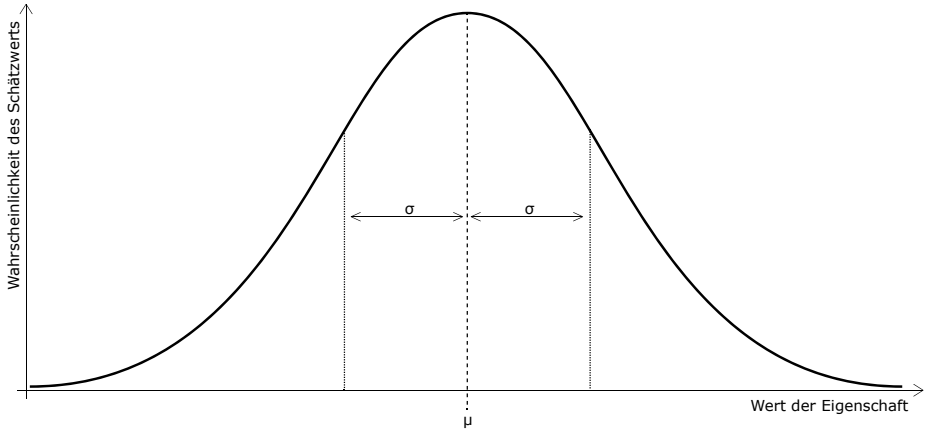


Abbildung 2.39.: Schätzprozess im Rahmen der Wahrnehmung

Die Schätzwerte streuen um den tatsächlichen Wert, wobei die Fähigkeit des Schätzenden angibt, wie stark diese Streuung ist. Diese Art der Schätzung wird für die Eigenschaften Artefaktqualität, Artefaktfortschritt, Artefaktumfang und Arbeitsgeschwindigkeit angewendet. Der Standardwert für die Schätzfähigkeit ist im Simulationsmodell auf $\sigma = 0.1$ festgelegt. Damit liegen 68% der Schätzwert in der Umgebung $\pm 10\%$, und 99% in der Umgebung $\pm 30\%$ um den tatsächlichen Wert. Die Schätzfähigkeit kann über das Szenario bei Bedarf angepasst werden.

Zur Wahrnehmung der Umgebung gehört auch die Einschätzung der anderen am Projekt beteiligten Mitarbeiter. Die Schätzung der Fähigkeiten und des Artefaktwissen der anderen Projektbeteiligten wird äquivalent zur Schätzung der Artefakteigenschaften durchgeführt, und in dazugehörige mentale Modelle abgelegt. Die der Schätzung zugrundeliegenden Standardnormalverteilung wird ebenfalls durch einen Wert im Charakter der Projektbeteiligten festgelegt. Dieser gibt an, wie genau der Projektbeteiligte seine Mitmenschen einschätzen kann. Um die Arbeitsqualität eines Kollegen zu ermitteln, wird das entsprechende Elementarmodell mit den tatsächlichen Eigenschaften des Kollegen verwendet, und das Ergebnis über die Zufallsfunktion gefiltert.

Da der Zeitdruck, den ein Projektbeteiligter erfährt, seiner Wahrnehmung zuzurechnen ist, ist die Regel zur Berechnung des Zeitdrucks, die in 2.4.1 im Rahmen der Elementarmodelle beschrieben ist, im Modul der Umgebungswahrnehmung realisiert. Das Gefühl von Zeitdruck basiert auf der Erkenntnis, nicht genügend Zeit für die anstehenden Aufgaben zu haben. Diese Erkenntnis wiederum basiert auf der Wahrnehmung und Einschätzung der offenen Aufgaben und der eigenen Fähigkeit, die Aufgabe in der vorgegebenen Zeit zu beenden. Damit enthält die Wahrnehmung des Zeitdrucks mit der Wahrnehmung der eigenen Fähigkeiten auch Aspekte der Selbstwahrnehmung. Es wäre deshalb auch denkbar, sie dem Modul der Selbstwahrnehmung zuzurechnen.

Für einige Planspielszenarien ist es sinnvoll auch die Informationen, die dem Spieler als Entscheidungsgrundlage präsentiert werden, durch ein Wahrnehmungsmodul zu filtern, da auch ein Projektleiter typischerweise nur über eingeschränkte und nicht exakte Informationen über das Projekt verfügt.

Selbstwahrnehmung

Das Modell der *Selbstwahrnehmung* gibt vor, wie ein Projektbeteiligter sich selbst, besonders seinen Zustand und seine Fähigkeiten wahrnimmt. Die mittels der Selbstwahrnehmung gewonnenen Informationen werden meist im Rahmen von Ergebnisprognosen bestimmter Aufgaben benötigt. So kann ein Projektbeteiligter beispielsweise abschätzen, wann er mit einer bestimmten Aufgabe fertig ist.

Wie auch bei der Umgebungswahrnehmung werden die tatsächlichen Werte dazu mittels einer auf der Normalverteilung basierenden Zufallsfunktion verändert. Die Standardabweichung der Verteilungsfunktion wird durch einen Wert im Charakter der Projektbeteiligten definiert, der angibt, wie gut

sich der Mitarbeiter selbst einschätzen kann. Das Simulationsmodell setzt für die Projektbeteiligten die Schätzung folgender Eigenschaften um:

- die eigene Fähigkeit zu jeder Aktivität bzw. Disziplin,
- das eigene Artefaktwissen zu jedem Artefakt des Produkts,
- die angenommene Arbeitsgeschwindigkeit zu jeder im Szenario definierten Aktivität.

Die Ergebnisse der Schätzungen werden in entsprechenden mentalen Modellen abgelegt. Die Werte werden bei Bedarf berechnet. Eine Ausnahme bildet die Arbeitsgeschwindigkeit, die im Anschluss an jede ausgeführte Aktivität berechnet wird. Dazu wird der durch die Aktivität erzeugte Projektfortschritt als Differenz zwischen dem Zustand vor und nach der Aktivität ermittelt, der Differenzwert durch die Wahrnehmungsfunktion gefiltert, und anschließend im mentalen Modell abgelegt.

Zusätzlich enthält die Selbstwahrnehmung eine Filterfunktion, um einen erzielten Lernfortschritt einzuschätzen, der beispielsweise verwendet wird, um das Motivationsmodell zu aktualisieren.

2.6.2. Handlungsplanung

Die Aufgabe der Handlungsplanung ist es, die verfügbare Zeit der einzelnen Mitarbeiter auf die verschiedenen, ihnen zur Verfügung stehenden Tätigkeit zu verteilen.

Dieser Abschnitt beschreibt die Verhaltensmodelle, die festlegen, wie ein Agent entscheidet, welche Tätigkeit er zu welchem Zeitpunkt ausführt. Dieser Prozess wird als Handlungsplanung bezeichnet. Eine Handlung ist dabei eine Menge von geplanten oder durchgeführten Tätigkeiten, die typischerweise darauf ausgelegt sind, ein bestimmtes Ziel zu erreichen.

Die Handlungsplanung kann individuell für jeden einzelnen Agenten realisiert werden. Dazu wird für jeden Agenten eine eigene Planungskomponente instanziiert, die sein Verhalten umsetzt. So können individuelle Implementierungen oder Konfigurationen der Planungskomponenten einfach umgesetzt und integriert werden. Die gemeinsame Schnittstelle, die dabei verwendet wird, besteht aus einer Methode, die zu einem gegebenen Zeitpunkt eine konkrete Handlung liefert. Außerdem enthält sie eine weitere Methode, über die beim Agenten nachgefragt werden kann, ob er an einer gemeinsamen Aktivität mit anderen Mitarbeitern teilnimmt. So bleibt jeder Agent selbstbestimmt.

Im Verlauf der Entwicklung hat sich herausgestellt, dass es von Vorteil ist, die Planung der Tätigkeiten in zwei Schritten vorzunehmen. Im ersten Schritt entscheidet der Agent darüber, ob er arbeitet, oder ob er seiner Freizeit nachgeht. Dieser Schritt wird im Folgenden als Arbeitszeitplanung bezeichnen. Entscheidet er sich für Arbeiten, bestimmt er im zweiten Schritt eine oder mehrere konkrete Tätigkeiten, die er durchführen möchte. Entscheidet der Agent sich hingegen für Freizeit, ist der zweite Schritt der Handlungsplanung trivial, da innerhalb der Freizeit alle Tätigkeiten in einer gemeinsamen Erholungstätigkeit abstrahiert sind, und nur diese zur Auswahl steht. Durch diese Abstraktion werden die Grenzen des Simulationsmodells umgesetzt, die es auf die Arbeitswelt beschränken. Durch das Vorgehen in zwei Schritten lässt sich das Verhaltensmodell besser strukturieren, und ist somit leichter verständlich, wartbar und erweiterbar.

Wie in Abbildung 2.40 gezeigt, enthält das Simulationsmodell drei aufeinander aufbauende Umsetzungen des Verhaltensmodells. Jede Umsetzung erweitert dabei das vorangehende Modell um weitere Aspekte. Die abstrakte Klasse Verhaltensmodell definiert die notwendigen Schnittstellen, um das Modell in das Gesamtmodell einzubinden. Eine entsprechende Beschreibung der Architektur findet sich in Abschnitt 2.7. Alle Umsetzungen des Verhaltensmodells können in Planspielen eingesetzt werden. Die Auswahl des konkreten Modells ist dabei von Szenario und Lernzielen des jeweiligen Planspiels abhängig. Im Folgenden werden die Modelle zur direkten Planung und zur dynamischen Arbeitszeitplanung beschrieben. Das VIE-Modell zur Handlungsplanung wird anschließend in den weiteren Abschnitten behandelt.

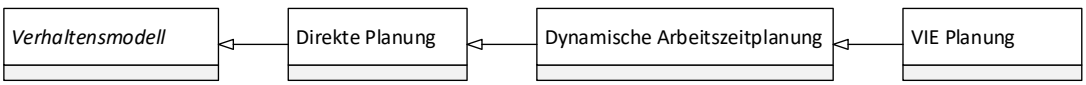


Abbildung 2.40.: Verhaltensmodelle des Simulationsmodells

Direkte Planung

Das Modul *Direkte Planung* setzt die Handlungsplanung auf die direkteste und einfachste Art um.

Arbeitsbeginn und Arbeitsende entsprechen exakt der vertraglich festgelegten Arbeitszeit. Damit ist definiert, welche Zeiträume zur Arbeitszeit, und welche zur Freizeit gehören. Überstunden oder Minusstunden werden nicht gemacht. Beim zeitlichen Übergang zwischen Arbeitszeit und Freizeit führt der Agent die entsprechende Tätigkeit zum Betreten beziehungsweise Verlassen des Arbeitsplatzes durch.

Während der Freizeit wird die Erholungstätigkeit durchgeführt. Alle Tätigkeiten während der Arbeitszeit werden direkt aus dem Projektplan entnommen und unreflektiert ausgeführt, unabhängig davon, ob die notwendigen Voraussetzungen dafür gegeben sind oder nicht. Gibt der Projektplan für einen Zeitabschnitt keine Aufgabe vor, macht der Agent Pause, indem er die Erholungstätigkeit ausführt. Da der Projektplan vom Spieler geschrieben wird, legt dieser die Tätigkeiten der Projektmitarbeiter damit direkt fest.

Bei der konkreten Planung einer Aufgabentätigkeit greift der Agent auf die im Projektplan vorhandenen Vorgaben zurück. Artefakt, Aktivität, Zeitraum und gegebenenfalls weitere Personen, mit denen gemeinsam gearbeitet werden soll, werden direkt übernommen.

Bei der Planung von Fortschritts- und Qualitätssicherungstätigkeiten wird auch die *spontane Kommunikation* (vgl. 2.5.3) geplant. Hierbei ermittelt der Agent das Artefaktwissen, das er für die geplante Handlung benötigt, und evaluiert, ob er durch kurze Gespräche mit seinen Kollegen gegebenenfalls vorhandene Wissenslücken füllen kann. Dazu wird zuerst der am besten geeignete Kollege gesucht und anschließend ermittelt, welcher Anteil der für die Aufgabe vorgesehenen Zeit in die Kommunikation investiert werden soll.

Zunächst wird dazu das gemeinsame Wissen der bearbeitenden Gruppe ermittelt, indem die Vereinigung über das Artefaktwissen der einzelnen Gruppenmitglieder gebildet wird. Anschließend wird die jeweilige Differenz zum Artefaktwissen der verfügbaren Kollegen, die nicht Teil der Gruppe sind, ermittelt. Verfügbar bedeutet, der Kollege ist am Arbeitsplatz und bereit einen Teil seiner Arbeitszeit für die Kommunikation zur Verfügung zu stellen. Die Zeit, die ein Mitarbeiter einem anderen zur Verfügung stellt, hängt dabei von seiner noch übrigen Arbeitszeit am betrachteten Tag, seinem aktuellen Zeitdruck und der Sympathie für den fragenden Mitarbeiter ab, und berechnet sich wie in Formel 2.49 gezeigt. Dabei wird der Zeitdruck auf maximal 5 begrenzt, um die Berechnung auch bei extremen Zeitdruckwerten stabil zu halten. Ein Mitarbeiter, der unter sehr großem Zeitdruck steht, stellt damit immer noch mindestens 20% seiner Zeit zur für die Kommunikation zur Verfügung, was sich im Rahmen des Modelltunings als ein gut funktionierender Wert herausgestellt hat.

$$\hat{t_\Delta} = \frac{{t_\Delta}' \cdot \psi_\mathrm{m}}{\min(\rho, 5)}$$

$\hat{t_\Delta}$: Zeit in Stunden, die der Kollege bereit ist zu investieren
${t_\Delta}'$: Zeit in Stunden, die dem Kollegen zur Verfügung steht
ψ_m : Sympathie/Antipathie des Kollegen gegenüber dem Fragensteller
ρ : Zeitdruck, den der Kollege momentan empfindet

(2.49)

Kollegen, die weniger als 5 Minuten Zeit zur Verfügung stellen, werden als nicht verfügbar betrachtet. Unter den verfügbaren Kollegen wird derjenige ausgewählt, der die höchste Differenz zum Wissensstand der Gruppe besitzt, und damit derjenige, der über das beste Potential verfügt, der Gruppe zu helfen.

Anschließend wird die tatsächlich in die spontane Kommunikation investierte Zeit ermittelt. Dazu wird der Einfluss der Abdeckung des Artefaktwissens auf die Aktivitätsdurchführung betrachtet, der angibt, wie sich dieser auf die Arbeitsergebnisse auswirkt. Außerdem wird die maximal vom entsprechenden Kollegen zur Verfügung gestellte Zeit und die erwartete Kommunikationsgeschwindigkeit

berücksichtigt. Mittels Formel 2.50 werden die Einflussfaktoren in die benötigte Zeit umgerechnet.

$$t_\Delta = \min((|\hat{K}| \cdot s \cdot 0.02) \cdot (1 - K_{\text{Einfluss}}(k_\Delta)), \hat{t_\Delta})$$

$$\begin{aligned} t_\Delta &: \text{Zeit in Stunden, die in die spontane Kommunikation investiert wird} \\ |\hat{K}| &: \text{Artefaktwissen des Kommunikationspartners, das der Gruppe fehlt} \\ s &: \text{Umfang des betrachteten Artefakts} \\ K_{\text{Einfluss}} &: \text{Einfluss der Wissensabdeckung auf die Bearbeitung} \\ k_\Delta &: \text{Wissensabdeckung der Bearbeitergruppe} \\ \hat{t}_\Delta &: \text{Zeit in Stunden, die der Kommunikationspartner zur Verfügung stellt} \end{aligned} \tag{2.50}$$

Der erste Faktor des Ausdrucks beschreibt den Zeitaufwand, der geschätzt wird, um das nachgefragte Wissen zu kommunizieren. Je höher dieser Aufwand ist, umso mehr Zeit muss investiert werden. Er ergibt sich als Produkt des zu kommunizierenden Wissens und der durchschnittlich benötigten Zeit von $0.02^h/_U$. Da das zu kommunizierende Wissen relativ vorliegt, muss es zusätzlich mit dem Artefaktumfang multipliziert werden, um es in zu kommunizierende Arbeitseinheiten umzurechnen. Der zweite Faktor gibt an, wie wichtig das Artefaktwissen für die beabsichtigte Bearbeitung ist. Je höher dieser Wert ist, umso mehr Zeit ist der Bearbeiter bereit in die Kommunikation zu investieren. Die Berechnung stellt außerdem sicher, dass die vom Kollegen für die Kommunikation angebotene Zeit nicht überschritten wird.

Ist eine Kommunikationszeit von mindestens 5 Minuten mit einem Kollegen möglich, wird eine zusätzliche Tätigkeit der spontanen Kommunikation vor der ursprünglichen Tätigkeit geplant. Ansonsten wird nur die ursprüngliche Tätigkeit geplant. Damit wird verhindert, dass die Kollegen sehr häufig wegen kleinen Wissenslücken gestört werden. Dies würde vor allem bei sehr kleinen Zeitschritten passieren, da hier die Planung häufiger ausgeführt wird, als es bei großen Zeitschritten der Fall ist. Liegt der Kommunikationsbedarf unter 5 Minuten, versucht der Mitarbeiter über das Studieren des entsprechenden Artefakts dieses Defizit, wie im dazugehörigen Modell vorgesehen (vgl. Abschnitt 2.5.3), selbst zu beheben. Gelingt ihm dies nicht, summieren sich die Defizite über die Zeitschritte hinweg auf, bis eine benötigte Kommunikationszeit von 5 Minuten überschritten wird, und es sich lohnt, den Kollegen bei seiner Arbeit zu unterbrechen.

Dynamische Arbeitszeitplanung

Dass die tatsächliche Arbeitszeit exakt der vertraglichen Arbeitszeit entspricht, ist eher ungewöhnlich. In der Realität schwankt die tägliche Arbeitszeit je nach Arbeitslast. So entstehen während Phasen in denen viel Arbeit anfällt Überstunden, und in Phasen mit weniger Arbeit Minusstunden, welche die zuvor gemachten Überstunden ausgleichen können. Dazu kommt, dass häufig eine *dynamische Arbeitszeitplanung* praktiziert wird. Mitarbeiter können in einem gewissen Rahmen selbst bestimmen, wann sie ihren Arbeitstag beginnen und beenden, anstatt sich an starre Zeiten halten zu müssen. Oft wird dazu eine Kernarbeitszeit definiert, zu der Anwesenheitspflicht herrscht. Der Zeitraum um diese Kernarbeitszeit kann jedoch selbst gestaltet werden.

Um diesen Aspekt umzusetzen, erweitert das Simulationsmodell die direkte Planung zur dynamischen Arbeitszeitplanung. Dabei bleibt die Planung der eigentlichen Tätigkeiten unverändert, die Zeitpunkte, zu denen der Mitarbeiter zwischen Arbeit und Freizeit wechselt, werden jedoch dynamisch gewählt. Dazu werden die Zeitpunkte, zu denen ein Mitarbeiter das Betreten und Verlassen des Arbeitsplatzes plant, in jedem Planungsschritt neu berechnet. Die weiteren geplanten Tätigkeiten werden jeweils, wie auch bei der direkten Planung, an den durch diese Zeitpunkte definierten Phasen der Arbeitszeit und Freizeit ausgerichtet.

Um den Zeitpunkt des Arbeitsbeginns zu berechnen, wird als Erstes der nächste Arbeitstag unter Berücksichtigung von Wochenende und Urlaub ermittelt. Anschließend wird für diesen Tag die Uhrzeit des Arbeitsbeginns berechnet. Die Basis dafür bildet der vertraglich festgelegte Arbeitsbeginn, der um maximal ± 1 Stunde verschoben wird. Die Verschiebung wird wie in Formel 2.51 gezeigt berechnet.

$$
\begin{aligned}
f_m &= 1 - m \\
f_{t_{\Delta\text{Überstunden}}} &= \left(\left(\frac{1 + 0.1 \cdot t_{\Delta\text{Überstunden}}}{0.1 \cdot t_{\Delta\text{Überstunden}}} \right)^{0.1 \cdot t_{\Delta\text{Überstunden}}} - 1 \right) \cdot 0.5 \\
f_\rho &= 1 - \rho^{1.3} \\
t_{\Delta\pm} &= \min\left(\max\left(f_m + f_\rho + f_{t_{\Delta\text{Überstunden}}}\right), -1\right), 1\right)
\end{aligned}
\tag{2.51}
$$

$t_{\Delta\pm}$: Abweichung zum vertraglichen Arbeitszeitbeginn in Stunden
$t_{\Delta\text{Überstunden}}$: Überstunden, entspricht $\max(t_{\Delta\text{Konto}}, 0)$
m : Motivation zur Leistungsbereitschaft
ρ : Zeitdruck

Dazu wird angenommen, dass die Entscheidung, früher oder später zur Arbeit zu gehen, im Wesentlichen von drei Faktoren abhängt. Bringt ein Mitarbeiter eine grundsätzlich höhere Motivation mit, wird er von sich aus früher zur Arbeit gehen. Ein anderer Grund früher mit der Arbeit zu beginnen, ist eine hohe Arbeitslast, und damit ein hoher Zeitdruck, den der Mitarbeiter empfindet. Entsprechend umgekehrt beginnen Mitarbeiter mit niedriger Motivation oder mit geringem Zeitdruck später. Der dritte Faktor ist das Arbeitszeitkonto des Mitarbeiters. Sobald ein Mitarbeiter eine gewisse Menge an Überstunden gesammelt hat, wird er versuchen, diese abzubauen. Neben dahingehenden Anweisungen seines Vorgesetzten, fühlt sich der Mitarbeiter auch weniger verpflichtet, rechtzeitig zur Arbeit zu erscheinen, da er durch die Überstunden schon in Vorleistung gegangen ist. Dabei nimmt das Simulationsmodell an, dass sich die Anzahl der Überstunden nicht linear auswirkt, sondern der Einfluss einer Schranke entgegenstrebt.

Das Gegenstück zum Arbeitsbeginn ist das Ende der täglichen Arbeitszeit. Um den Endzeitpunkt zu berechnen, wird zunächst anhand des tatsächlichen Arbeitsbeginns und der vertraglichen täglichen Arbeitszeit das reguläre Arbeitszeitende ermittelt. Wird dieser Zeitpunkt gewählt, werden weder Überstunden noch Minusstunden gemacht. Anschließend wird, wie in Formel 2.52 gezeigt, die Abweichung von diesem Zeitpunkt berechnet.

$$
t_{\Delta\pm} = t_{\Delta\text{Vertrag}} \cdot \begin{cases} (\rho - 1) \cdot 0.2 & , t_{\Delta\text{Konto}} > t_{\Delta\text{Vertrag}} \wedge \rho < 1 \\ \min\left(\max\left((\rho - 1) \cdot 0.5, 0\right), 1\right) \cdot (\mathrm{e} \cdot 0.5)^2 & , \text{sonst} \end{cases}
\tag{2.52}
$$

$t_{\Delta\pm}$: Abweichung zum regulären Arbeitszeitende in Stunden
$t_{\Delta\text{Konto}}$: Arbeitszeitkonto in Stunden
$t_{\Delta\text{Vertrag}}$: vertragliche Arbeitszeit pro Tag in Stunden
ρ : Zeitdruck
e : Energie

Die Abweichung wird relativ zur vertraglichen Arbeitszeit festgelegt. Dabei werden zwei Fälle unterschieden: Verfügt der Mitarbeiter über mehr als einen vollständigen Arbeitstag an Überstunden, und hat gleichzeitig mehr Zeit zur Verfügung als er für die Erledigung seiner Aufgaben benötigt, wird die Arbeitszeit verkürzt, um Überstunden abzubauen. In allen anderen Fällen werden in Abhängigkeit von Zeitdruck und Fitness des Mitarbeiters Überstunden gemacht.

Da alle Parameter in bestimmten Situationen extreme Werte annehmen können, beispielsweise kann durch Fehlplanung ein unverhältnismäßig hoher Zeitdruck entstehen, muss die Auswirkung der Arbeitszeitplanung beschränkt werden, um das Simulationsmodell stabil, und die Simulationsergebnisse realistisch zu halten. Die konkreten Funktionen, ihre Parameter und die Einflussgrenzen haben sich während der Entwicklung des Modells und seiner Evaluation ergeben. Insbesondere die Abstimmung zwischen Zeitdruck, Arbeitszeit und Fitness der Mitarbeiter, und die damit verbundene Produktivität, sowohl kurzfristig als auch langfristig betrachtet, bieten gute Indikatoren, wie gut eine bestimmte Gewichtung geeignet ist. Dabei hat sich herausgestellt, dass an einigen Stellen ein

linearer Einfluss nicht ausreicht, um die gewünschten Effekte stabil zu erzeugen.

Im Zuge der Entwicklung der dynamischen Arbeitszeitplanung wurden ein von Arbeitsbeginn und Arbeitsende unabhängiger Ansatz getestet. Dabei wurde die Frage, ob ein Mitarbeiter zu einem bestimmten Zeitpunkt arbeitet, in die Faktoren *Arbeitsmotivation*, *Pflichtgefühl*, *Erholungsbedürfnis* und *Erschöpfungsgefühl* zerlegt. Arbeitsmotivation und Pflichtgefühl gehen positiv, Erholungsbedürfnis und Erschöpfungsgefühl negativ in die Bewertung ein. Die einzelnen Faktoren werden als Kurven im 24-Stunden-Intervall dargestellt, die mittels Basiskurven, wie beispielsweise der Tagesleistungskurve oder eine Kurve für den Biorhythmus gebildet, und durch Faktoren wie Energieniveau, Zeitdruck, und Zeitkonto entlang der y-Achse skaliert und verschoben werden. Die Summe der Kurven ergibt zu jedem Zeitpunkt einen Wert, der als Indikator verwendet werden kann, um zu entscheiden, ob der Projektbeteiligte arbeitet oder nicht.

Die Idee hinter diesem Ansatz war, eine detailliertere Steuerung der Arbeitszeit zu erreichen, die individuell für den einzelnen Mitarbeiter konfiguriert werden kann. Es hat sich herausgestellt, dass der Ansatz prinzipiell funktioniert, jedoch schnell instabil wird, wenn die Parameter und Kurven nicht sehr exakt gewählt werden. Das äußert sich beispielsweise darin, dass Mitarbeiter mitten in der Nacht anfangen zu arbeiten oder zu stark von der vertraglich vereinbarten Arbeitszeit abweichen. Dadurch kann sich wiederum der Arbeitszeitraum der einzelnen Mitarbeiter verschieben. Das geht mitunter soweit, dass Mitarbeiter statt tagsüber nachts arbeiten. Gemeinsame Aktivitäten werden dadurch ineffizient oder sogar unmöglich. Insgesamt hat sich gezeigt, dass das Simulationsmodell durch diesen Ansatz zu schnell instabil wird, wenn die Parameter nicht präzise gewählt werden. Das widerspricht dem Ziel, das Modell einfach handhab- und einsetzbar zu halten. Deswegen wurde dieser Ansatz verworfen und auf das oben dargestellte Verfahren reduziert.

2.6.3. VIE-Planung

Das *VIE-Planungsmodell* erweitert die dynamische Arbeitszeitplanung um Planungsaspekte für die eigentlichen Aufgaben, die ein Mitarbeiter während seiner Arbeitszeit bearbeitet. Im Gegensatz zu den zuvor besprochenen Planungsmodulen, erlaubt es den Mitarbeitern vom vorgegebenen Projektplan abzuweichen.

VIE steht für Valenz-Instrumentierung-Erwartung und beschreibt eine von Victor Vroom aufgestellte Theorie der Handlungsplanung [Vroom, 1994]. Das VIE-Planungsmodul basiert auf dieser und ähnlicher Ideen, aus dem Bereich der Handlungsplanung als Teilgebiet der Psychologie. Die primären Aspekte des Moduls sind zusammen mit einigen Beispielen auch in [Nassal, 2016] beschrieben.

Die mit der VIE-Planung umgesetzte Handlungsplanung unterscheidet sich vom typischerweise in Planspielen eingesetzten Vorgehen. Herkömmliche Ansätze verwenden meist einen Projektplan, um die Tätigkeiten der simulierten Mitarbeiter zu wählen. Der Spieler gibt so exakt vor, welcher Mitarbeiter zu einem bestimmten Zeitpunkt was zu tun hat. Die Mitarbeiter selbst haben nur sehr geringe Freiheitsgrade in ihrer Handlungsweise, die sich oft auf das Festlegen von Arbeitspausen beschränken. Der Spieler wird dadurch gezwungen sehr detailliert zu planen. In der Realität gibt der Projektleiter jedoch nur einen groben Plan vor, das Mikromanagement übernehmen die Mitarbeiter selbst. Sie entscheiden beispielsweise eigenständig, wann sie Code schreiben, diesen dokumentieren, testen oder mit anderen darüber reden. Dabei unterscheidet sich das Verhalten der einzelnen Mitarbeiter im Allgemeinen voneinander. Wie jeder einzelne handelt, hängt von seinem Charakter und den aktuellen Umständen ab. Diese Tatsache zu erkennen, zu verstehen und darauf zu reagieren, ist ein wichtiges Lernziel für angehende Projektmanager. Entsprechend ist es wichtig, diesen Aspekt in Planspielen adäquat umzusetzen.

Die Idee für eine neue Art der Handlungsplanung lieferte das PECS-Referenzmodell aus [Urban, 2000], welches eine grundsätzliche Architektur für Handlungsplanungsmodelle für Agenten vorschlägt, die nicht ausschließlich auf logischen Entscheidungen basiert, sondern auch emotionale und soziale Aspekte berücksichtigen kann. PECS liefert jedoch nur wenig konkrete Hinweise, wie die einzelnen Komponenten ausgestaltet werden sollen, daher stützt sich das VIE-Modul des Simulationsmodells hauptsächlich auf die VIE-Theorie von Vroom [Vroom, 1994] und die Motivationstheorie von Pritchard und Ashwood [Pritchard, 2008].

In der Psychologie wird die Wahl einer Handlung aus einer Menge von Handlungsalternativen unter anderem mit der *Erwartungs-mal-Wert-Theorie* erklärt [Schuler, 2006, S. 390]. Dabei wird von der betrachteten Person diejenige Handlung gewählt, die unter Berücksichtigung des Nutzens des Handlungsergebnisses und der Wahrscheinlichkeit, dass dieses Ergebnis durch die gewählte Handlung eintritt, am attraktivsten ist.

Victor Vroom beschreibt in [Vroom, 1994] ein auf dieser Theorie basierendes Auswahlverfahren für Handlungen. Dieser Ansatz gehört zu den wenigen, für die die Zusammenhänge zumindest in Grundzügen mathematisch definiert sind. Laut Vroom haben drei Größen einen Einfluss auf die Wahl einer Handlung:

- Die **Erwartung** gibt an, wie hoch die Wahrscheinlichkeit ist, dass ein beabsichtigtes Ergebnis durch eine bestimmte Handlung auch tatsächlich erreicht wird. Die Erwartung wird beispielsweise durch die Fähigkeiten der handelnden Person beeinflusst.

- Die **Instrumentalität** beschreibt, wie sich ein bestimmtes Handlungsergebnis auf mögliche Folgen auswirkt. Eine Folge beschreibt dabei ein Ergebnis in einem größeren Kontext, beispielsweise eine Beförderung. Die Instrumentalität kann sich dabei sowohl positiv, als auch negativ auf eine Folge auswirken.

- Die **Valenz** einer Folge beschreibt, wie wünschenswert diese Folge für die betrachtete Person ist. Beispielsweise kann eine Beförderung erwünscht sein, wenn sie ein höheres Gehalt bedeutet, oder unerwünscht sein, wenn sie zu weniger Freizeit führt.

Vroom geht davon aus, dass die Person bewusst oder unterbewusst Handlungsalternativen gegeneinander abwiegt, indem diese in Bezug auf den persönlichen Nutzen evaluiert werden. Dazu werden die drei Einflussfaktoren wie in Formel 2.53 kombiniert [Nassal, 2016].

$$\begin{aligned} &\text{Motivationskraft}(\tau) = \text{Valenz}(r) \cdot \text{Erwartung}(r) \\ &\text{Valenz}(r) = \sum_{o_i \in O} \text{Valenz}(o_i) \cdot \text{Instrumentalität}(o_i, r) \end{aligned} \tag{2.53}$$

r : erwartetes Ergebnis der Handlung τ

O : Menge der berücksichtigten Folgen

Es wird ermittelt, wie hoch die Valenz einer Handlung H ist, indem deren gewünschtes Ergebnis r in Bezug auf seine Folgen betrachtete wird. Zuerst wird die Valenz des Handlungsergebnisses berechnet, die angibt, wie wünschenswert das Ergebnis aus Sicht der Person ist. Dazu wird für jede relevante Folge o_i betrachtet, wie sich das Ergebnis auf die Folge auswirkt. Dies wird durch die Instrumentalität bestimmt, die beispielsweise von den Zielen, Leitlinien und dem Führungsstil des Vorgesetzten abhängt. Zusammen mit der Valenz der Folge, die wiederum von den persönlichen Zielen und dem Charakter des Mitarbeiters abhängt, ergibt sich die Valenz des Ergebnisses. Um die Motivationskraft der Handlung zu ermitteln, wird diese Valenz mit der Erwartung in Form der Wahrscheinlichkeit, dass das betrachtete Ergebnis unter den gegebenen Rahmenbedingungen eintritt, gewichtet.

Die Motivationskraft bestimmter Handlungsalternativen kann laut Vroom von einem Vorgesetzten primär über die Instrumentalität beeinflusst werden. Rahmenbedingungen, die sich auf die Erwartung auswirken, können gar nicht, oder nur indirekt beeinflusst werden. So können zum Beispiel durch eine Schulung die Fähigkeiten eines Mitarbeiters verbessert werden. Die Instrumentalität hingegen lässt sich häufig direkt verändern. So kann beispielsweise ein gewünschtes Ergebnis mit einer Folge belohnt werden, die für den Mitarbeiter wünschenswert ist. Ob es sich dabei um eine Beförderung oder mehr Freizeit handeln sollte, hängt vom jeweiligen Mitarbeiter und dessen persönlichen Zielen ab. Es ist für einen Vorgesetzten daher wichtig, seine Mitarbeiter und ihre Bedürfnisse zu kennen, um sein Team optimal motivieren zu können.

Die VIE-Theorie eignet sich gut für die Umsetzung der Handlungsplanung in Planspielen, da sie zum einen eine rechnerische Grundlage bietet, zum anderen gut strukturiert ist. Die Berechnung der

Erwartung eines Ergebnisses kann individuell für jeden Agenten auf Basis dessen Wahrnehmung und Fähigkeiten umgesetzt werden. Die Valenz der Folgen kann ebenfalls individuell für die einzelnen Agenten über ihren Charakter festgelegt werden. Die Beeinflussung der Motivation durch den Spieler wird durch eine entsprechende Instrumentalität umgesetzt, die auf das konkrete Planspiel, seine Lernziele und die damit verbundenen möglichen Spieleraktionen angepasst werden kann.

Ein weiteres Modell, welches die menschliche Handlungsentscheidung abbildet, stammt von Robert D. Pritchard und Elissa L. Ashwood [Pritchard, 2008]. Es basiert auf der Theorie von Naylor, Pritchard und Ilgen [Naylor, 1980], und ist wie auch das Modell von Vroom dem Bereich der Erwartungstheorie zuzuschreiben. Pritchard und Ashwood entwickelten dieses Modell mit dem Ziel, es auch für die Praxis anwendbar zu machen. Es basiert auf der Annahme, dass jeder Mensch einen gewissen Pool an Ressourcen, wie Zeit und Energie, zur Verfügung hat, und diesen auf unterschiedliche Handlungen verteilt, sodass dadurch seine Bedürfnisse möglichst gut befriedigt werden.

Pritchard und Ashwood gehen davon aus, dass alle Menschen die gleichen Grundbedürfnisse haben, die jedoch unterschiedlich stark ausgeprägt sind. Die Maslowsche Bedürfnishierarchie [Maslow, 1943] beschreibt eine mögliche Sicht auf diese Grundbedürfnisse und ihre Zusammenhänge, und ist das wohl am weitesten verbreitete Modell in diesem Bereich.

Laut Pritchard und Ashwood lässt sich die Handlungsplanung als eine Evaluationskette aus *Handlung*, *Ergebnis*, *Evaluation*, *Folgen* und *Bedürfnisbefriedigung* betrachten. Die einzelnen Teile werden dabei durch *Kontingenzen* verbunden. Abbildung 2.41 zeigt diese Handlungskette. Die gestrichelten Pfeile beschreiben im Simulationsmodell vorgenommene Modifikationen gegenüber dem Original von Pritchard und Ashwood, die weiter unten diskutiert werden.

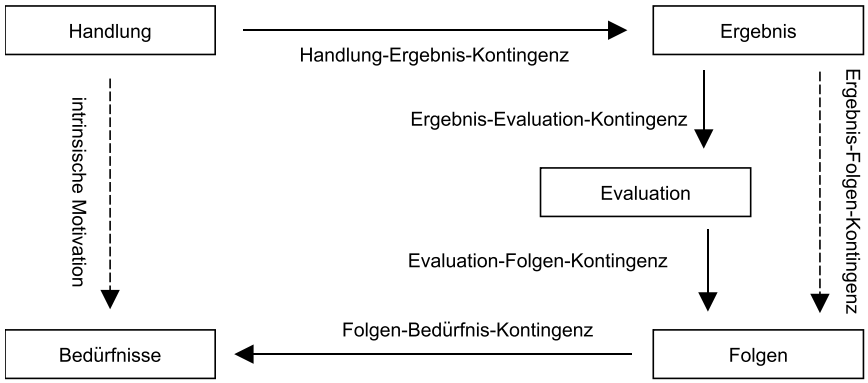


Abbildung 2.41.: Motivationsprozess nach Pritchard und Ashwood (inklusive Modifikationen) [Nassal, 2016]

Die Kontingenz zwischen zwei Elementen beschreibt die Stärke ihres Zusammenhangs. Die Kontingenzen im Modell von Pritchard und Ashwood sind gerichtet. Eine hohe Kontingenz ist dabei gleichbedeutend mit einer hohen Wahrscheinlichkeit, dass ein Teil in einem anderen resultiert. So führt eine Handlung zu einem Ergebnis, und die Kontingenz zwischen Handlung und Ergebnis beschreibt, wie sehr die betrachtete Person die Kontrolle über das von ihr erzeugte Ergebnis hat. Eine sehr hohe Kontingenz bedeutet in diesem Fall, dass die Person das Ergebnis vollständig durch ihre Handlung bestimmt, während eine sehr niedrige Kontingenz bedeutet, dass sie keinerlei Einfluss auf das Ergebnis nehmen kann.

Beispielsweise ist bei einem Mitarbeiter mit guten Fähigkeiten und entsprechenden Mitteln die Handlungs-Ergebnis-Kontingenz höher, als bei einem Mitarbeiter, der die entsprechende Handlung nicht beherrscht, oder dem die dazu benötigten Mittel nicht zur Verfügung stehen. Eine niedrige Kontingenz ergibt sich auch, wenn Dritte einen Einfluss auf die Ergebnisse haben.

Die Kontingenz zwischen Ergebnis und Evaluation beschreibt den Zusammenhang zwischen einem erzielten Ergebnis und der dazugehörigen Bewertung, beispielsweise durch Kollegen oder Vorgesetzte. Hier ist eine hohe Kontingenz vorhanden, wenn ein Mitarbeiter weiß, dass ein gutes Ergebnis auch mit Sicherheit positiv bewertet wird.

Eine positive Evaluation führt im Idealfall zu einer positiven Folge für den Mitarbeiter. Eine solche Folge kann beispielsweise Lob durch den Vorgesetzten, verbesserte Karrierechancen oder Ähnliches sein. Die Kontingenz beschreibt mit welcher Sicherheit eine positive Evaluation auch eine positive Folge nach sich zieht. Wie sehr eine bestimmte Folge ein Bedürfnis befriedigt, wird über die Folge-Bedürfnis-Kontingenz ausgedrückt.

Jede Handlung wird somit durch ihre Kraft motiviert, die persönlichen Bedürfnisse der betrachteten Person zu befriedigen. Diese Kraft wiederum ergibt sich aus den Kontingenzen der Evaluationskette. Eine hohe Kontingenz führt zu einer hohen Motivationskraft. Dabei muss immer die ganze Kette betrachtet werden. Sind zwei Teile nur durch eine geringe Kontingenz verbunden, sinkt die Motivationskraft dadurch stark ab, da die Kette an diesem Punkt geschwächt ist. So eine Schwäche kann auch durch Stärken an anderen Stellen nur bedingt ausgeglichen werden.

Beispielsweise kann eine betrachtete Handlung mit hoher Wahrscheinlichkeit zu einem gewünschten Ergebnis führen, das vom Vorgesetzten sehr positiv bewertet wird. Diese Bewertung könnte wiederum mit einer hohen Wahrscheinlichkeit zu einem beruflichen Aufstieg in der Firma führen. Falls der betrachteten Person ihre Freizeit wichtiger ist als ihre Karriere, wird die betrachtete Handlung niedrig motiviert sein, wenn sich die Folge des beruflichen Aufstiegs negativ auf das Freizeitbedürfnis auswirkt.

Das VIE-Planungsmodell des Simulationsmodells basiert auf dem Modell von Vroom, das um einige Aspekte des Modells von Pritchard und Ashwood erweitert wurde. Ziel war es, einen Bewertungsprozess für alternative Handlungen zu entwickeln, der sich an den oben beschriebenen Modellen orientiert.

Sowohl Vroom, als auch Pritchard und Ashwood, beschreiben die Komponenten ihres Modells lediglich in einer abstrakten Form. Für das VIE-Modell müssen daher zunächst die einzelnen Elemente in Bezug auf den Anwendungsfall Softwareprojekt und das Simulationsmodell an sich konkretisiert werden. Das VIE-Modell fasst dabei die *Ergebnis-Evaluation-Kontingenz* und die *Evaluation-Folgen-Kontingenz* zur *Ergebnis-Folgen-Kontingenz* zusammen, und ignoriert damit den Schritt der Evaluation im Modell von Pritchard und Ashwood. Diese Vereinfachung ist sinnvoll, da das Simulationsmodell keine Evaluation von Arbeitsergebnissen durch einen Vorgesetzten vorsieht.

Weder das Modell von Vroom, noch das Modell von Pritchard und Ashwood berücksichtigen die intrinsische Motivation von Handlungen. Handlungen können jedoch auch aus sich selbst heraus motiviert sein, wenn es einem Mitarbeiter Spaß macht sie auszuführen. Um diesen Aspekt abzubilden, wird eine zusätzliche Kontingenz *intrinsische Motivation* zwischen Handlung und Bedürfnisse eingeführt. Sie entspricht der Valenz der Handlung an sich, unabhängig ihrer Ergebnisse und deren Folgen. Die Stärke der Valenz wird durch die Sympathie für die Aktivität festgelegt, die für jeden Projektbeteiligten, jeweils für jede Aktivität, und damit auch zu jeder Handlung im Szenario festgelegt ist. Um diese Kontingenz in das restliche Modell einzufügen, wird ein Pseudoergebnis und eine Pseudofolge für die intrinsische Motivation eingeführt, die jeweils über eine konstante maximale Kontingenz mit den restlichen Teilen des Modells verbunden ist.

Im Simulationsmodell entspricht eine Handlung einer Menge von Aufgaben. Eine Aufgabe besteht wiederum aus der durchzuführenden Aktivität, dem Artefakt, auf das diese Aktivität angewendet werden soll und der Personengruppe, welche die Aufgabe ausführt.

Das VIE-Modell ist auf Handlungen, bestehend aus einer einzelnen Aufgabe beschränkt. Sequenzen von Aufgaben werden nicht unterstützt. Ein Projektbeteiligter ist somit nicht in der Lage, aufeinander aufbauende Aufgaben zu planen, um zum Beispiel Bedingungen zu schaffen, die benötigt werden um eine bestimmte andere Aufgabe bearbeiten zu können. Eine dahingehende Erweiterung des VIE-Modells wäre eine wertvolle Ergänzung des Simulationsmodells, wird jedoch im Rahmen dieser Arbeit nicht behandelt.

Die Rolle des VIE-Modells ist es, zu jeder möglichen Aufgabe deren Motivationskraft zu berechnen, um so eine persönliche Rangfolge der Aufgaben für den betrachteten Mitarbeiter zu erstellen. Der Mitarbeiter wählt anschließend die Aufgabe mit der höchsten Motivationskraft als seine nächste Handlung aus, die er entweder bis zur Aktualisierung der Planung im nächsten Zeitschritt, oder bis zum Ende seines Arbeitstages ausführt. Gegebenenfalls wird die Aufgabe, wie im Abschnitt zur direkten Planung gezeigt, um spontane Kommunikation ergänzt.

Als Ausgangsmenge wird die Menge aller Aufgaben als das kartesische Produkt aus Aktivitäten, Artefakten und der Potenzmenge aller Projektbeteiligter gebildet. Diese Menge wird auf die für

den Mitarbeiter zum aktuellen Zeitpunkt durchführbaren Aufgaben beschränkt. Die Kriterien dafür entsprechen denen, die auch in den Tätigkeitsmodellen vor Beginn einer Aufgabe geprüft werden. Es werden alle Artefakte ignoriert, für die mindestens eine Vorbedingung nicht erfüllt ist, es werden alle Aktivitäten ignoriert, für die der betrachtete Mitarbeiter nicht die notwendige Rolle einnimmt. Es werden außerdem alle Elemente der Potenzmenge ignoriert, in denen der betrachtete Mitarbeiter nicht enthalten ist, oder die einen Mitarbeiter enthalten, der nicht am Arbeitsplatz ist. Außerdem wird berücksichtigt, dass Mitarbeiter die Teilnahme an einer Aufgabe ablehnen können, wenn sie dazu keine Zeit haben oder aus anderen Gründen nicht teilnehmen wollen. Die im Simulationsmodell vorhandenen Planungsmodelle berücksichtigen dazu lediglich den Zeitaspekt, indem sie prüfen, ob der betreffende Mitarbeiter das angefragte Zeitfenster schon für andere Aktivitäten verbraucht hat. Neue Planungsmodelle können die vorhandenen Modelle verwenden, um hier bei Bedarf auch einen anderen Ansatz zu realisieren.

Für jede der Aufgaben aus der so beschränkten Grundmenge wird anschließend die Motivationskraft ermittelt, indem die Kontingenzen entlang der von Pritchard und Ashwood beschriebenen Evaluationskette berechnet werden.

Die folgenden Abschnitte beschreiben die einzelnen Teile der Evaluationskette in ihrem Bezug auf das Simulationsmodell. Abschnitt 2.6.4 umfasst die Darstellung der Ergebnisse einer Handlung und die Handlungs-Ergebnis-Kontingenz. Abschnitt 2.6.5 beschreibt die Handlungsfolgen und die Ergebnis-Folgen-Kontingenz. Im Abschnitt 2.6.6 wird beschrieben, wie Bedürfnisse definiert sind, und wie diese über die Folgen-Bedürfnis-Kontingenz mit den Folgen zusammenhängen. Eine Beschreibung der einzelnen Schritte findet sich auch in [Nassal, 2016].

2.6.4. Ergebnisse und Handlungs-Ergebnis-Kontingenz im VIE-Modell

Um die nächste Handlung eines betrachteten Mitarbeiters zu selektieren, muss die Motivationskraft aller Handlungsalternativen T berechnet werden. Um eine Handlungsalternative $\tau \in \mathrm{T}$ zu bewerten, müssen im ersten Schritt die Ergebnisse betrachtet werden, die durch die Handlung erzielt werden können. Eine Handlung kann dabei im Allgemeinen zu unendlich vielen verschiedenen Ergebnissen führen, die sich jedoch in der Wahrscheinlichkeit, mit der sie aus der Handlung hervorgehen, unterscheiden.

Das Simulationsmodell interpretiert die Handlungs-Ergebnis-Kontingenz als die Wahrscheinlichkeit, mit der eine Handlung τ zu einem bestimmten Ergebnis r führt. Vroom nennt diese Wahrscheinlichkeit die *Erwartung* E. Zusammen mit der Valenz des Ergebnisses V lässt sich so die Motivationskraft einer Handlung berechnen.

$$\text{Motivationskraft}(\tau) = \sum_{r \in R} V(r) \cdot E(\tau, r) \tag{2.54}$$

Um die Valenz des Ergebnisses zu erhalten, muss die restliche Evaluationskette, von Ergebnis bis Bedürfnis, ausgewertet werden, was in den Abschnitten 2.6.5 und 2.6.6 beschrieben wird.

Um eine mathematisch optimale Lösung zu erhalten, müssten alle möglichen Ergebnisse und deren Evaluationsketten berücksichtigt werden. Da dies dem menschlichen Gehirn nicht möglich ist, wird in der Realität nur eine kleine Untermenge der möglichen Lösungen betrachtet, deren Wahrscheinlichkeit hoch genug ist, um relevant zu sein. Diese Selektion wird subjektiv von der jeweiligen Person, typischerweise unterbewusst getroffen. Das VIE-Modell adaptiert diesen Ansatz und reduziert die Berechnung auf das wahrscheinlichsten zu erwartende Ergebnis $\hat{r}$. Dadurch lässt sich die aufwändige Berechnung auf ein Minimum reduzieren, was zu einer höheren Performanz des gesamten Simulationsablaufs führt. Eine Gewichtung unterschiedlicher Ergebnisse anhand ihrer Wahrscheinlichkeit wird dadurch unnötig, und die Motivationskraft einer Handlung kann als die Valenz ihres wahrscheinlichsten Ergebnisses interpretiert werden.

$$\text{Motivationskraft}(\tau) = V(\hat{r}) \tag{2.55}$$

Ziel des ersten Evaluationsschritts ist es daher, das zu erwartende, und damit das wahrscheinlichste Ergebnis einer gegebenen Handlungsalternative zu berechnen. Jede Handlung hat das Ziel, den Zustand des Projekts zu verändern. Diese Veränderung stellt das Handlungsergebnis dar. Um die Handlungsergebnisse quantitativ darstellen zu können, definiert das VIE-Modell acht Kategorien,

für die jeweils ein einzelner Wert ermittelt wird. Diese Werte bilden den Ergebnisvektor $\vec{r} \in \mathbb{R}^8$, der das zu erwartende Ergebnis einer Handlung anhand dieser acht Kategorien darstellt. Dabei ist zu beachten, dass es sich dabei um die subjektive Einschätzung des Mitarbeiters im Rahmen der Handlungsplanung handelt. Das tatsächliche Ergebnis, das bei Wahl der Handlung durch das Simulationsmodell berechnet wird, kann davon beliebig abweichen.

- Mit **Arbeitsfortschritt** wird der aus der Handlung resultierende Fortschritt quantifiziert. Voraussetzung dafür ist, dass dieser objektiv oder subjektiv messbar ist. Der zu erwartende Arbeitsfortschritt wird vom betreffenden Mitarbeiter anhand der Aktivität und der damit verbundenen Arbeitsgeschwindigkeit geschätzt. Diese wird über das Modul für die Selbstwahrnehmung bezogen, und basiert damit auf den Erfahrungswerten des Mitarbeiters.

$$
\begin{aligned}
r_{\text{Fortschritt}} &= v \cdot (1 - t_{\Delta\text{Wechsel}}) + 0.5 \cdot \max(1 - (s - c), 0) \\
v &: \text{angenommene Arbeitsgeschwindigkeit} \\
t_{\Delta\text{Wechsel}} &: \text{benötigte Zeit für den Aufgabenwechsel} \\
s &: \text{angenommener Umfang des zu bearbeitenden Artefakts} \\
c &: \text{angenommene Vollständigkeit des zu bearbeitenden Artefakts}
\end{aligned}
\tag{2.56}
$$

  Formel 2.56 zeigt die Berechnung des durch die Handlung zu erwartenden Arbeitsfortschritts. Diese vom Mitarbeiter gemachte Schätzung kann unter Umständen erheblich vom Arbeitsfortschritt abweichen, der bei der tatsächlichen Durchführung der Handlung erzielt wird. Die Schätzung des Fortschritts setzt sich aus drei Aspekten zusammen: der erwartete Fortschritt in Arbeitseinheiten bezogen auf ein Arbeitsintervall von einer Stunde, der gegebenenfalls benötigten Zeit für den Aufgabenwechsel und einem Bonus für das Fertigstellen von Artefakten. Um die erzielten Arbeitseinheiten zu schätzen, wird die Arbeitsgeschwindigkeit mit dem Zeitintervall von 1h multipliziert. Die Zeit, die für einen Aufgabenwechsel veranschlagt wird, wird zuvor davon abgezogen. Sie wird direkt mit dem Modell für den Aufgabenwechsel berechnet. Dabei wird der Einfachheit halber auf eine Schätzung durch den Mitarbeiter verzichtet und der tatsächliche Wert verwendet. Da das Beenden eines Artefakts zusätzlich motiviert, ist hierfür ein Bonus vorgesehen, wenn weniger als eine Arbeitseinheit aussteht. Dies hilft zu verhindern, dass Artefakte nur beinahe beendet werden.

- Die **Qualitätsänderung** beschreibt die zu erwartende Veränderung des Qualitätsniveaus, des betroffenen Artefakts. Durch eine Aufgabe kann sich die Qualität eines Artefakts sowohl verbessern, als auch verschlechtern. Um abzuschätzen, wie eine Handlung die Qualität beeinflusst, werden die Fähigkeiten der daran beteiligten Mitarbeiter ausgewertet.

  Dazu werden deren Fähigkeiten über die Umgebungswahrnehmung, beziehungsweise die Selbstwahrnehmung geschätzt und davon, äquivalent zum Vorgehen im entsprechenden Elementarmodell, das Maximum der Gruppe ausgewählt. Außerdem wird mittels der Umgebungswahrnehmung die aktuelle Qualität des betroffenen Artefakts und sein Schwierigkeitsgrad geschätzt. Daraus lassen sich zwei Aspekte ableiten: Die Differenz zwischen Fähigkeit und Schwierigkeit liefert ein Indikator für die Eignung der Gruppe für die Aufgabe an sich. Die Differenz zwischen der Fähigkeit der Gruppe und der aktuellen Qualität des Artefakts lässt einen Rückschluss zu, ob sich die Qualität des betroffenen Artefakts durch die Handlung verbessern oder verschlechtern wird. Formel 2.57 zeigt, wie die Aspekte zusammengesetzt werden. Da der Basiswert für Fähigkeiten bei 0.5, der Basiswert für Qualität jedoch bei 0.7 liegt, wird diese Differenz durch +0.2 ausgeglichen. Da es sich hier nur um eine Abschätzung handeln soll, wird an dieser Stelle darauf verzichtet, Fähigkeit und Schwierigkeit mittels des entsprechenden Elementarmodells in den Qualitätswert umzurechnen. Ein direkter Vergleich von Fähigkeiten, Schwierigkeit und Qualität ist möglich, da alle Werte im selben Wertebereich liegen.

  Bei der Berechnung muss berücksichtigt werden, dass es Artefakte gibt, die noch nicht begonnen wurden, und für die deshalb auch keine Qualität geschätzt werden kann. In solchen Fällen

wird der zweite Aspekt ignoriert und nur der vordere Summand verwendet. Handelt es sich bei der Aktivität um eine Qualitätsaktivität, so wird $r_{\text{Qualität}}$ nach unten auf 0 beschränkt um negative Werte zu vermeiden. Das ist notwendig, da durch Qualitätsarbeit die Qualität nicht sinken kann.

$$
\begin{aligned}
& r_{\text{Qualität}} = (\hat{l} - \theta) + (\hat{l} - q + 0.2) \\
& \hat{l} : \text{angenommenes Fähigkeitsniveau der Bearbeitergruppe} \\
& \theta : \text{angenommene Schwierigkeit des Artefakts} \\
& q : \text{angenommene Qualität des Artefakts}
\end{aligned}
\tag{2.57}
$$

- Mit der **Einhaltung des Projektplans** wird evaluiert, wie gut die gewählte Aufgabe zu den im Projektplan vorgesehenen Aufgaben passt.

  Der Spieler, in seiner Rolle als Projektleiter, gibt seinen Mitarbeitern einen Projektplan vor, der beschreibt, wann welche Aufgaben von welchen Mitarbeitern durchzuführen sind. Nicht immer macht dessen Einhaltung aus Sicht des Mitarbeiters Sinn. Das kann sich zum einen auf das Projekt an sich beziehen, aber auch die persönlichen Ziele des betroffenen Mitarbeiters können den Entscheidungen im Projektplan entgegenstehen. Beispielsweise ist für das Projekt notwendig, dass eine bestimmte Aufgabe fertiggestellt wird, der Mitarbeiter möchte jedoch lieber eine Aufgabe bearbeiten, die für ihn interessanter ist.

  Ein Projektleiter erwartet von seinen Mitarbeitern, dass sie die über den Projektplan gemachten Vorgabe umsetzen. Um zu ermitteln, wie gut eine gewählte Aufgabe den Projektplan erfüllt, wird ein Relevanzmaß für Aufgaben in Bezug auf einen bestimmten Mitarbeiter definiert, das folgende Eigenschaften aufweist:

  - Aufgaben, die nicht im Projektplan sind haben eine Relevanz von 0.
  - Aufgaben an denen der betrachtete Mitarbeiter laut Projektplan beteiligt ist, haben eine höhere Relevanz als Aufgaben an denen er nicht beteiligt ist.
  - Je besser die personelle Belegung einer Aufgabe zum Projektplan passt, umso höher ist die Relevanz der Aufgabe.
  - Eine aktive Aufgabe, d. h. eine Aufgabe, die zum aktuellen Zeitpunkt laut Projektplan zu bearbeiten ist, hat eine höhere Relevanz als eine vergleichbare nicht aktive Aufgabe.
  - Eine zeitlich nähere Aufgabe hat eine höhere Relevanz als eine vergleichbare Aufgabe, die zeitlich weiter entfernt liegt. Der zeitliche Bezugspunkt für Aufgaben in der Vergangenheit ist deren festgelegtes Ende, bei Aufgaben in der Zukunft wird deren Startzeitpunkt als Bezugspunkt verwendet.
  - Je weniger Zeit pro offenem Aufgabenteil bleibt, umso höher ist die Relevanz der Aufgabe.
  - Eigene Aufgaben, die kurz vor ihrer Fertigstellung stehen haben eine höhere Relevanz als vergleichbare Aufgaben, für die das nicht zutrifft.

  Um ein solches Maß zu realisieren, werden zunächst alle Aufgaben im Projektplan gesucht, die in Artefakt und Aktivität mit der zu bewertenden Aufgabe übereinstimmen. Wird keine passende Aufgabe im Projektplan gefunden, ist $r_{\text{Projektplan}} = 0$. Ansonsten wird für jede der gefundenen Aufgaben eine Kenngröße ermittelt, die angibt, wie gut diese mit der zu bewertenden Aufgabe übereinstimmt. Dazu wird für jeden der oben genannten Bewertungsaspekte mittels der folgenden Formeln eine eigene Kenngröße ermittelt:

  x_{eigen} gibt an, ob der Mitarbeiter irgendwann im Projektplan für die Aufgabe vorgesehen ist. Dann gilt $x_{\text{eigen}} = 1$, sonst $x_{\text{eigen}} = -1$.

  $x_{\text{Mitarbeiter}}$ gibt an, wie gut die personelle Belegung B' zur für die Aufgabe vorgesehenen Belegung B passt. B beziehungsweise B' ist dabei die entsprechende Menge der Mitarbeiter.

$$
x_{\text{Mitarbeiter}} = \frac{|B \cap B'|}{|B|} \tag{2.58}
$$

x_{aktiv} beschreibt, ob die betrachtete Aufgabe aus dem Projektplan zum aktuellen Zeitpunkt t aktiv ist.

$$x_{\text{aktiv}} = \begin{cases} 1, & t_{\text{Aufgabenbeginn}} \leq t \leq t_{\text{Aufgabenende}} \\ 0, & \text{sonst} \end{cases} \tag{2.59}$$

x_{Zeit} beschreibt die zeitliche Distanz der Aufgabe aus dem Projektplan in Bezug auf den aktuellen Zeitpunkt t.

$$\begin{aligned} t_{\Delta\text{Distanz}} &= \begin{cases} t - t_{\text{Aufgabenende}}, & t > t_{\text{Aufgabenende}} \\ t - t_{\text{Aufgabenbeginn}}, & t < t_{\text{Aufgabenbeginn}} \\ 0, & \text{sonst} \end{cases} \\ x_{\text{Zeit}} &= 2^{-\frac{t_{\Delta\text{Distanz}}}{t_{\Delta\text{Aufgabendauer}}}} \end{aligned} \tag{2.60}$$

$x_{\text{Zeitdruck}}$ beschreibt den aktuellen Zeitdruck ρ der für die Aufgabe aus dem Projektplan gilt. Dazu wird die Zeitdruckfunktion P aus dem Elementarmodell wie in 2.5 beschrieben verwendet. Um zu extreme Auswirkungen zu vermeiden wird ρ auf maximal 1.5 beschränkt.

$x_{\text{Abschluss}}$ beschreibt, wie kurz eine Aufgabe vor ihrem Abschluss steht. Dabei werden Aufgaben berücksichtigt, für die ein Arbeitsaufwand von unter einem Tag angenommen wird, was 8 Arbeitseinheiten entspricht.

$$x_{\text{Abschluss}} = \max\left(1 - \frac{s-c}{8}, 0\right) \tag{2.61}$$

Die einzelnen Kenngrößen werden, wie in Formel 2.62 gezeigt, gewichtet und zur Relevanz der betrachteten Aufgabe des Projektplans kombiniert. Die Bewertung in Bezug auf den gesamten Projektplan ergibt sich aus dem Maximum der Relevanzwerte aller passenden Aufgaben des Projektplans.

$$\begin{aligned} \text{Relevanzwert} = \quad & 0.6 \cdot x_{\text{eigen}} + 0.3 \cdot x_{\text{Mitarbeiter}} + 0.3 \cdot x_{\text{aktiv}} + 0.3 \cdot x_{\text{Zeit}} \\ & + 0.5 \cdot x_{\text{Zeitdruck}} + 0.5 \cdot x_{\text{Abschluss}} - 1 \end{aligned} \tag{2.62}$$

Die Gewichte haben sich anhand unterschiedliche Testläufe verschiedener Szenarien ergeben. Bei ihrer Wahl muss nicht nur das Verhältnis zwischen den einzelnen Kenngrößen berücksichtigt werden, sondern auch die Größe des Ergebnisses im Vergleich zu den anderen Komponenten des Ergebnisvektors.

- Mit der **Verbesserung der eigenen Fähigkeiten** wird der Lernfortschritt bewertet, den der betrachtete Mitarbeiter durch die Ausführung der zu evaluierenden Handlung erwartet.

  Dazu wird zunächst das potentiell zu erwerbende Wissen als Differenz aus bekanntem Wissen und im Artefakt vorhandenen Wissen mittels der Umgebungs- und Selbstwahrnehmung geschätzt. Außerdem wird das Elementarmodell für den Fähigkeitszuwachs verwendet, um den Zuwachs der relevanten Fähigkeiten zu ermitteln. Die notwendigen Eingangsparameter werden mit Hilfe der Umgebungs- und Selbstwahrnehmung geschätzt. Dabei entsteht jeweils ein Wert für den erwarteten Fähigkeitszuwachs in Bezug auf die Aktivitäten und Werte für den Fähigkeitszuwachs der jeweils für die Aufgabe relevanten Disziplinen. Ist mehr als eine Disziplin relevant, werden die Ergebnisse der einzelnen Disziplinen zu einer gewichteten Summe zusammengefasst. Als Gewichte werden dabei die jeweiligen Anteile der Disziplinen an der Aufgabe verwendet, die das Szenario vorgibt.

  Die Gesamtbewertung ergibt sich, wie in Formel 2.63 gezeigt, aus der Kombination dieser drei Aspekte. Durch die verwendeten Faktoren werden die Ergebnisse so skaliert, dass sie in ihrer Größe zueinander und mit den anderen Werten des Ergebnisvektors vergleichbar sind. Die verwendete Wurzelfunktion führt dazu, dass kleinere Fähigkeitszuwachse einen überproportional

hohen Einfluss bekommen, und damit besser berücksichtigt werden.

$$r_{\text{Lernen}} = \sqrt{l_{\Delta+_a}} \cdot 50 + \sqrt{l_{\Delta+_D}} \cdot 50 + k_\Delta \cdot 0.3$$

$l_{\Delta+_a}$: angenommener Wert für den Fähigkeitszuwachs bezogen auf die Aktivität der Aufgabe

$l_{\Delta+_D}$: angenommener Wert für den Fähigkeitszuwachs bezogen auf die relevanten Disziplinen (gewichtete Summe)

k_Δ : potentiell zu erwerbendes Wissen (2.63)

- Die **Unproduktive Zeit** beschreibt, ob ein Mitarbeiter seine Arbeitszeit im Sinne seiner Aufgaben nutzt, oder für andere Tätigkeiten aufwendet.

  Prinzipiell kann ein Mitarbeiter die ihm zur Verfügung stehende Zeit in die ihm gestellten Aufgaben investieren, oder sie für andere Tätigkeiten nutzen, die nicht im Rahmen seiner Arbeit liegen. Zeit, die nicht in die Projektaufgaben investiert wird, wie beispielsweise Pausen, wird als unproduktive Zeit bezeichnet.

  Führt ein Mitarbeiter eine Aufgabe aus, so ist seine unproduktive Zeit $r_{\text{unproduktiv}} = 0$. Arbeitet der Mitarbeiter während seiner Arbeitszeit nichts, beträgt die unproduktive Zeit $r_{\text{unproduktiv}} = 1$. Dass ein Mitarbeiter zu einem bestimmten Zeitpunkt nichts arbeitet, kann aus verschiedenen Gründen vorkommen. Beispielsweise kann es sein, dass es für ihn keine durchführbare Aufgabe gibt oder er für die offenen Aufgaben nicht über die benötigten Fähigkeiten verfügt, und damit mehr Schaden als Nutzen verursachen würde. Auch mangelnde Motivation kann ein Grund dafür sein, dass ein Mitarbeiter lieber nichts tut, als eine entsprechende Aufgabe zu bearbeiten.

- **Verbreitetes Wissen** beschreibt das eigene Wissen, das im Rahmen einer Aufgabe an andere Mitarbeiter weitergegeben wird. Insbesondere bei Kommunikationsaufgaben besteht das Ziel in der Weitergabe von Wissen. Eine solche Weitergabe hat für den betroffenen Mitarbeiter keinen direkten Nutzen. Für die empfangenden Mitarbeiter kann das kommunizierte Wissen jedoch notwendig sein. Die Verbreitung von Wissen ist für den Projekterfolg in der Regel sehr wichtig.

  Um ein Indikator für das verbreitete Wissen zu berechnen, wird das Wissen der betrachteten Person mit dem Wissen des Rests der Gruppe verglichen. Je mehr exklusives Wissen der betrachtete Mitarbeiter über das zu bearbeitende Artefakt, bezogen auf die Gruppe der restlichen Bearbeiter besitzt, umso höher ist der zu erwartende Wissenstransfer.

  Der Indikator berechnet sich wie in Formel 2.64 gezeigt. Die notwendigen Werte, die das Wissen der einzelnen Personen über das Artefakt beschreiben, werden dabei über die Wahrnehmungskomponenten bezogen.

$$r_{\text{verbreitetes Wissen}} = \max\left(\left(K_0 - \bigcup_{i=1}^{|B|} K_i\right) \cdot |B|, 0\right)$$

K_0 : angenommenes Wissen des betrachteten Mitarbeiters

K_i : angenommenes Wissen der Kollegen

B : Menge der Kollegen (2.64)

- **Störung von Kollegen**. Benötigt ein Mitarbeiter zur Bewältigung einer Aufgabe die Unterstützung eines Kollegen, so stört er diesen bei der Ausführung dessen Tätigkeiten. Eine solche Störung kommt beispielsweise zustande, wenn Kommunikationsbedarf besteht oder die direkte Mitarbeit eines Kollegen an einer Aufgabe erforderlich oder gewünscht ist. Durch ständige Störung eines Kollegen, kann dessen Produktivität erheblich gesenkt werden, und muss deshalb entsprechend bei der Auswahl der Aufgabe berücksichtigt werden.

  Um zu ermitteln, ob ein Kollege durch eine Aufgabe gestört wird, oder ob sie zu seinem Aufgabenbereich gehört, werden zuerst alle Aufgaben T im Projektplan ermittelt, die in

Artefakt und Aktivität mit der zu planenden Aufgabe übereinstimmen, und nicht vollständig in der Vergangenheit liegen. Anschließend wird für jede dieser Aufgaben ein individueller Faktor berechnet, der die Störung quantifizieren würde, wenn der Projektplan nur aus dieser Aufgabe bestehen würde.

Dabei wird von der Annahme ausgegangen, dass ein Mitarbeiter gestört wird, wenn er zur Teilnahme an einer Aufgabe aufgefordert wird, für die er im Projektplan nicht vorgesehen ist. Er wird auch gestört, wenn die Aufgabe noch in ferner Zukunft liegt, und ihn daher momentan nicht interessiert. Je weiter die Aufgabe in der Zukunft liegt, umso höher ist der Störfaktor. Um diese Annahmen zu realisieren, wird für jeden Mitarbeiter die zeitlich nächste Aufgabe im Projektplan gesucht, die mit der zu evaluierenden Aufgabe hinsichtlich Artefakt und Aktivität übereinstimmt. Der Störfaktor ergibt sich aus der Summe der Zeitdifferenzen. Diese sind auf 30 Tage limitiert. Existiert für einen nachgefragten Mitarbeiter keine passende Aufgabe, wird dieses Maximum angenommen. Formel 2.65 zeigt die dazugehörige Berechnung. Der verwendete Skalierungsfaktor von 0.5 passt die Berechnung an die der anderen Elemente des Ergebnisvektors an.

$$
\begin{aligned}
&\mathrm{T}'_b = \{\tau' \in \mathrm{T} | b \in B_{\tau'} \wedge t_{\mathrm{Ende}_{\tau'}} \geq t\} \\
&r_{\mathrm{Störung}} = 0.5 \cdot \sum_{b \in B_0} \begin{cases} 1 & , \mathrm{T}'_b = \emptyset \\ \min\limits_{\tau \in \mathrm{T}'_b} \left(\min\left(\max\left(t_{\mathrm{Beginn}_\tau} - t, 0 \right), 30 \right) \cdot \dfrac{1}{30} \right) & , \text{sonst} \end{cases}
\end{aligned}
\tag{2.65}
$$

B_0 : Menge der für die geplante Aufgabe vorgesehener Kollegen
T : Menge Aufgaben im Projektplan
t : Zeitpunkt der Planung
$B_{\tau'}$: Menge der für die Aufgabe τ' vorgesehenen Mitarbeiter
$\mathrm{T}'_b \subset \mathrm{T}$: Menge der Aufgabe eines Mitarbeiters b aus dem Projektplan, die nicht in der Vergangenheit liegen
t_{Beginn_τ} : Zeitpunkt des vorgesehenen Beginns der Aufgabe τ

- Die **Intrinsische Motivation** beschreibt den Spaß bei der Arbeit, den ein Mitarbeiter von einer Aufgabe erwartet. Der Spaß resultiert dabei ausschließlich aus der Durchführung der Tätigkeit, und ist unabhängig von den dabei erzielten direkten und indirekten Ergebnissen.

  Wie viel Spaß einem betrachteten Mitarbeiter eine konkrete Aufgabe macht, hängt von der Aufgabe und dem Charakter des Mitarbeiters ab. Dieser beschreibt dazu jeweils einen Sympathie-/Antipathiewert ψ_a für die einzelnen Aktivitäten, und jeweils einen Sympathie-/Antipathiewert ψ_m für die im Projekt vorhandenen Kollegen. Diese Werte geben an, wie gerne ein Mitarbeiter eine bestimmte Aktivität durchführt, beziehungsweise wie gerne er mit einem bestimmten Kollegen zusammenarbeitet.

  Um die intrinsische Motivation zu berechnen, wird als Basiswert der Sympathie-/Antipathiewert der Aufgabenaktivität verwendet. Dieser wird durch die an der Aufgabe teilnehmenden Kollegen, anhand des jeweiligen Sympathie-/Antipathiewerts für den Kollegen, nach oben oder nach unten verändert. Da der neutrale Wert hierfür bei 0.5 liegt, muss ψ_m dazu entsprechend um -0.5 verschoben werden. Formel 2.66 zeigt die dazugehörige Berechnung. Die entsprechenden Faktoren dienen dazu, den berechneten Wert mit den anderen Werten des Ergebnisvektors vergleichbar zu machen.

$$
r_{\mathrm{intrinsisch}} = 0.8 \cdot \psi_a + 0.4 \cdot \sum_{b \in B} (\psi_{\mathrm{m}b} - 0.5)
\tag{2.66}
$$

ψ_a : Sympathie/Antipathie gegenüber der durchzuführenden Aktivität
B : Menge der an der Aufgabe beteiligten Kollegen
$\psi_{\mathrm{m}b}$: Sympathie/Antipathie gegenüber Kollegen b

Das im VIE-Modell umgesetzte und hier beschriebene Vorgehen zur Einschätzung der Handlungsergebnisse realisiert nur eine der vielen denkbaren Metriken. Die Wahl fiel auf die oben beschriebenen Formeln, da diese nicht zu komplex sind, und trotzdem brauchbare Ergebnisse liefern. Es ist problemlos möglich, diese in Form von Erweiterungen gegen andere Berechnungen auszutauschen. Dabei muss jedoch berücksichtigt werden, dass hier lediglich eine Abschätzung realisiert werden soll. Das Simulationsmodell oder Teile davon zu verwenden, um genaue Vorhersagen zu treffen, ist in den meisten vielen Fällen nicht sinnvoll. Da einige Faktoren nicht natürlich nach oben beschränkt sind, war es nicht möglich, einen eingeschränkten Wertebereich für die einzelnen Komponenten zu definieren. Entsprechend wichtig ist es, die einzelnen Komponenten des Ergebnisvektors durch die Definition der Metrik vergleichbar zu halten, da alle weiteren Berechnungen von der Annahme der Vergleichbarkeit ausgehen, und keine weitere Normierung stattfindet.

2.6.5. Folgen und Ergebnis-Folgen-Kontingenz im VIE-Modell

Im nächsten Schritt muss die Valenz des erwarteten Ergebnisses berechnet werden. Dazu wird das Ergebnis zunächst auf die zu erwartenden Folgen abgebildet. Die Ergebnis-Folgen-Kontingenz beschreibt hierbei die Wahrscheinlichkeit, dass ein Ergebnis zu einer Menge betrachteter Folgen führt. Den Zusammenhang zwischen dem Ergebnis einer Handlung $\vec{r}$ und einer bestimmten Folge o_i beschreibt Vroom mit der Instrumentalität $I(\vec{r}, o_i)$. Sie gibt an, wie sich das Ergebnis $\vec{r}$ auf die Folge o_i auswirkt. Bezogen auf das Modell von Pritchard und Ashwood bestimmt die Instrumentalität die Ergebnis-Folgen-Kontingenz. Zusammen mit der Valenz der Folge, deren Berechnung im nächsten Abschnitt behandelt wird, ergibt sich so die Valenz des Handlungsergebnisses.

$$V(\vec{r}) = \sum_{o_i \in O} V'(o_i) \cdot I(\vec{r}, o_i) \tag{2.67}$$

O ist die Menge aller betrachteten Folgen und $\vec{o} \in \mathbb{R}^7$ der dazu passende Bewertungsvektor, der zu jeder Folge einen entsprechenden Wert enthält. Nach Vroom ist die Instrumentalität das primäre Mittel, das ein Vorgesetzter hat, um die Motivation und das Verhalten seiner Mitarbeiter zu beeinflussen. Die anderen beiden Kontingenzen kann ein Vorgesetzter nicht, oder nur sehr schwer beeinflussen. Die Handlungs-Ergebnis-Kontingenz kann beispielsweise nur indirekt über das Schaffen bessere Rahmenbedingungen für die einzelnen Handlungen gelöst werden.

Jedes Ergebnis einer Handlung kann unterschiedliche Folgen nach sich ziehen. Um die möglichen Folgen zu beschreiben, wurden für das VIE-Modell sieben Kategorien definiert, die sich an den Ergebnissen der Pittsburgh-Studie [Herzberg, 1959, Neuberger, 1974] orientieren (vgl. 2.4.5). Der Folgenvektor $\vec{o}$ enthält für jede Kategorie ein Wert, der angibt, wie gut diese erfüllt ist. Je größer der Wert, umso besser ist die Kategorie erfüllt. Um die Werte zu erhalten, wird der Ergebnisvektor mittels der Instrumentalität auf den Folgenvektor abgebildet. Die einzelnen Kategorien sind im VIE-Modell wie folgt festgelegt:

- **Freiraum** beschreibt, wie viel Zeit ein Mitarbeiter während seiner Arbeitszeit zur freien Verfügung hat. In dieser Zeit kann der Mitarbeiter sich erholen oder Projekte und Ideen realisieren, die nicht direkt mit dem aktuellen Projekt zu tun haben. Dazu gehört beispielsweise Optimierungen des Arbeitsplatzes, aber auch Aktivitäten wie beispielsweise private Ganggespräche. Freiraum ist einer der Faktoren, die in der Pittsburgh-Studie direkt beschrieben werden.

  Im VIE-Modell resultiert Freiraum direkt aus der produktiven, beziehungsweise unproduktiven Zeit eines Mitarbeiters.

- **Beziehung zwischen Kollegen** beschreibt die zu erwartende Veränderung der sozialen Anerkennung und Beliebtheit der betrachteten Person aus Sicht ihrer Kollegen. Die Pittsburg-Studie unterscheidet hier zwischen Beziehung zum Vorgesetzten, Beziehung zu Kollegen und Beziehung zu Untergebenen. Da im Simulationsmodell keine Mitarbeiterhierarchien abgebildet sind, und der einzige Vorgesetzte durch den Spieler repräsentiert wird, beschränkt sich dieser Aspekt hier auf die Beziehung zwischen den Kollegen.

  Die wichtigste Quelle für die Verbesserung kollegialen Beziehung ist die Verbreitung von Wissen, da dies zum einen die Fähigkeiten der Person demonstriert, zum anderen als Hilfe für die

betroffenen Kollegen gewertet wird. Zusätzlich wirkt sich fehlendes Engagement negativ auf das kollegiale Verhältnis aus.

Das Simulationsmodell bildet keine soziale Interaktion zwischen Mitarbeitern ab, und beschränkt die Beziehung zwischen Mitarbeitern auf die im Charakter der Mitarbeiter angegebenen Werte für Sympathie/Antipathie, die über den Simulationsverlauf hinweg durch das Simulationsmodell nicht verändert werden.

- **Anerkennung** beschreibt die Bewertung und Wertschätzung der eigenen Arbeitsleistung durch die Kollegen.

  Anerkennung erlangt ein Mitarbeiter typischerweise durch die von ihm erbrachte Leistung, die sich sowohl in der Quantität, als auch in der Qualität seiner Arbeitsergebnisse widerspiegelt. Außerdem führt die Verbreitung von Wissen zu Anerkennung, da der Mitarbeiter dadurch seine Fähigkeiten und sein Wissen demonstriert. Bittet ein Mitarbeiter seine Kollegen zu häufig um Hilfe, wird dies negativ bewertet, weil er dadurch Unfähigkeit demonstriert. Auch zu viel unproduktive Zeit wirkt sich negativ auf die Anerkennung aus.

- **Beförderung und Lohn** beschreibt die Veränderung der finanziellen Vergütung und des eigenen Status im Unternehmen. Dieser kann beispielsweise durch einen Aufstieg im Unternehmen, typischerweise auch verbunden mit einer Gehaltserhöhung, oder durch einmalige Sonderzahlungen, Boni oder Gewinnbeteiligungen entstehen.

  Die Pittsburgh-Studie unterscheidet zwischen Lohn und Beförderung. Lohn bezieht sich auf die rein finanziellen Aspekte, Beförderung beinhaltet den Status innerhalb des Unternehmens. Das VIE-Modell fasst beide Aspekte zusammen, da eine Beförderung in den allermeisten Fällen mit einer Lohnerhöhung einhergeht. Da das Simulationsmodell keine Mitarbeiterhierarchie abbildet, werden die anderen Aspekte des Status ignoriert.

  Beförderung und Lohn ergibt sich im VIE-Modell als Folge der Arbeitsergebnisse anhand ihrer Quantität und Qualität, der Disziplin des Mitarbeiters, die sich in der Einhaltung des Projektplans zeigt und seiner allgemeinen Fähigkeiten. Verbessert ein Mitarbeiter seine Fähigkeiten, so steigen dadurch auch seine Chancen auf eine bessere Position im Unternehmen. Fehlendes Engagement, das sich durch hohe unproduktive Zeiten zeigt, wirkt sich hingegen negativ aus.

- **Sicherheit der Arbeitsstelle** beschreibt den subjektiven Eindruck eines Mitarbeiters in Bezug auf die Wahrscheinlichkeit entlassen zu werden. Neben diversen unternehmenspolitischen Faktoren, auf die ein Mitarbeiter in der Regel keinen Einfluss hat, und die deshalb im VIE-Modell ignoriert werden, ist die eigene Leistung die Hauptquelle für diese Einschätzung.

  Sicherheit ist die direkte Folge der Arbeit nach Plan. Je besser ein Mitarbeiter sich an seine Vorgaben hält, und je schneller und besser er arbeitet, umso geringer ist die Wahrscheinlichkeit einer Entlassung. Da ein gut ausgebildeter und erfahrener Mitarbeiter für das Unternehmen wertvoll ist, hat die Verbesserung der eigenen Fähigkeiten eine positive Auswirkung auf die Sicherheit der Arbeitsstelle. Das vorhandene Wissen zeigt sich unter anderem durch dessen Verbreitung, die somit ebenfalls einen positiven Einfluss hat. Unproduktive Zeit und das ständige Nachfragen bei Kollegen hingegen beeinflussen die Sicherheit der Arbeitsstelle negativ.

- **Wachstum** beschreibt die persönliche Weiterentwicklung einer Person. Dazu gehören insbesondere der Erwerb neuer, oder die Verbesserung vorhandener Fähigkeiten, sowie der Erwerb neuen Wissens.

  Das VIE-Modell beschreibt Wachstum als eine direkte Folge der Verbesserung der eigenen Fähigkeiten, wie sie im Ergebnisvektor abgebildet wird.

- **Intrinsische Motivation** beschreibt die erfahrene Befriedigung durch die Arbeit selbst, unabhängig von den weiteren Folgen der Handlung. Sie wird in der Pittsburgh-Studie als relevanter Faktor genannt. Die Folge einer Handlung beschreibt demnach die Befriedigung die aus der Handlung entsteht. Diese entspricht genau dem im Handlungsergebnis formulierten Wert für die intrinsische Motivation.

Das VIE-Modell realisiert die Instrumentalisierung als lineare Abbildung des Ergebnisvektors $\vec{r}$ auf den Folgenvektor $\vec{o}$, die sich an den bei der Beschreibung der Folgen angegebenen Zusammenhängen orientiert. Diese Abbildung kann, als folgende Matrix dargestellt werden:

$$
\begin{array}{l}
 \\
\textit{intr. Mot.} \\
\textit{unpr. Zeit} \\
\textit{verbr. Wissen} \\
\textit{Fortschritt} \\
\textit{Störung} \\
\textit{Qualität} \\
\textit{Projektplan} \\
\textit{Fähigkeiten}
\end{array}
\begin{array}{c}
\begin{array}{ccccccc}
\textit{intr. Mot.} & \textit{Freir.} & \textit{Beziehung} & \textit{Anerk.} & \textit{Lohn} & \textit{Sicherheit} & \textit{Wachstum}
\end{array} \\
\left(\begin{array}{ccccccc}
1.0 & & & & & & \\
 & 1.0 & -0.2 & -0.2 & -0.3 & -0.4 & -0.2 \\
 & & 0.45 & 0.45 & & 0.1 & \\
 & & & 0.33 & 0.33 & 0.33 & \\
 & & & -0.7 & -0.2 & -0.3 & \\
 & & & 0.5 & 0.3 & 0.3 & \\
 & & & 0.25 & 0.5 & 0.5 & \\
 & & & 0.1 & 0.1 & 0.1 & 1.0
\end{array}\right)
\end{array}
\tag{2.68}
$$

Bei der konkreten Realisierung eines Planspiels, das auf dem Simulationsmodell aufbaut, kann die Instrumentalisierung entsprechend verändert oder ausgetauscht werden, um spezifische Effekte zu realisieren. So könnte dem Spieler beispielsweise die Möglichkeit angeboten werden Boni zu vergeben, wenn Mitarbeiter sich weiterbilden oder besonders gute Qualität liefern. Entscheidet der Spieler sich für solche Boni, muss die Instrumentalisierung so angepasst werden, dass Qualität und Fähigkeiten auf die Folge *Lohn* entsprechend abgebildet werden. Dazu können entweder die entsprechenden Werte in der oben gezeigten Matrix erhöht werden, es kann jedoch auch eine vollständig neue Implementierung der Instrumentalisierung entwickelt werden, welche die gewünschten Aspekte umsetzt. Dass solche Boni tatsächlich ausgezahlt werden und damit das Budget des Projekts beeinflussen, ist nicht Teil des Verhaltensmodells und muss an anderer Stelle im Planspiel, beispielsweise durch einen weiteren Agenten, der dieses Verhalten realisiert, umgesetzt werden.

2.6.6. Bedürfnisse und Folgen-Bedürfnis-Kontingenz im VIE-Modell

Die letzte noch fehlende Komponente ist die Berechnung der Valenz $V'(o_i)$ einer Folge o_i. Sie beschreibt, wie wünschenswert diese Folge für die betrachtete Person ist. Grundlage dafür sind die Bedürfnisse der Person, die im Rahmen des Szenarios, in Form des Bedürfnisvektors $\vec{n} \in \mathbb{R}^6$ festgelegt werden.

Jeder Mensch verfügt über eine Menge von Bedürfnissen, die er versucht möglichst gut zu befriedigen. Dabei sind ihm nicht alle Bedürfnisse gleich wichtig, was einer entsprechenden Priorisierung entspricht. Er muss in vielen Situationen entscheiden, welche Bedürfnisse er bevorzugt befriedigt, und welche hintenangestellt werden. Das VIE-Modell definiert sechs Bedürfnisse, die auf der Maslowschen Bedürfnishierarchie [Maslow, 1943] basieren. Dabei wurden nur die für den Arbeitsalltag relevanten Bedürfnisse berücksichtigt. Körperliche Stärke, Liebe oder Religion sind hier beispielsweise nicht relevant. Außerdem wird davon ausgegangen, dass die Grundbedürfnisse, wie Essen, Trinken und Schlafen unabhängig von der Arbeitssituation und des Verhaltens des betrachteten Mitarbeiters erfüllt, und auf Dauer sichergestellt sind. Folgende Bedürfnisse sind im VIE-Modell definiert:

- **Spaß** beschreibt, wie wichtig es einer Person ist, Spaß am Leben zu haben und es zu genießen. Neben vielen Faktoren die außerhalb der Arbeit liegen, kann dieses Bedürfnis durch die Arbeit selbst, und durch die Bedingungen beeinflusst werden, unter denen die Arbeit stattfindet.

- **Freizeit und Gesundheit** beschreibt das Verlangen nach Erholung, körperlichem Wohlbefinden und freier Zeit, die unabhängig von der Arbeit ausgestaltet werden kann. Die Erfüllung steigt mit dem Freiraum, und fällt mit Beförderung und Lohn, da davon ausgegangen wird, dass dies mit mehr Arbeit und damit weniger Freizeit in Verbindung steht.

- **Status** beschreibt die Anerkennung durch das allgemeine soziale Umfeld, und damit, wie wichtig einer Person die Meinung ihres direkten und indirekten Umfelds über sie ist. Die Anerkennung bezieht sich dabei sowohl auf die Arbeitskollegen, als auch auf Bekannte, Verwandte, Freunde und sogar die allgemeine Öffentlichkeit. Das Bedürfnis wird im VIE-Modell durch eine gute Beziehung zu den Kollegen, Anerkennung durch die Kollegen und einer entsprechenden Position im Unternehmen befriedigt.

- Das Bedürfnis nach **Macht, Einfluss und Verantwortung** wird im Wesentlichen durch den beruflichen Status befriedigt, den eine Person innehat, und ist somit direkt von Beförderung und Lohn abhängig.
- **Finanzen und Sicherheit** beschreibt das Verlangen nach Reichtum, finanzieller Unabhängigkeit und der damit einhergehenden Sicherheit, für sich und andere sorgen zu können. Es wird hauptsächlich durch einen entsprechenden beruflichen Status, ein hohes Gehalt und eine sichere Arbeitsstelle befriedigt.
- Die **Persönliche Weiterentwicklung** beschreibt den Wissensdurst, der durch neues Wissen und neuen Fähigkeiten befriedigt werden kann. Dieses Bedürfnis wird im VIE-Modell daher hauptsächlich durch das persönliche Wachstum einer Person befriedigt.

Um die Valenz $V'(\vec{o})$ des Folgenvektors zu berechnen, wird der Bedürfnisvektor $\vec{n}$ mittels einer linearen Abbildung T auf einen zu den Folgen passenden Valenzvektor abgebildet, dessen Komponenten damit $V'(o_i)$ entsprechen. Da Folgen spielspezifisch sein können, und bei der Entwicklung eines Planspiels gegebenenfalls angepasst werden müssen, wird dieser letzte Schritt der Evaluationskette nach Pritchard und Ashwood dazu verwendet, eine planspielunabhängige Bedürfnisrepräsentation zu ermöglichen. Diese wiederum erlaubt es Charakterdefinitionen über verschiedene Szenarien und Planspiele hinweg zu verwenden und zu vergleichen. Die Abbildung ist im VIE-Modell wie folgt definiert:

$$V'(o_i) = \vec{T_i} \cdot \vec{n} \tag{2.69}$$

T entspricht dabei der folgenden Matrix, $\vec{T_i}$ enthält die $i$te Zeile:

$$\begin{array}{l} \\ \textit{Spaß} \\ \textit{Freizeit} \\ \textit{Status} \\ \textit{Macht u. V.} \\ \textit{Finanzen} \\ \textit{Weiterentw.} \end{array} \begin{array}{c} \begin{array}{ccccccc} \textit{intr. Mot.} & \textit{Freir.} & \textit{Beziehung} & \textit{Anerk.} & \textit{Lohn} & \textit{Sicherheit} & \textit{Wachstum} \end{array} \\ \begin{pmatrix} 1.0 & 0.3 & & & & & \\ & 1.1 & & & -0.5 & & \\ & & 0.4 & 0.4 & 0.2 & & \\ & & & & 1.0 & & \\ & & & & 0.34 & 0.66 & \\ & & & & & & 1.0 \end{pmatrix} \end{array} \tag{2.70}$$

Sowohl die Instrumentalisierungsfunktion, als auch die Abbildung des Bedürfnisvektors auf den Valenzvektor der Folgen sind im Rahmen dieser Arbeit frei definiert worden. Die konkreten Werte haben sich anhand mehrerer unterschiedlicher Testläufe im Rahmen verschiedener Szenarien ergeben. Dabei wurden die oben bei der Beschreibung der einzelnen Kategorien genannten Einflüsse berücksichtigt.

Die Evaluationskette des VIE-Modells bildet ein konkretes projektspezifisches Ergebnis auf die allgemeinen, arbeitsunabhängigen Bedürfnisse einer Person ab. Mit der Bewertung der Folgen wird dabei zunächst ein Zwischenschritt gemacht, der das Ergebnis auf die allgemeinen Auswirkungen bezogen auf die Arbeitsdomäne abbildet. Dieses Ergebnis wird anschließend auf die allgemeinen Bedürfnisse abgebildet. Prinzipiell lässt sich das erwartete Handlungsergebnis auch direkt auf die allgemeinen Bedürfnisse abbilden. Durch den Zwischenschritt der Folgen wird die Abbildung jedoch einfach nachvollziehbar, und ist damit auch leichter anpassbar.

Die Kategorien des Ergebnisvektors und der Folgen können bei Bedarf beliebig angepasst werden. Wird eine neue Ergebniskategorie eingeführt, muss dafür eine entsprechende Schätzfunktion definiert werden, mit der ein entsprechender Erwartungswert bezogen auf diese Kategorie für jede mögliche Handlung berechnet werden kann. Wird die Menge der Folgen verändert, müssen entsprechend die Instrumentalisierung und die Abbildung der Bedürfnisse auf den Valenzvektor der Folgen angepasst werden. Auch der Bedürfnisvektor kann bei Bedarf angepasst werden. Im Rahmen dieser Arbeit ist es nicht gelungen, eine Norm für die Werte, die durch die Evaluationskette fließen zu definieren, welche beispielsweise die Komponenten der Vektoren auf einen Wertebereich von $[-1, +1]$ beschränkt. Deswegen muss sowohl bei der Berechnung der Erwartungswerte, als auch bei der Definition der Abbildungen besonders darauf geachtet werden, dass die einzelnen Kategorien miteinander vergleichbar bleiben. Das bedeutet, dass die sich darauf beziehenden Werte eine gemeinsame Größenordnung

aufweisen, und die Abbildungen keine Verzerrung verursachen, die einzelne Kategorien ungerechtfertigt bevorzugt oder benachteiligt.

2.6.7. Beispielkonfigurationen des VIE-Modells

In diesem Abschnitt werden drei Beispielkonfigurationen des VIE-Modell beschrieben und das daraus resultierende Verhalten erläutert. Dazu werden verschiedene Charakterprofile definiert, die sich lediglich anhand ihres Bedürfnisprofils unterscheiden [Nassal, 2016].

Um eine Vergleichbarkeit der unterschiedlichen Profile zu ermöglichen, werden die Rahmenbedingungen für alle im Folgenden beschriebenen Szenarien identisch gewählt. Typischerweise gibt es eine Korrelation zwischen den Bedürfnissen und den anderen Eigenschaften einer Person. So kann man bei einem Mitarbeiter, der ein hohes Bedürfnis nach persönlicher Weiterentwicklung hat, auch eine hohe Lernfähigkeit erwarten. Um die Vergleichbarkeit nicht zu gefährden, werden jedoch auch die Charaktereigenschaften der betrachteten Mitarbeiter, mit Ausnahme ihres Bedürfnisvektors identisch gewählt. Abbildung 2.42 zeigt die Struktur des verwendeten Beispielszenarios.

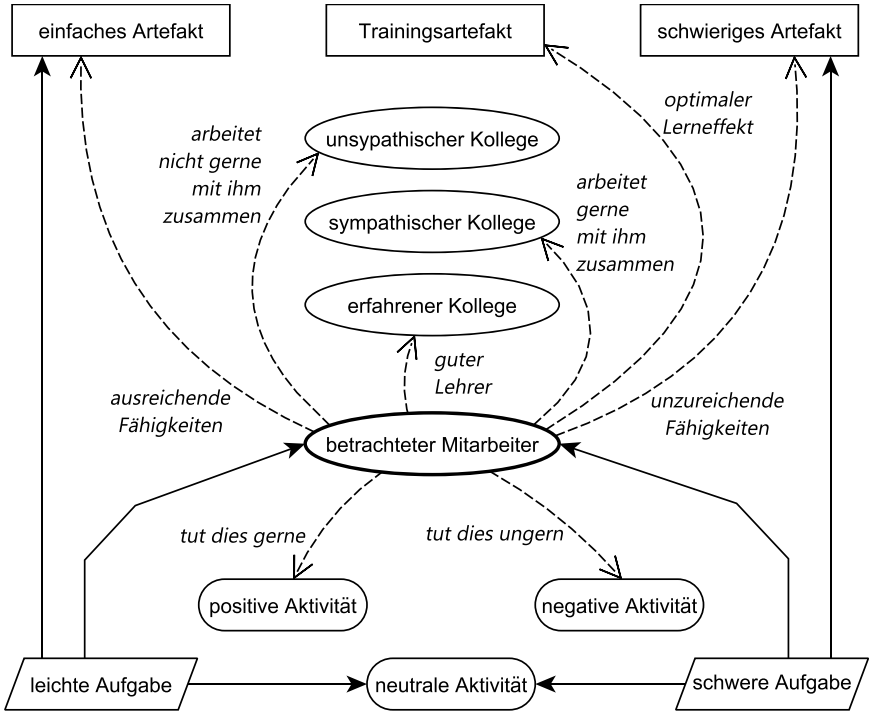


Abbildung 2.42.: Rahmenszenario für VIE-Beispiele (Erstveröffentlichung in [Nassal, 2016])

Das Szenario enthält drei Fortschrittsaktivitäten, die mit *neutral*, *positiv* und *negativ* bezeichnet sind. Sie unterscheiden sich ausschließlich in der Sympathie/Antipathie, die ihnen der Mitarbeiter entgegenbringt. Die positive Aktivität ($\psi_a = 1.0$) führt er gerne durch, die negative Aktivität ($\psi_a = 0.0$) vermeidet er wenn möglich, die neutrale Aktivität ($\psi_a = 0.5$) führt er weder besonders gern noch besonders ungern durch. Jede der Aktivitäten kann auf beliebigen Artefakten, und von allen im Szenario definierten Mitarbeitern durchgeführt werden.

Im Szenario sind drei voneinander unabhängige Artefakte definiert, die sich ausschließlich in ihrem Schwierigkeitsgrad unterscheiden. Dieser bezieht sich auf eine durchschnittliche Fähigkeit von $l = 0.5$, über die der betrachtete Mitarbeiter verfügt. Es existiert ein *einfaches Artefakt* ($\theta = 0.0$), das der betrachtete Mitarbeiter problemlos bearbeiten, und bei dem er mit einer hohen Arbeitsgeschwindigkeit und guter Qualität des Arbeitsergebnisses rechnen kann. Konträr dazu ist ein *schwieriges Artefakt* ($\theta = 0.9$) definiert, für das der betrachtete Mitarbeiter keine ausreichenden Fähigkeiten besitzt, und bei dem er entsprechend mit geringem Fortschritt und schlechter Ergebnisqualität rechnen muss. Ein drittes Artefakt ist in Form eines *Trainingsartefakts* ($\theta = 0.65$) definiert, das mit seinem

Schwierigkeitsgrad für einen optimalen Lerneffekt beim betrachteten Mitarbeiter sorgt, da es ihn fordert, jedoch nicht überfordert. Im Gegensatz dazu, kann der Mitarbeiter bei den anderen beiden Artefakten nicht mit einem sonderlichen Lerneffekt rechnen, da das einfache Artefakt keine neuen Erkenntnisse bringt, und das schwierige Artefakt ihn überfordert.

Der betrachtete Mitarbeiter entspricht einem durchschnittlichen Mitarbeiter. Das bedeutet, dass alle seine Charakterwerte und weitere Eigenschaften den jeweils definierten Standardwerten entsprechen (siehe hierzu Anhang A.1). Insbesondere die für das Szenario relevanten Fähigkeiten sind mit dem dafür vorgesehenen Wert für eine durchschnittliche Fähigkeit ($l = 0.5$) belegt. Der betrachtete Mitarbeiter ist Teil eines Teams, das drei weitere Mitarbeiter umfasst. Das Szenario definiert einen unsympathischen und einen sympathischen Kollegen, die in ihren Eigenschaften einem durchschnittlichen Mitarbeiter entsprechen. Mit dem unsympathischen Kollegen arbeitet der betrachtete Mitarbeiter nicht gerne zusammen, was durch einen Sympathie-/Antipathiewert von $\psi_{\mathrm{m}} = 0.0$ realisiert wird. Das Gegenstück dazu bildet der sympathische Kollege ($\psi_{\mathrm{m}} = 1.0$), mit dem der betrachtete Mitarbeiter gerne zusammenarbeitet. Dem dritten Kollegen gegenüber, ist die Einstellung des betrachtenden Mitarbeiters neutral ($\psi_{\mathrm{m}} = 0.5$). Im Gegensatz zu den anderen beiden Kollegen, besitzt er für die neutrale Aktivität die maximalen Fähigkeiten ($l = 1.0$). Dieser *erfahrene Kollege* ist damit in der Lage, schnelle und gute Ergebnisse bei Aufgaben, welche die neutrale Aktivität beinhalten, zu liefern, und den anderen Teammitgliedern bei solchen Aufgaben helfen zu können.

Der im Szenario enthaltene Projektplan enthält zwei voneinander unabhängige Aufgaben, die dem betrachteten Mitarbeiter zugewiesen sind. Bei der *leichten Aufgabe* soll der Mitarbeiter alleine das einfache Artefakt mit der neutralen Aktivität bearbeiten. Äquivalent dazu soll er bei der *schweren Aufgabe* alleine das schwierige Artefakt mit der neutralen Aktivität bearbeiten. Beide Aufgaben haben die gleiche Priorität. Sie sind als die Anweisung des Projektleiters an den betrachteten Mitarbeiter zu sehen, jedoch nicht dessen einzigen Handlungsmöglichkeiten. Er kann beispielsweise auch das Trainingsartefakt bearbeiten, oder Kollegen bitten, ihm bei seinen Aufgaben zu helfen. Er hat auch die Möglichkeit die Artefakte mit einer anderen Aktivität zu bearbeiten.

Um zu zeigen, wie sich unterschiedliche Bedürfniskonfigurationen auf das Verhalten auswirken, werden im Folgenden unterschiedliche Bedürfnisvektoren definiert, und betrachtet, welche Rangfolge der möglichen Handlungsoptionen sich daraus ergeben. Die Rangfolge wird dabei anhand der Motivationskraft durch das VIE-Modell berechnet. Die Ergebnisse unterscheiden sich in einigen Punkten von den in [Nassal, 2016] angegebenen Werten, da zwischenzeitlich einige Anpassungen an den Modellen stattgefunden haben.

Berufseinsteiger

Der Berufseinsteiger hat gerade im Team angefangen. Er kommt direkt von der Hochschule und verfügt entsprechend über viel theoretisches Wissen, jedoch wenig Praxiserfahrung. Er kennt sich außerdem mit dem Projekt in das er kommt noch nicht aus. Der Berufseinsteiger ist alleinstehend und hat keine Kinder. Er lebt sparsam und braucht daher wenig Geld. Da er noch nicht über entsprechende Erfahrung verfügt, verhält er sich unsicher, ist jedoch engagiert sich zu verbessern.

Auf Basis dieses Profils lässt sich folgender Bedürfnisvektor definieren:

$$\vec{n} = \begin{pmatrix} 0.2 \\ 0.3 \\ 0.7 \\ 0.4 \\ 0.3 \\ 0.9 \end{pmatrix} \begin{matrix} \textit{Spaß} \\ \textit{Freizeit und Gesundheit} \\ \textit{Status} \\ \textit{Macht, Einfluss, und Verantwortung} \\ \textit{Finanzen und Sicherheit} \\ \textit{persönliche Weiterentwicklung} \end{matrix} \tag{2.71}$$

Der Schwerpunkt des Berufseinsteigers liegt damit auf der persönlichen Weiterentwicklung; er möchte sich möglichst schnell einarbeiten und auf das Niveau seiner Kollegen kommen. Ein weiterer Schwerpunkt liegt auf der Anerkennung durch sein soziales Umfeld, da er bei seinen Kollegen einen positiven Eindruck hinterlassen und als gleichwertiges Teammitglied anerkannt werden möchte.

Im Rahmen des oben beschriebenen Szenarios berechnet das VIE-Modell auf Basis des Bedürfnisvektors die folgende Rangfolge für die möglichen Handlungsalternativen:

	Aktivität	Artefakt	beteiligte Kollegen	Motivationskraft
1.	Neutral	Einfach	—	0.358
2.	Neutral	Training	erfahrener Kollege	0.297
3.	Neutral	Einfach	sympathischer Kollege	0.287
4.	Positiv	Einfach	—	0.286
5.	Positiv	Training	—	0.273
6.	Neutral	Einfach	erfahrener Kollege	0.273
7.	Neutral	Einfach	unsympathischer Kollege	0.259
	...	...	...	...
29.	Neutral	Schwierig	erfahrener Kollege	0.157
	...	...	...	...
64.	Nichts tun	—	—	-0.046
	...	...	...	...
73.	Negativ	Schwierig	erfahrener K., unsymp. K.	-0.173

In der persönlichen Priorisierung der Handlungsalternativen sind die oben genannten Schwerpunkte wiederzuerkennen. Die Handlung mit der höchsten Motivationskraft wird vom Mitarbeiter als seine nächste Aufgabe selektiert. Er führt damit die im Projektplan als *einfache Aufgabe* beschriebene Aufgabe durch, indem er die *neutrale Aktivität* auf das *einfache Artefakt* anwendet. Er führt die Aufgabe, wie im Projektplan vorgesehen, alleine aus. Besonders für einen Neuling ist es eine gute Idee, sich an die Vorgaben zu halten, um Vorgesetzte und Team zufriedenzustellen. Es ist auch nachvollziehbar, dass er die leichte Aufgabe der schweren Aufgabe vorzieht, da er hier mit besseren Ergebnissen rechnen kann, und so nicht schon zu Beginn seiner Karriere Probleme im Projekt verursacht.

Auch die Handlungsalternative, die einen hohen Lernerfolg verspricht, weist eine hohe Motivationskraft auf. Hier überwiegen, unter Berücksichtigung des Hintergrunds sich schnell einzuarbeiten, der Vorteil, dass das Artefakt *Training* einen hohen Erfahrungsgewinn verspricht dem Nachteil, dass im Projektplan andere Aufgaben für den Einsteiger vorgesehen sind. Um seinen Lernfortschritt zu optimieren, wird der *erfahrene Kollege* hinzugezogen, der bei der Einarbeitung helfen kann. Dabei nimmt der Einsteiger den Nachteil in Kauf, dass er diesen Kollegen von dessen Arbeit abhält. Die gemeinsame Arbeit mit dem erfahrenen Kollegen hat außerdem den Vorteil, dass die zu erwartenden Arbeitsergebnisse deutlich besser sind, als wenn der Einsteiger die Aufgabe alleine durchführen würde. Eine Grundvoraussetzung dazu ist, dass der gewünschte Kollege auch bereit ist zu helfen. Das wird im Rahmen des Aufbaus der Handlungsalternativen geprüft, indem beim entsprechenden Kollegen nachgefragt wird. Das Simulationsmodell beschränkt sich derzeit darauf, an dieser Stelle die verfügbare Zeit des Kollegen zu prüfen, um auszuschließen, dass er diese im aktuellen Zeitschritt schon aufgebraucht hat.

Auch bei den weiteren Handlungsalternativen lässt sich beobachten, dass diese entsprechend der oben beschriebenen Schwerpunkte geordnet sind. Die erste Aufgabe, die das schwierige Artefakt behandelt ist auf Platz 15 der 73 Alternativen zu finden. Aufgaben, die das *schwere Artefakt* enthalten, haben eine niedrige Motivationskraft, da hier weder ein Arbeitserfolg, noch ein Lernerfolg zu erwarten ist. Man sieht außerdem, dass Aufgaben an denen der unsympathische Kollege beteiligt ist, eine niedrige Motivationskraft aufweisen als ihr Pendant mit dem sympathischen Kollegen. Hier werden die anderen beiden Kollegen bevorzugt. Auch *Nichts tun* ist eine Alternative, die zwar sehr weit unten in der Rangfolge steht, jedoch nicht auf dem letzten Platz zu finden ist.

Zusammengefasst kann festgestellt werden, dass das typische Verhalten eines neuen Mitarbeiters ohne viel Erfahrung adäquat abgebildet wird. Einfache Aufgaben versucht er möglichst exakt nach Vorgabe zu erledigen, bei schweren Aufgaben ist er sehr zurückhaltend. Während seiner Einarbeitungszeit konzentriert er sich oft mehr auf den Erwerb von Wissen und die Einarbeitung als auf produktive Arbeit. Wenn es ihm entsprechend von Nutzen ist, greift er auf den Rat und die Hilfe von Kollegen zurück, die ihm den Einstieg erleichtern und ihn mit dem nötigen Wissen versorgen.

Das hier beschrieben Verhalten ist nur ein mögliches Beispiel für einen Einsteiger. Diverse Veränderungen am Bedürfnisvektor oder am Rahmenszenario führt gegebenenfalls zu einem anderen

Verhalten. Dieses Verhalten ist jedoch immer in sich konsistent. Der Spieler hat so die Möglichkeit, seine Mitarbeiter kennen und einschätzen zu lernen. Dieses Wissen kann er wiederum einsetzen, um sein Projekt besser zu steuern. Das Verhalten eines Mitarbeiters kann sich im Verlauf eines Projekts ändern. Das passiert entweder, wenn sich der Bedürfnisvektor durch Ereignisse im Spiel ändert, oder sich die Rahmenbedingungen verändern. Der Einsteiger wird beispielsweise mit der Zeit seine Fähigkeiten verbessern, nicht mehr auf die Hilfe seiner Kollegen angewiesen sein und sich auch an schwierigere Aufgaben herantrauen.

Zu beachten ist auch, dass der berechnete Wert der Motivationskraft nur im Vergleich mit anderen Werten desselben Mitarbeiters in derselben Situation betrachtet werden darf. Die absolute Größe ist nicht aussagekräftig. Erfahrungen haben auch gezeigt, dass die genaue Reihenfolge im oberen Bereich der Rangfolge das Szenario adäquater erfüllt, als es im unteren Wertebereich der Fall ist. Da jedoch nur der höchste Motivationswert verwendet wird, um die durchzuführende Handlung auszuwählen, hat dies keine Auswirkungen auf die Qualität der Handlungsplanung.

Karrieremensch

Der Karrieremensch ist eine hart arbeitende Person, die finanziell abgesichert ist. Er hat keine Familie, jedoch erfordert sein hoher Lebensstandard viel Geld. Spaß ist ihm nicht wichtig und er legt wenig Wert auf seine Freizeit. Am Ende zählen für ihn nur die Ergebnisse, und um diese zu erreichen ist er sehr fleißig und korrekt.

Ein solches Profil kann beispielsweise durch folgenden Bedürfnisvektor abgebildet werden:

$$\vec{n} = \begin{pmatrix} 0.0 \\ 0.1 \\ 0.2 \\ 1.0 \\ 0.7 \\ 0.3 \end{pmatrix} \begin{matrix} \textit{Spaß} \\ \textit{Freizeit und Gesundheit} \\ \textit{Status} \\ \textit{Macht, Einfluss, und Verantwortung} \\ \textit{Finanzen und Sicherheit} \\ \textit{persönliche Weiterentwicklung} \end{matrix} \tag{2.72}$$

Der Schwerpunkt des Karrieremenschen liegt hauptsächlich auf Macht, Einfluss, und Verantwortung, und dem Erwerb von finanziellen Mitteln. Alle anderen Bedürfnisse spielen nur eine untergeordnete Rolle.

Auf Basis dieser Bedürfnisse berechnet das VIE-Modell für das Rahmenszenario folgende Rangfolge der Handlungsalternativen:

	Aktivität	**Artefakt**	**beteiligte Kollegen**	**Motivationskraft**
1.	Neutral	Einfach	—	0.727
2.	Neutral	Einfach	sympathischer Kollege	0.575
3.	Neutral	Einfach	erfahrener Kollege	0.575
	...	...	...	...
17.	Neutral	Schwierig	erfahrener Kollege	0.355
18.	Neutral	Training	erfahrener Kollege	0.322
19.	Neutral	Schwierig	—	0.318
	...	...	...	...
72.	Nichts tun	—	—	-0.240
73.	Negativ	Schwierig	smyp. K., unsypm. K. erf. K.	-0.240

Die Handlungswahl des Karrieremenschen ist leicht nachvollziehbar. Da er ein disziplinierter Mensch ist, der sich möglichst genau an die ihm gestellten Vorgaben hält, orientiert sich seine Wahl stark am vorliegenden Projektplan. Die entsprechenden Handlungsalternativen, finden sich somit am oberen Ende der Rangfolge. Er weicht weder von der vorgegebenen Aktivität, noch von den vorgegebenen Artefakten ab. Dabei bevorzugt er die einfachere Aufgabe, da er hier mit schnelleren und besseren Ergebnissen rechnen kann. Die Handlungen auf Platz 1-16 betreffen alle das einfache Artefakt. Das schwierige Artefakt findet sich zum ersten Mal auf Platz 17. Da er durch die Unterstützung des erfahrenen Kollegen deutlich bessere Ergebnisse erwarten kann, möchte er diesen entsprechend

hinzuziehen. Auf Platz 18 findet sich eine Handlung, die dem Karrieremenschen hilft, seine Fähigkeiten, die er für die ihm gestellten Aufgaben benötigt, zu verbessern.

Der Karrieremensch hat keine Hemmungen andere Kollegen um Hilfe zu bitten, wenn er sich dadurch einen Vorteil verspricht. Dass diese dabei bei ihrer eigenen Arbeit gestört werden, beeinflusst ihn wenig. Ihn interessiert auch nicht, ob die Zusammenarbeit Spaß macht, sondern nur die zu erwartenden Ergebnisse. Da ihm beispielsweise der erfahrene Kollege bei der schwierigen Aufgabe helfen kann, wird dieser entsprechend miteinbezogen, auch wenn dazu vom Projektplan abgewichen werden muss.

Zusammenfassend lässt sich feststellen, dass sich der Karrieremensch deutlich stärker am Projektplan orientiert, als der Berufseinsteiger. Sein oberstes Ziel ist es, die geforderten Ergebnisse möglichst schnell und in guter Qualität abzuliefern. Kann er dies nicht leisten, stellt er die entsprechende Aufgabe nach hinten oder greift auf seine Kollegen zurück, um seine Defizite zu kompensieren. Dabei ignoriert er bei Bedarf die personelle Besetzung, die im Projektplan für die einzelnen Aufgaben vorgesehen ist.

Tüftler

Der Tüftler ist ein sehr interessierter Mensch. Seine Karriere ist ihm nicht besonders wichtig, Geld benötigt er so gut wie keines. Er möchte vor allem Spaß bei der Arbeit haben und hat gleichzeitig ein großes Interesse daran, seine Fähigkeiten zu verbessern und neue Dinge zu lernen. Sein sozialer Status und die Anerkennung seiner Kollegen sind ihm dabei egal.

Ein passender Bedürfnisvektor des Tüftlers kann wie folgt aussehen:

$$\vec{n} = \begin{pmatrix} 0.9 \\ 0.3 \\ 0.1 \\ 0.1 \\ 0.1 \\ 1.0 \end{pmatrix} \begin{matrix} \textit{Spaß} \\ \textit{Freizeit und Gesundheit} \\ \textit{Status} \\ \textit{Macht, Einfluss, und Verantwortung} \\ \textit{Finanzen und Sicherheit} \\ \textit{persönliche Weiterentwicklung} \end{matrix} \tag{2.73}$$

Daraus ergibt sich folgende Rangfolge:

	Aktivität	**Artefakt**	**beteiligte Kollegen**	**Motivationskraft**
1.	Positiv	Training	sympathischer Kollege	0.514
2.	Positiv	Training	sympathischer Kollege, erfahrener Kollege	0.504
3.	Positiv	Training	—	0.451
	...	...	...	...
7.	Neutral	Training	erfahrener Kollege, sympathischer Kollege	0.408
8.	Positiv	Einfach	sympathischer Kollege	0.380
9.	Positiv	Einfach	sympathischer Kollege, erfahrener Kollege	0.370
10.	Positiv	Training	unsympathischer Kollege	0.470
	...	...	...	...
28.	Neutral	Einfach	sympathischer K.	0.250
	...	...	...	...
31.	Neutral	Schwierig	sympathischer Kollege, erfahrener Kollege	0.236
49.	Nichts tun	—	—	-0.143
	...	...	...	...
73.	Negativ	Schwierig	—	-0.081

Im Gegensatz zu den anderen beiden Charakteren, ignoriert der Tüftler die ihm gemachten Vorgaben weitgehend. Die von ihm am höchsten priorisierte Handlung, die positive Aktivität auf dem Trainingsartefakt auszuführen, ist nicht im Projektplan enthalten. Er wählt diejenige Handlung, die ihm Spaß macht und bei der er am meisten lernen kann. Aufgaben bearbeitet er lieber mit einem sympathischen Kollegen, beispielsweise einem Freund, als alleine. Daher entspricht die Handlung mit der höchsten Motivationskraft keiner Aufgabe des Projektplans. Stattdessen wird die Aktivität mit dem höchsten Sympathiewert gewählt und auf das Artefakt mit dem höchsten Lerneffekt angewendet.

Wenn dem Tüftler etwas keinen Spaß macht, lässt er es sein. Entsprechend finden sich die laut Rahmenszenario im Projektplan für ihn vorgesehenen Aufgabe in der Rangliste sehr weit hinten auf Platz 28 und 31. Jedoch auch hier arbeitet er lieber mit einem Kollegen zusammen, weil ihm das mehr Spaß macht als alleine zu arbeiten, obwohl der Projektplan das anders vorsieht. Im Gegensatz zu den oben beschriebenen Charakteren ist seine Hemmschwelle zum *Nichts tun* deutlich geringer. Die dazugehörige Handlung findet sich in der Mitte der Rangliste. Am Ende der Rangliste sind hingegen alle Handlungen zu finden, die mit der *negativen Aktivität* verbunden sind und deren Schwierigkeit keinen Lerneffekt verspricht. Der Tüftler tut somit eher nichts, als eine Handlung zu wählen, die nicht in sein Konzept passt.

2.6.8. Zusammenhang zwischen Bedürfnisvektor und Motivationsmodell

Das Simulationsmodell trennt die Handlungsplanung von der eigentlichen Handlungsausführung. Aus diesem Grund unterscheidet das Simulationsmodell zwischen den Anspruchswerten des Motivationsmodells, und den Komponenten des Bedürfnisvektors, obwohl diese sehr ähnliche Informationen enthalten und beide auf den Ergebnissen der Pittsburgh-Studie basieren. Deshalb ist es bei der Konfiguration des Charakters eines Mitarbeiters wichtig, zu berücksichtigen, dass sich die beiden Teile nicht widersprechen, um einen in sich konsistenten Charakter zu erzeugen. Bei einem inkonsistenten Charakter könnte beispielsweise das VIE-Modell mit einem Schwerpunkt auf Weiterentwicklung konfiguriert werden, während die Anspruchswerte im Motivationsmodell so gewählt sind, dass Lerneffekte sich nicht positiv auf die Motivation auswirken. Prinzipiell ist eine solche Konstellation zwar denkbar, jedoch sehr ungewöhnlich und deshalb für ein Planspiel typischerweise ungeeignet.

Um eine widersprüchliche Konfiguration zu vermeiden, können die Parameter des Motivationsmodells aus dem Bedürfnisvektor generiert werden, indem die Abbildung des Bedürfnisvektors auf den Folgenvektor (Formel 2.69, Formel 2.70) verwendet wird. Der sich dabei ergebende Valenzvektor der Folgen kann anschließend auf die Anspruchswerte im Motivationsmodell übertragen werden, da beide auf den Kategorien der Pittsburgh-Studie basieren. Folgende Zuordnung ist ein Vorschlag, wie eine solche Übertragung definiert werden kann:

Valenz der Folge im VIE-Modell	$\rightarrow$	**Anspruch im Motivationsmodell**
Beförderung und Lohn		Entlohnung und Gehalt
Sicherheit der Arbeitsstelle		Sicherheit der Arbeitsstelle
Beziehung zwischen Kollegen		Zwischenmenschliche Beziehungen
Freiraum		Arbeitsbedingungen
(Beförderung und Lohn + Anerkennung) $\cdot$ 0.5		Leistung und Erfolgserlebnisse
Intrinsische Motivation		Arbeitsinhalte
Wachstum		Wachstum

2.7. Realisierung des Simulationsmodells

Die Architektur des Simulationsmodells ist so gewählt, dass sie Anpassungen und Erweiterungen ermöglicht. Dadurch kann zum einen das Modell besser an die konkreten Anforderungen des Planspiels angepasst werden, indem Teile davon verfeinert, verändert oder ausgetauscht werden, zum anderen ist es möglich, das Simulationsmodell zu erweitern, um fehlende Aspekte zu ergänzen. Es ist auch möglich, Teilmodelle auszutauschen, sowie deren Parameter zu verändern, sollten sich hierfür bessere Alternativen ergeben. Dadurch wird die langfristige Weiterentwicklung des Simulationsmodells ermöglicht.

Die gewählte Architektur erlaubt Anpassungen und Erweiterungen des Simulationsmodells auf verschiedenen Ebenen. Es können die einzelnen Agenten verändert und angepasst werden, um spezielles Verhalten unabhängig vom Simulationsmodell zu realisieren. Diese Anpassung kann unabhängig von der operationalen Semantik des Simulationsmodells geschehen, und erlaubt damit prinzipiell ein beliebiges Verhalten des Agenten, das jedoch gegebenenfalls noch implementiert werden muss. Dabei können die einzelnen Teile des Simulationsmodells wiederverwendet werden, da sie seiteneffektfrei implementiert sind, und damit vom Rest des Simulationsmodells unabhängig verwendet werden

können. Für die Agenten, welche die operationale Semantik des Simulationsmodells nutzen, sollten die Abläufe auf Agentenebene nicht verändert werden, um eine korrekte Ausführung zu gewährleisten. Die einzelnen Elementarmodelle können jedoch problemlos im Rahmen ihrer Schnittstellen verändert, beziehungsweise ausgetauscht werden, um die von ihnen abgebildeten Aspekte durch Alternativen zu realisieren. Auf unterster Ebene können die einzelnen Parameter, und die in die Berechnung eingehenden mathematischen Funktion verändert werden, um die Elementarmodelle bei Bedarf an ein konkretes Planspiel anzupassen.

Objektorientierung

Um eine Änder- und Erweiterbarkeit zu ermöglichen, ist das Simulationsmodell nach dem Ansatz der objektorientierten Programmierung entworfen. Abbildung 2.43 zeigt eine Übersicht über die Klassenstruktur, die dem Simulationsmodell zugrunde liegt. Die Methoden liefern die Schnittstellen der Modelle, und passen entsprechend zu den oben beschriebenen Elementar-, Tätigkeits-, und Verhaltensmodellen. Alle Assoziationen sind, wenn nicht anders angegeben, als 1:1 Relationen zu betrachten.

Die zentrale Entität ist die Klasse *Spiel*, die alle Elemente der Simulation enthält. Ihre Instanzen verfügen über direkte oder indirekte Referenzen auf alle notwendigen Daten, sowie auf die zu verwendenden Teilmodelle. Alle oben beschriebene Elementarmodelle sind in entsprechende Klassen gekapselt, sodass diejenigen Modelle, die inhaltlich eng zusammenhängen und daher gut aufeinander abgestimmt werden müssen, in jeweils einer gemeinsamen Klasse zu finden sind. Die einzelnen Elementarmodelle sind über eine oder mehrere seiteneffektfreie Methoden zugreifbar.

Bei der Initialisierung des Spiels werden die Modellklassen instanziiert und der Spielinstanz zugewiesen, um sie darüber zentral zugreifbar zu machen. Um Elementarmodelle zu verändern, kann die jeweilige Klasse vererbt, und die darin enthaltenen Methoden überschrieben werden. Bei der Initialisierung muss lediglich die in der Spielinstanz abgelegte Modellinstanz gegen eine Instanz der neuen Modellklasse ersetzt werden. Eine Besonderheit stellt dabei das Motivationsmodell dar, das zustandsbehaftet ist, und daher für jeden Projektbeteiligten einzeln instanziiert werden muss.

Spezialisierung von Aktivitäten und Artefakten

Aktivitäten und Artefakte können bei Bedarf erweitert werden, um sie an ein konkretes Szenario beziehungsweise Planspiel anzupassen.

Die Tätigkeitsmodelle sind in der Klasse *Aktivität* und deren Kindklassen *Fortschrittsaktivität*, *Qualitätssicherungsaktivität* und *Kommunikationsaktivität* realisiert. Dabei sind vor allem die in die Modelle eingehenden Funktionen für den Einfluss der Gruppengröße F_a, sowie des Kontexteinflusses Z, und Wissenseinflusses F_k durch die Methoden `EinflussGruppengröße`, `KontextEinflussSelbst`, `KontextEinflussAndere`, und `EinflussWissensabdeckung` umgesetzt. Diese können überschrieben werden, um sie so für die einzelnen Aktivitäten individuell anpassen zu können. Auf diese Weise wird Qualitätssicherungsarbeit nochmals in *Testen* und *Inspektion* unterteilt. Die Klasse Aktivität realisiert die gemeinsamen Aspekte der Tätigkeitsmodelle. Die bei den Tätigkeitsmodellen beschriebenen spezifischen Aspekte von Testen und Inspektion sind in den jeweiligen Kindklassen umgesetzt.

Der Teil des Modells für die Artefaktbearbeitung, der die ermittelte Arbeitsleistung auf die Artefakte anwendet, ist in der Klasse *Artefakt* in den Methoden `Bearbeiten` und `DefektEntfernen` implementiert. Dadurch lässt sich die abstrakte Darstellung der Artefakte, wie sie das Simulationsmodell definiert, besser anpassen und konkretisieren. Ist der Abstraktionsgrad für ein bestimmtes Planspiel zu groß, kann eine detailliertere Artefaktdarstellung erstellt werden, indem die entsprechenden Methoden überschrieben, und die Artefaktklasse gegebenenfalls um weitere Attribute erweitert wird.

Um zu erläutern, welches Vorgehen dafür vorgesehen ist, dient das folgende Beispiel, bei dem angenommen wird, dass in einem Planspiel statt der abstrakten Artefaktrepräsentation für Codefragmente, konkrete Codeausschnitte verwendet werden sollen. Dem Spieler werden der Fortschritt und die Qualität der Artefakte nicht nur als abstrakte Werte dargestellt, er kann auch den tatsächlichen Code betrachten.

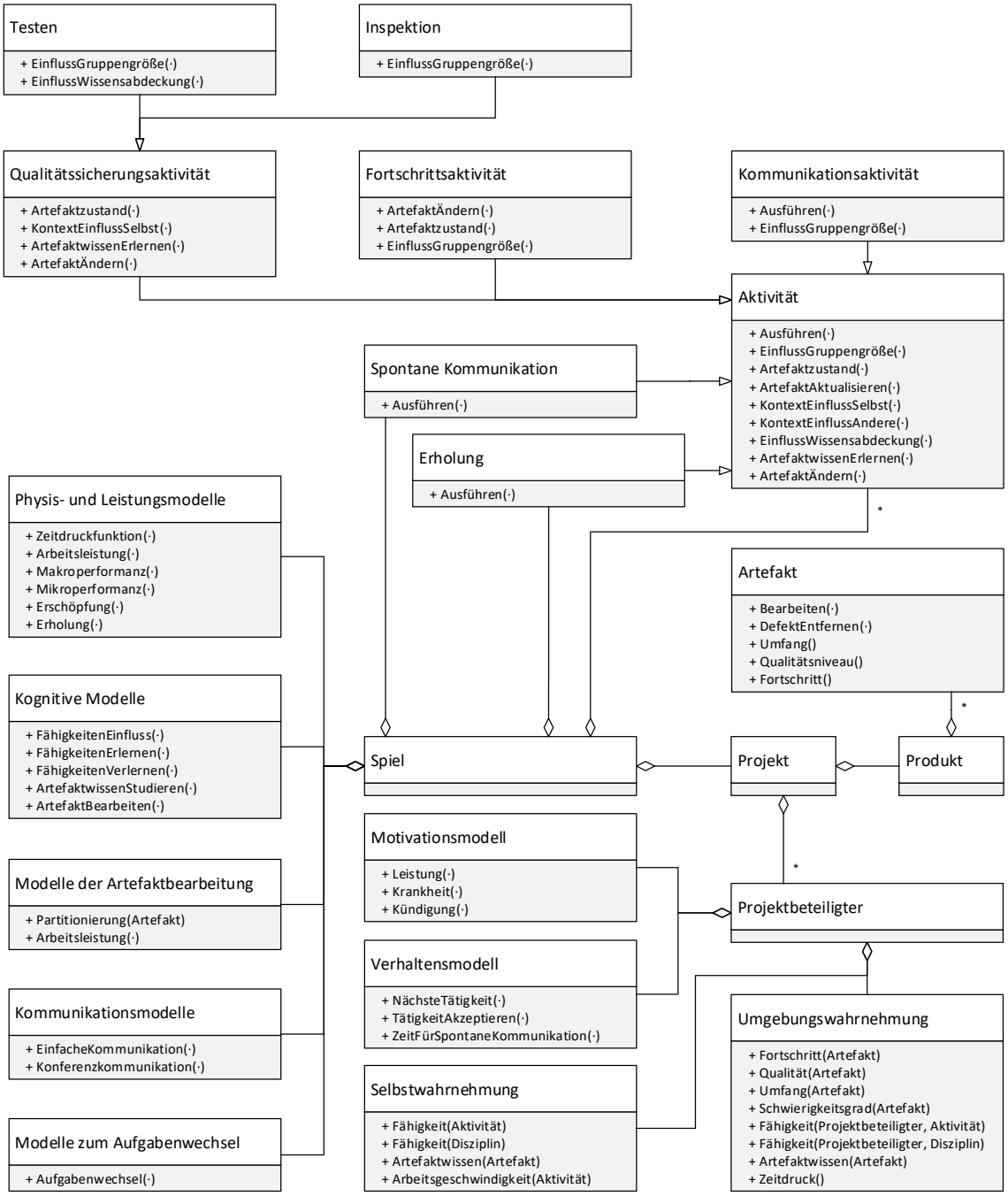


Abbildung 2.43.: UML-Klassendiagramm zum Aufbau der Modellarchitektur

Dies kann mittels einer neuen Klasse umgesetzt werden, die von der Artefaktklasse des Simulationsmodells abgeleitet wird. Alle neuen Aspekte werden in der neuen Klasse realisiert. Einer davon ist der konkrete Code des Artefakts. Dieser muss bei der Definition des Szenarios vom Spieleentwickler anhand der Lernziele des Spiels festgelegt werden. Anhand des Codes kann die vom Simulationsmodell benötigte Eigenschaft für den Umfang des Artefakts berechnet werden. Dazu wird die Methoden `Umfang` überschrieben und mittels einer geeigneten Metrik ein einzelner Wert berechnet, der angibt, wie viele Stunden ein durchschnittlicher Mitarbeiter benötigt, um diesen Code zu erstellen. Einen möglichen Ansatz dafür bietet beispielsweise die Kostenschätzung mittels COCOMO.

Ebenso angepasst werden kann die Repräsentation der Defekte, um konkrete Fehler, beispielsweise

in Form von Mutanten des Ursprungscodes, zu ermöglichen. Dazu muss zum einen die Defektklasse um die Repräsentation der Mutanten erweitert werden, zum anderen muss die Methode `Bearbeiten` des Artefakts verändert werden, um diese Defekte auf Basis der vom Simulationsmodell berechneten Arbeitsleistung zu erzeugen.

Mittels des im Artefakt hinterlegten Codes, des Werts für seinen Fortschritt, und der Menge der im Artefakt enthaltenen Defekte kann in einer neu zu definierenden Methode der aktuelle Zustand des Codes gebildet werden, der dem Spieler angezeigt werden kann.

Auf die gleiche Weise, können auch Aktivitäten konkretisiert werden. Dazu können sowohl die einzelnen Methoden der Aktivitätsklasse, welche die aktivitätsspezifischen Funktionen enthalten, als auch das Tätigkeitsmodell oder einzelne Teile davon ausgetauscht werden.

Wie das Beispiel zeigt, beschränken sich alle notwendigen Änderungen für die Artefaktanpassung auf die Artefaktklasse. Weitere Klassen müssen nicht betrachtet werden, und das eigentliche Simulationsmodell bleibt unverändert. Die Architektur des Simulationsmodells ist darauf ausgelegt, Anpassungen möglichst lokal zu ermöglichen. So sind diese einfacher durchzuführen, und der Entwickler muss sich nur in die dafür relevanten Teile des Simulationsmodells einarbeiten.

Spezialisierung und Individualisierung des Verhaltens von Projektbeteiligten

Wie die Aktivitäten und Artefakte, so können auch die Projektbeteiligten, insbesondere deren Wahrnehmung und Verhalten an das konkrete Planspiel angepasst werden. Dabei besteht auch die Möglichkeit, für die einzelnen Projektbeteiligten in einer Simulation ein jeweils individuelles Verhalten zu entwickeln.

Die beiden Wahrnehmungskomponenten *Selbstwahrnehmung* und *Umgebungswahrnehmung* sind, wie auch das *Motivationsmodell*, jeweils als eigene Instanz beim jeweiligen Projektbeteiligten referenziert. Damit ist es möglich, für jeden Projektbeteiligten eine individuelle Wahrnehmung zu realisieren, die über die reine Parameterverzerrung der im Simulationsmodell enthaltenen Komponente hinausgeht. Dazu können, unter Beibehaltung der vom Rest des Simulationsmodells genutzten Schnittstellen, die entsprechenden Methoden überschrieben werden. Eine Anpassung der Wahrnehmung ermöglichen es beispielsweise, Mitarbeiter zu realisieren, die Aufwände konsequent unterschätzen, dazu muss die Methode `Umfang` der Umgebungswahrnehmung ausgetauscht werden, oder ihre eigenen Fähigkeiten überschätzen, indem die Methode `Fähigkeit` der Selbstwahrnehmung verändert wird. Ob, und welche Anpassungen sinnvoll sind, hängt dabei von den jeweiligen Lernzielen und dem damit verbundenen Szenario des Planspiels ab.

Jeder Projektbeteiligte verfügt über eine individuelle Instanz eines Verhaltensmodells, das zur Planung seiner Tätigkeiten verwendet wird. Durch das Austauschen dieser Instanz, kann nicht nur zwischen den drei im Simulationsmodell enthaltenen Verhaltensmodellen *Direkte Planung*, *Dynamische Arbeitszeitplanung* und *VIE-Planung* gewählt werden, es besteht auch die Möglichkeit, diese in veränderter Form einzusetzen oder eigene Modelle zu verwenden, solange diese die Schnittstellen des Simulationsmodells `NächsteTätigkeit`, `TätigkeitAkzeptieren` und `ZeitFürSpontaneKommunikation` erfüllen. Die Menge der möglichen Verhaltensweisen und -muster, die Projektbeteiligte im Rahmen eines Projekts haben können, ist zu groß, um sie mit einem generellen Modell abzubilden. Das Simulationsmodell liefert dazu mit den enthaltenen Verhaltensmodellen einen möglichen Ansatz, erlaubt es jedoch auch eigen, auf das jeweilige Planspiel angepasste Modelle zu entwickeln.

3 Evaluation des Simulationsmodells

Dieses Kapitel beschreibt die Evaluation des Simulationsmodells. Ziel der Evaluation ist es, die Validität des Modells nachzuweisen, und damit zu belegen, dass die Konzeption im Allgemeinen und das Modell im Speziellen für die Simulation von Projektverläufen von Softwareprojekten geeignet ist.

> There is nothing in either physical or social sciences about which we have perfect information. We can never prove that a model is an exact representation of "reality". Conversly, among those things of which we are aware, there is nothing of which we know absolutely nothing. So we always deal with information which is of intermediate quality – it is better than nothing and short of perfection. Models are then to be judged, not on an absolute scale than condemns them for failure to be perfect, but on a relative scale that approves them if they succeed in clarifying out knowledge and our insights into systems.
>
> — Jay Forrester [Forrester, 1968]

Da ein Modell immer eine Abstraktion der Realität ist und diese entsprechend nur verkürzt darstellt, ist der Nachweis einer absoluten Korrektheit unmöglich. Entsprechend muss Validität in diesem Kontext anders definiert werden: Ein Modell ist valide, wenn es korrekt für einen bestimmten Aspekt eingesetzt werden kann [Forrester, 1980]. Somit ist es unerlässlich den konkreten Zweck eines Modells festzulegen, bevor es etabliert werden kann. Das in dieser Arbeit besprochene Simulationsmodell ist für den Einsatz in der Lehre entwickelt worden. Mit dem Modell sollen Planspiele gebaut werden, anhand derer Studierende die Abläufe im Bereich der Softwareentwicklung kennen und verstehen lernen können. Dazu muss das Modell die in Softwareprojekten vorhandenen Zusammenhänge adäquat darstellen und die dort auftretenden Effekte erzeugen.

Eine formale Verifikation des Simulationsmodells ist aufgrund seiner Struktur und seines Umfangs ein sehr aufwändiger Prozess, der im Rahmen dieser Arbeit nicht geleistet werden kann. Die Evaluation stützt sich deshalb auf die Validierung des Modells, bei der es nicht darum geht, eine absolute Wahrheit zu zeigen, sondern vielmehr Vertrauen in das Modell zu erzeugen, was laut [Forrester, 1980] ein angemessenes Kriterium für Validität ist. Dazu stehen unterschiedliche Ansätze zur Verfügung, die in Abschnitt 3.2 genauer erläutert werden. Welche Ansätze davon für die Evaluation des Simulationsmodells ausgewählt, und wie diese umgesetzt wurden, wird in Abschnitt 3.3 ausgeführt.

Um das Simulationsmodell zu validieren, wird gezeigt, dass es die für die Lehre relevanten Effekte adäquat erzeugt. Welche Effekte ausgewählt wurden, und nach welchen Kriterien diese Auswahl stattgefunden hat, wird in Abschnitt 3.1 beschrieben. Abschnitt 3.2 beschreibt die grundlegenden Techniken zur Validierung von Systemmodellen, Abschnitt 3.3 diskutiert die daraus für die Validierung des Simulationsmodells ausgewählten Techniken. Abschnitt 3.4 bespricht die durchgeführten Experimente und deren Ergebnisse. In Abschnitt 3.5 werden die Validitätskriterien bezogen auf die durchgeführten Experimente betrachtet.

3.1. Auswahl und Konkretisierung der zu zeigenden Effekte

Um das Simulationsmodell evaluieren zu können, muss zunächst festgelegt werden, welche Eigenschaften von diesem Modell erwartet werden. Den Rahmen gibt der angestrebte Einsatzzweck, das Modell für die Entwicklung von Planspielen im Bereich der Softwaretechnik- und Projektmanagementlehre zu verwenden, vor. Auf dessen Basis müssen die konkreten nachzuweisenden Eigenschaften definiert werden.

Da sich die Evaluation an den Inhalten der Softwaretechniklehre orientiert, basieren die zu zeigenden Effekte auf den Lerninhalten, die vermittelt werden sollen. Da keine allgemeingültige Liste dieser Inhalte existiert, müssen diese zuerst identifiziert werden. Die Literatur in diesem Bereich liefert eine große Menge an relevanten Aspekten und Aussagen, die im Rahmen dieser Arbeit nicht vollständig betrachtet werden kann. Ziel ist es daher, die wichtigsten davon zu selektieren und die Evaluation auf diese zu konzentrieren.

86 Aussagen über die Softwareentwicklung

Im Rahmen ihrer Dissertation zum Thema Planspiele im Bereich Softwaretechnik hat Emily Navarro eine Liste von 85 „Fundamentalen Regeln des Software Engineering“ gesammelt [Navarro, 2006]. Diese Liste basiert auf einer Vielzahl von Aussagen in der entsprechenden Fachliteratur. Obwohl sie keinen Anspruch auf Vollständigkeit erheben darf, ist diese Liste eine wertvolle Sammlung wichtiger Aspekte aus dem Bereich der Softwareentwicklung und des dazugehörigen Projektmanagements. Die Evaluation dieser Arbeit stützt sich vor allem auf die dort gemachten Aussagen, um zu zeigen, dass das Simulationsmodell die wichtigsten Aspekte korrekt abbildet.

Im ersten Schritt wurden die von Navarro gesammelten Aussagen analysiert. Einige Punkte in Navarros Liste beinhalten mehrere Aussagen und wurden entsprechend aufgeteilt. Alle Aussagen wurden anschließend in eine der folgenden Kategorien eingeordnet:

- **Entwicklungsprozess:** beschreibt Aspekte, die sich auf die Auswahl oder Abfolge von Aktivitäten während des Entwicklungsprozesses beziehen.

  Zu dieser Kategorie gehört beispielsweise die Aussage »If you don't do a system architectual design with well-defined interfaces, integration will be a big mess.« [Navarro, 2006], die laut Navarro aus [Sommerville, 2007] entnommen ist.

- **Personalmanagement:** beschreibt Aspekte, die sich auf die Auswahl, den Einsatz oder die Führung der beteiligten Mitarbeiter beziehen. Darunter fallen insbesondere auch die Zusammenhänge des Zustands bzw. der Eigenschaften der Mitarbeiter und deren Arbeitsleistung bzw. Arbeitsergebnisse.

  Ein Beispiel für diese Kategorie ist die Aussage »If a project is late and you add more people, the project will be even later.« [Navarro, 2006], die auf Brooks' Buch *The Mythical Man-Month* [Brooks, 1995] basiert.

- **Entwicklungsaktivität:** beschreibt Aspekte, die sich auf die Art und Weise der Durchführung einer oder mehrerer bestimmter Entwicklungsaktivitäten beziehen. Beispielsweise, wie Code zu schreiben ist oder wie Tests zu entwerfen sind.

  Ein Beispiel hierfür ist die Aussage »The error detection effectiveness of reviews depends greatly on the completeness and correctness of the documents used as a reference.« [Navarro, 2006] basierend auf [Weller, 1993].

- **Arbeitsumgebung:** beschreibt Aspekte, die sich auf die Gestaltung des allgemeinen Arbeitsumfelds und die damit verbundenen Auswirkungen beziehen.

  Aussagen wie »If you disable Internet surfing, productivity will go down.« [Navarro, 2006] entnommen aus [Tvedt, 1996] gehören in diese Kategorie.

- **Projektrahmen:** beschreibt Aspekte, die den Rahmen des Projekts, d. h. die Bedingungen unter denen das Projekt abläuft, betrifft. Zu diesen Aspekten gehören vor allem die Interaktion mit dem Kunden und dessen Einfluss auf das Projekt, sowie Abhängigkeiten von Drittanbietern.

  Hierzu gehört die Aussage »The two "task-type" variables that have the greatest impact on programmer productivity are the programming language and the quality of external documentation.« [Navarro, 2006] nach [Abdel-Hamid, 1991].

Abbildung 3.1 zeigt die quantitative Verteilung der Aussagen auf die einzelnen Kategorien.

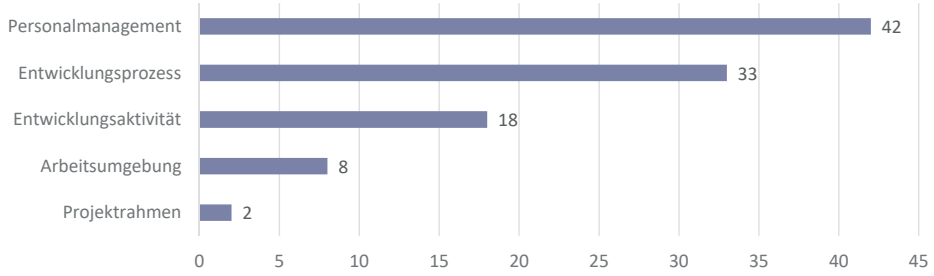


Abbildung 3.1.: Aufteilung der Aussagen auf die beschriebenen Kategorien

Nicht alle Aussagen fallen in den Rahmen, den das Simulationsmodell abdeckt. Insbesondere die Aussagen die in die Kategorie *Entwicklungsaktivität* fallen, beschreiben oft einen sehr konkreten Aspekt der Aktivitätsdurchführung. Da Aktivitäten im Simulationsmodell nur abstrakt betrachtet werden, sind solche Aussagen für die Evaluation nicht relevant. Das Simulationsmodell befasst sich beispielsweise nicht damit, wie guter Code geschrieben wird, sondern nur mit der Tatsache, dass guter Code geschrieben werden muss. Daher sind lediglich entsprechende Rahmenbedingungen, die zu gutem Code führen, wie beispielsweise eine entsprechende Ausbildung des Entwicklers, relevant.

Gleiches gilt für die Kategorien *Projektrahmen* und *Arbeitsumgebung*. Hier eingeordnete Aspekte fließen meistens nur indirekt in die Simulation ein, und können in einer entsprechenden Spielumgebung als Teil der Storyline oder durch zusätzliche Entscheidungsmöglichkeiten des Spielers umgesetzt werden. Das Verhalten des Kunden und die Gestaltung der Arbeitsumgebung sind ebenfalls nicht Teil des Simulationsmodells und müssen bei Bedarf als entsprechende Erweiterung umgesetzt werden.

Alle Aussagen wurden zusätzlich nach folgenden Aspekten kategorisiert, um ihre Relevanz für die Evaluation des Simulationsmodells zu klären:

- **validierbar:** Das Simulationsmodell beinhaltet das erforderliche Verhalten, um den beschriebenen Effekt zu erzeugen. Der Aspekt ist damit in jeder Simulation enthalten.

  Beispielsweise kann die Aussage »A factor that affect productivity is work force experience level.« [Navarro, 2006] aus [Abdel-Hamid, 1991] durch eine positive Korrelation der Produktivität mit den vorhandenen Fähigkeiten bzw. des Projektwissens der Mitarbeiter am Simulationsmodell gezeigt werden.

- **modellierbar:** Der Aspekt ist nicht direkter Bestandteil des Simulationsmodells, lässt sich jedoch durch eine entsprechende Konfiguration des Szenarios modellieren und anschließend nachweisen. Er ist damit mittels des Simulationsmodells indirekt abbildbar.

  Beispielsweise erfüllt das Simulationsmodell die Aussage »Reviews of non-technical documents (e.g., requirements specification, user manual) are more effective if the customer is involved.« [Navarro, 2006] entnommen aus [Pressman, 1997], wenn für die entsprechenden Artefakte eine bestimmte Disziplin, zum Beispiel in Form von *Anwendungsdomänenwissen* gefordert wird, die nur der Kunde besitzt.

- **fehlend:** Der Aspekt ist nicht Bestandteil des Simulationsmodells und lässt sich auch nicht durch entsprechende Szenarien nachbilden. Er kann nur durch eine Erweiterung und Anpassung des Simulationsmodells in die Simulation eingebracht werden.

  Ein Beispiel für einen fehlenden Aspekt ist die Aussage »Broad application knowledge is acquired more through relevant experience than through training.« [Navarro, 2006] entnommen aus [Curtis, 1988]. Obwohl es für ein entsprechendes Planspiel ein sinnvoller Aspekt des Projektmanagements ist, lässt sich explizites Training nicht mit dem Simulationsmodell simulieren. Die Aussage lässt sich damit nicht nachweisen. Aspekte dieser Kategorie liefern häufig Ansätze für wertvolle Erweiterungen des Simulationsmodells.

- **irrelevant:** Der Aspekt liegt außerhalb des Rahmens dieser Arbeit, beschreibt ein Metakonzept, ist zu allgemein formuliert oder ist zu spezifisch auf einen konkreten Spezialfall bezogen. Er hat daher keine Relevanz für die Evaluation des Simulationsmodells. Teilweise können solche Aspekte in einer entsprechenden Spielumgebung oder Storyline, die um das Simulationsmodell geschaffen wird, realisiert werden.

  Aussagen wie »Use defensive programming« [Navarro, 2006] beziehen sich auf die Details der Aktivitätsdurchführung. Da das Simulationsmodell auf einer höheren Abstraktionsebene ansetzt, ist sie ein gutes Beispiel für Aussagen, die nicht relevant für die Evaluation dieser Arbeit sind. Sie ist außerdem ein Beispiel für Aussagen, die nicht für alle Situationen gelten und sollte daher nicht Bestandteil eines allgemeinen Simulationsmodells sein.

Abbildung 3.2 zeigt die quantitative Verteilung der Aussagen auf die Einstufung bezüglich der Prüfbarkeit. Sie zeigt, dass knapp 60% der von Navarro identifizierten Aspekte für das Simulationsmodell relevant sind, und als Grundlage der zu prüfenden Hypothesen verwendet werden können.

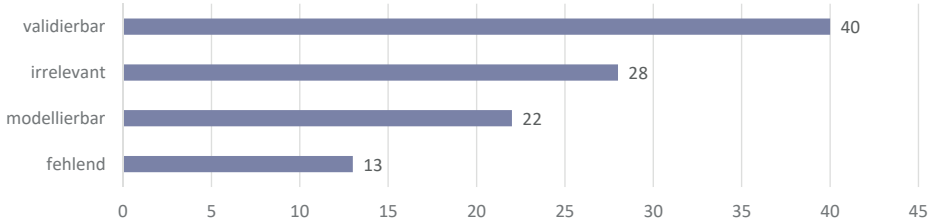


Abbildung 3.2.: Einstufung der Aussagen bezüglich des Simulationsmodells

Eine vollständige Liste aller Aussagen von Navarro, mit ihrer jeweiligen Kategorisierung und der dazugehörigen Begründung findet sich in Anhang A.5.

Formulierung zu prüfender Hypothesen

Um das Simulationsmodell evaluieren zu können, müssen die zu prüfenden Aussagen als entsprechende Hypothesen formuliert werden. Die Zuordnung der Aussagen von Navarro zu den hier beschriebenen Hypothesen findet sich in Anhang A.5. Die einzelnen Hypothesen werden im Rahmen der dazugehörigen Experimente und deren Ergebnisse in Abschnitt 3.4.1 genauer beschrieben. Folgende Aussagen über das Simulationsmodell werdens geprüft:

H1 *Je höher die Fähigkeiten eines Mitarbeiters für eine Aktivität sind, umso produktiver ist er dabei.*

H2 *Je größer das Artefaktwissen eines Mitarbeiters ist, umso höhere ist seine Produktivität.*

H3 *Je höher die Schwierigkeit eines Artefakts ist, umso geringer ist die Produktivität der daran arbeitenden Mitarbeiter.*

H4 *Je höher die Produktivität der Mitarbeiter ist, umso geringer ist die benötigte Zeit für die Fertigstellung des Projekts.*

H5 *Je höher die Sympathie eines Mitarbeiters für eine Aufgabe ist, umso höher ist seine Produktivität bei der Bearbeitung dieser Aufgabe.*

H6 *Je höher die Adaptionsfähigkeit eines Mitarbeiters ist, umso produktiver ist er.*

H7 *Je besser die Voraussetzungen für einen Arbeitsschritt sind, umso höher ist die Produktivität in diesem Arbeitsschritt.*

H8 *Je höher die Motivation eines Mitarbeiters ist, umso höher ist seine Produktivität.*

H9 *Es existiert ein Punkt, an dem eine Steigerung der Arbeitsbedingungen zu keinem Anstieg der Motivation mehr führt.*

H10 *Je größer das Mitarbeiterteam ist, umso früher wird das Projekt fertiggestellt.*

H11 *Je größer das Mitarbeiterteam ist, umso niedriger ist die Produktivität pro Mitarbeiter.*

H12 *Es ist effektiver, wenige Mitarbeiter mit hohen Fähigkeiten zu nutzen, als viele Mitarbeiter mit niedrigen Fähigkeiten.*

H13 *Werden zu einem späten Zeitpunkt im Projekt neue Mitarbeiter hinzugefügt, verzögert sich das Projektende.*

H14 *Je mehr Defekte in einem Dokument sind, umso höher ist die Rate mit der Defekte darin aufgedeckt werden.*

H15 *Mitarbeiter, die an Qualitätssicherungsaktivitäten teilnehmen, erweitern dadurch ihr Artefaktwissen.*

H16 *Der Autor eines Dokuments kann die darin gefundene Defekte schneller beheben als der Mitarbeiter, der sie gefunden hat.*

H17 *Begleitende Qualitätssicherung führt zu einer höheren Produktivität als Qualitätssicherung ausschließlich am Ende eines Projekts.*

H18 *Inspektionen sind mit einer Gruppe bestehend aus 4 Mitarbeitern am effizientesten.*

H19 *Je höher der Zeitdruck auf einen Mitarbeiter ist, umso niedriger ist seine Produktivität, wenn der Zeitdruck über einen langen Zeitraum aufrechterhalten bleibt.*

H20 *Je höher der Zeitdruck auf einen Mitarbeiter ist, umso höher ist seine Produktivität, wenn der Zeitdruck nur über einen kurzen Zeitraum aufrechterhalten bleibt.*

H21 *Je mehr freie Zeit ein Mitarbeiter hat, umso niedriger ist seine Produktivität.*

H22 *Mitarbeiter, die an einer gemeinsamen Aufgabe arbeiten sind produktiver, wenn sie miteinander reden.*

H23 *Wird ein Artefakt von einem Mitarbeiter zweimal erstellt, ist der Zeitaufwand für die erste Erstellung größer als für die zweite.*

H24 *Die Produktivität ist vom Umfang der Artefakte unabhängig.*

H25 *Je mehr Aufgaben parallel bearbeitet werden können, um so kürzer ist die Projektlaufzeit, vorausgesetzt, es stehen genügend geeignete Mitarbeiter zur Verfügung.*

3.2. Validierung von Systemmodellen

Dieser Abschnitt beschreibt, welche grundsätzlichen Möglichkeiten zur Verfügung stehen, um die Validität eines Simulationsmodells zu zeigen. Die Ansätze stammen im Wesentlichen aus der Arbeit von Jay W. Forrester und Peter M. Senge, in der beschrieben wird, wie System Dynamics Modelle evaluiert werden können [Forrester, 1980], und die Grundlage für die Evaluation der Modelle in [Madachy, 1994] und [Madachy, 2007] war. Dabei wird das Modell verschiedenen *Tests* unterzogen. Je mehr Tests es besteht, umso größer ist das Vertrauen in das Modell. Der Begriff des Tests darf dabei nicht mit dem des Testens in der Qualitätssicherung von Softwaresystemen verwechselt werden. Forrester und Senge verwenden ihn vielmehr im Sinne von Prüfungen, denen das Modell unterzogen wird. Die Tests von Forrester und Senge sind auf System-Dynamics-Modelle ausgelegt. Die Autoren

ordnen sie in drei Kategorien ein:

- **Tests der Modellstruktur** vergleichen die Strukturen des Originals mit denen des Modells. Dabei sollen Unterschiede und Inkonsistenzen auf struktureller Ebene aufgedeckt werden, die verhindern oder erschweren, dass das Modell adäquat auf die Realität übertragbar ist.

- **Tests des Modellverhaltens** vergleichen das vom Modell erzeugte Verhalten mit dem des Originals. Dazu werden Simulationsverläufe und -ergebnisse, die mit dem Modell berechnet, sowie Vorhersagen die mit dem Modell getroffen wurden, mit entsprechenden Referenzdaten des Originals verglichen.

- **Tests der Regelimplikation** prüfen, ob Regeln und Regeländerungen, die aus dem Modell abgeleitet werden, auf das Original übertragen werden können. Eine Regel beschreibt dabei nicht die Eigenschaften des Modells, sondern die Vorgaben nach denen es verwendet wird. Eine Regel ist beispielsweise eine Unternehmensdirektive, ein gesetzlicher Rahmen, ein vorgegebener Prozess oder eine Managementstrategie.

Obwohl die von Forrester und Senge beschriebenen Tests auf System-Dynamics-Modelle ausgelegt sind, lassen sich einige davon auch auf das Simulationsmodell anwenden. Wird anhand des Modells eine neue Regel identifiziert, ist dies mit einem Lernerfolg gleichzusetzen. Je mehr korrekte Regelimplikationen das Simulationsmodell beim Lernenden verursacht, umso höher ist der durch das Modell verursachte Lerneffekt. Dabei ist wichtig, dass die gefundenen Regeln auch auf das Original übertragen werden können.

Die folgenden Tests sind eine Zusammenfassung der in [Forrester, 1980] und [Madachy, 2007] beschriebenen Möglichkeiten.

Tests der Modellstruktur

- **Strukturverifikation.** Dieser Test vergleicht die Struktur des Modells direkt mit der Struktur des realen Systems. Da das Modell eine Verkürzung der Realität ist, bildet es im Allgemeinen auch deren Struktur nicht vollständig ab. Um diesen Test zu bestehen ist es ausreichend, die Konsistenz zwischen Modellstruktur und Originalstruktur zu belegen.

  Ein entsprechender Test besteht typischerweise in einer Modellinspektion durch einen oder mehrere Experten oder einem Abgleich der entsprechenden Annahmen bezüglich der Struktur mit der relevanten Literatur.

- **Parameterverifikation.** Um die Parameter des Modells zu validieren, werden diese mit den Parametern des realen Systems verglichen. Ziel ist, es zu zeigen, dass die Parameter des Simulationsmodells den aus der Realität bekannten Parametern entsprechen. Dabei ist sowohl der konzeptionelle Aspekt, der prüft, ob die Parameter der Simulation eine Entsprechung im realen System haben, als auch der numerische Aspekt, d. h. ob die Werte und Wertebereiche der Simulationsparameter plausible sind, relevant.

- **Konsistenz der Modelldimensionen.** Dieser Test überprüft, ob die Werte der Variablen des Modells in einem sinnvollen Bereich liegen. Dazu werden insbesondere deren Einheiten und die verwendeten Umrechnungen betrachtet. Ziel ist es, zu zeigen, dass das Modell hinsichtlich der Dimensionen seiner Variablen ausgeglichen ist.

- **Adäquatheit der Modellgrenzen.** Da ein Modell immer nur einen Ausschnitt der Realität abbilden kann, ist ein wichtiger Aspekt in Bezug auf die Zweckmäßigkeit des Modells, ob die Grenzen des Modells an der richtigen Stelle gezogen wurden. Dabei ist die Frage zu klären, ob alle relevanten Aspekte im Modell vorhanden sind. Welche Aspekte relevant sind, hängt dabei vom Zweck des Modells ab. Dabei ist insbesondere zu prüfen, ob die benötigten Parameter gesetzt, und die gewünschten Ergebnisse am Modell abgelesen werden können.

  Ob ein Modell für einen bestimmten Zweck vollständig ist, ist schwierig zu prüfen. Die Prüfung besteht daher meistens aus einer guten Argumentation. Alternativ kann man zeigen, dass

ein zusätzlicher Teilaspekt, den man dem Modell hinzufügen würde, keinen Einfluss auf das Ergebnis der Studie hat, die mit dem Modell durchgeführt werden soll. Dazu muss dieser Aspekt allerdings modelliert und in das Modell integriert werden. Wird ein Modell durch so einen Test erweitert, müssen gegebenenfalls auch die Ergebnisse der anderen Tests neu betrachtete werden.

Tests des Modellverhalten

- **Parametersensitivität.** Dieser Test soll zeigen, wie sich das Verhalten des Modells verändert, wenn die Parameter des Systems verändert werden. Zusammenhänge zwischen einer Parameteränderung und einer Änderung des Verhaltens, müssen auch im Original vorhanden sein.

  Dazu werden die Parameterwerte gezielt oder zufällig verändert, und dabei der Einfluss auf das Verhalten beobachtet und analysiert. Typischerweise wirken sich jeweils nur wenige Parameter auf ein bestimmtes Verhalten aus, während die anderen Parameter keinen großen Einfluss haben. Um diesen Test zu bestehen, muss der beobachtete Zusammenhang zwischen Parameteränderung und Verhaltensänderung im realen System nachvollzogen werden können.

- **Extrembedingungen.** Dieser Test hilft sicherzustellen, dass das Modell auch für Randfälle korrekte Ergebnisse liefert. Das soll unabhängig davon sein, ob diese in der Realität auftreten oder nicht. Auch wenn der Modelleinsatz klar auf einen bestimmten Parameterbereich eingeschränkt ist, ist das Testen von Extrembedingungen hilfreich, um grundsätzliche Mängel in der Modellstruktur aufzudecken. Unerwünschte Effekte, die unter Extrembedingungen auftreten, treten oft auch unter Normalbedingungen auf, sind dort jedoch schwerer erkennbar.

- **Reproduktion von Verhalten.** Ziel dieses Tests ist es zu prüfen, wie gut das durch das Modell generierte Verhalten zum beobachteten Verhalten des Originals passt.

  Typischerweise wird ein Modell entwickelt, um einen bestimmten Sachverhalt eines realen Systems abzubilden. Ein solcher Sachverhalt äußert sich typischerweise durch seine Auswirkungen auf die Systemumgebung oder kann durch nach außen sichtbare Änderungen des Systems wahrgenommen werden. In diesem Test wird der Sachverhalt mittels des Modells nachgebildet. Dazu werden die im Original herrschenden Ausgangsbedingungen zum Beobachtungszeitpunkt auf das Modell übertragen, und anschließend geprüft, ob die im Original beobachteten Veränderungen auch im Modell vorhanden sind.

  Liegen empirische Daten über das Original vor, können diese mit denen des Modells verglichen werden. Da das Modell keine exakte Nachbildung der Realität ist, ist auch nicht zu erwarten, dass die von ihm erzeugten Daten exakt mit denen des Originals übereinstimmen. Um eine Vergleichbarkeit herzustellen, können statistische Auswertungsmethoden herangezogen werden.

  Korrelationsanalysen zwischen den unterschiedlichen Eingabe- und Ausgabewerten des Modells können Hinweise auf Abhängigkeiten zwischen den Parametern liefern. Werden die gleichen Korrelationen in Vergleichsdaten, die aus dem realen System stammen, gefunden, ist das ein Hinweis auf die Validität des Modells.

  Eine weitere Möglichkeit ist, die Auswirkungen unterschiedlicher Rahmenbedingungen, zu vergleichen. Liegen Aussagen oder Daten bezüglich des Originals vor, die signifikante Änderungen eines oder mehrerer Parameter in Abhängigkeit unterschiedlicher Rahmenbedingungen zeigen, können diese gegebenenfalls auch im Modell nachgewiesen werden. Entsprechende statistische Test müssen auf den Daten des Originals und den Daten des Modells zu den gleichen Erkenntnissen führen. Umgekehrt darf das Modell keine Unterschiede für verschiedene Rahmenbedingungen aufweisen, die nicht auch im Original zu finden sind.

- **Vorhersage von Verhalten.** Dieser Test soll zeigen, ob mit dem Modell Muster und Ereignisse für die Zukunft vorhergesagt werden können. Er stellt das Gegenstück zum Test zur Reproduktion von Verhalten dar, bei dem die Beobachtungen aus der Realität im Modell nachvollzogen werden. Beim Vorhersagetest wird das am Modell beobachtete Verhalten in

der Realität nachvollzogen. Beide Tests belegen gemeinsam, dass eine Übertragung der Beobachtungen und Erkenntnisse zwischen Original und Modell in beide Richtungen möglich ist.

Nachzuweisen, dass ein Modell das Verhalten eines Systems vorhersagen kann, ist ein schwieriger Test. Das typische Vorgehen dabei ist, das Modell entsprechend eines realen Systems zu konfigurieren und anschließend anhand des Modells eine Prognose für die Zukunft des realen Systems aufzustellen. Der Test ist erfolgreich, wenn die Prognose später am realen System in einem entsprechenden Rahmen von Ungenauigkeiten beobachtet werden kann. Für einen adäquaten Nachweis sind mehrere solche Versuche notwendig. Dieser Test kann daher meist nur über einen längeren Zeitraum durchgeführt werden.

- **Beobachtung von Verhaltensanomalien.** Während des ganzen Testprozesses kann es passieren, dass anormale Effekte beobachtet werden, die der Realität direkt oder indirekt widersprechen. In diesem Fall liegt in der Regel entweder ein Fehler im Modell oder den Annahmen auf denen das Modell aufbaut vor. Je mehr Tests ohne auftretende Verhaltensanomalien durchgeführt werden können, umso größer wird das Vertrauen in die Korrektheit des Modells. Dieses Testkonzept wird begleitend zu den anderen Tests angewandt, indem deren Ergebnisse auf entsprechende Anomalien hin untersucht werden.

- **Beobachtung unerwarteten Verhaltens.** Ähnlich zu den anormalen Effekten kann es passieren, dass unerwartete Effekte während der Modellevaluation beobachtet werden. Wenn diese Effekte der Realität nicht direkt widersprechen, liegt nicht zwangsläufig eine Anomalie vor. Es kann auch sein, dass bisher unbeachtetes Verhalten des realen Systems durch das Modell aufgedeckt wurde.

  Es ist wichtig, den beobachteten Effekt weiter zu untersuchen, um entscheiden zu können, ob hier ein Fehlverhalten des Modells vorliegt, oder ob durch die Beobachtung eine neue Erkenntnis in Bezug auf das reale System erlangt werden kann. Dazu sollten zuerst die Zusammenhänge im Modell geklärt werden, um zu verstehen wie das beobachtete Verhalten dort zustande gekommen ist. Anschließend sollten diese Zusammenhänge auch in der Realität nachgewiesen werden. Dazu muss die Beobachtung aus der Simulation im realen System reproduziert werden können.

Tests der Regelimplikation

- **Systemverbesserungen.** Werden Regelveränderungen, die zu Leistungsverbesserungen führen, anhand des Modells erkannt, können diese Veränderungen auch in der Realität umgesetzt werden. Ein dortiger ebenfalls messbarer Leistungsanstieg ist ein starker Nachweis der Modellvalidität.

  Tests zur Systemverbesserung liefern zwar mitunter die wertvollsten Erkenntnisse, da sie zum einen die Validität, zum anderen den Nutzen des Modells zeigen, sind jedoch mit Schwierigkeiten verbunden. Um solche Tests durchzuführen, muss das Modell zunächst auf einem Stand sein, der es erlaubt, nützliche Regeln abzuleiten. Eine solche Regel in der Realität umzusetzen ist wiederum mit einem Risiko verbunden. Sollte das Modell fehlerhaft sein, kann die Anwendung einer aus ihm abgeleiteten Regel das reale System auch verschlechtern.

- **Vorhersage von Verhaltensänderungen.** Dieser Test untersucht, ob das Modell das korrekte Verhalten voraussagt, wenn sich eine entsprechende Regel ändert.

  Um eine entsprechende Validität belegen zu können, stehen verschiedene Techniken zur Verfügung. Eine Möglichkeit ist es, die Regeln im Modell zu ändern und die Plausibilität der dadurch entstehenden Verhaltensänderungen zu prüfen. Das Problem hierbei ist, dass die Plausibilität ohne entsprechende Referenzen nur schwer gezeigt werden kann.

  Als Alternative dazu, kann eine Regeländerung, die im realen System stattfindet, im Modell nachgebildet werden, um die Verhaltensänderung des Modells mit der Verhaltensänderung im realen System zu vergleichen. Existieren Systeme der gleichen Klasse, die jedoch nach

unterschiedlichen Regeln arbeiten, können die jeweiligen Regelkonfigurationen im Modell nachgebildet, und das Verhalten mit dem der verschiedenen realen Systeme verglichen werden.

- **Regelsensitivität.** Dieser Test soll klären, ob die Ableitung einer Regel stark von der Parametrisierung des Modells abhängt. Da eine Parametrisierung oft ungenau ist, weil keine exakten Werte vorliegen oder Einflüsse nur geschätzt werden können, sollte eine abgeleitete Regel nicht nur für bestimmte exakte Parameterwerte gelten, sondern für ganze Wertebereiche gültig sein.

  Um eine solche Gültigkeit zu zeigen, werden die jeweiligen Parameter innerhalb eines entsprechenden Wertebereichs variiert. Die jeweils abgeleiteten Regeln dürfen sich nicht widersprechen.

3.3. Vorgehensweise zur Evaluation des Simulationsmodells

Im Rahmen dieser Arbeit war es nicht möglich, alle der in Abschnitt 3.2 genannten Tests zur Evaluation des Simulationsmodells einzusetzen. Dafür sind vor allem zwei Gründe ausschlaggebend gewesen: Zum einen standen reale Systeme, mit denen ein Vergleich zwischen Modell und Original möglich gewesen wäre, nicht zur Verfügung. Zwar liegen in der Literatur einige Daten über Softwareprojekte vor, die so gefundenen Informationen enthalten jedoch nicht die Messwerte, die für die Validierung des Simulationsmodells notwendig sind. Eine Erhebung eigener, auf das Simulationsmodell bezogener Messwerte an einem realen Projekt war nicht möglich. Zum anderen war der zeitliche Rahmen dieser Arbeit beschränkt.

Die einzige Möglichkeit, Vergleiche mit realen Systemen anzustellen, beschränkt sich auf studentische Projekte. Diese weichen von Projekten, wie sie in der Industrie durchgeführt werden, jedoch meist erheblich ab, da an der Universität der Lernerfolg und nicht das Projektergebnis im Vordergrund steht. So arbeiten die Studierenden beispielsweise in der Regel nicht Vollzeit an einem Projekt, die Aufgabenstellung ist klar vorgegeben, und Änderungen an den Anforderungen während der Projektlaufzeit sind selten.

Projekte aus der Industrie zu betrachten ist daher der bessere Ansatz. Die Laufzeit solcher Projekte beträgt typischerweise mehrere Jahre. Eine entsprechende Begleitung eines solchen Projekts war daher aufgrund des beschränkten Zeitrahmens dieser Arbeit nicht möglich. Eine Alternative dazu wäre eine entsprechende Projektdokumentation gewesen, die den Verlauf eines Projekts detailliert genug beschreibt, um sie anhand des Modells nachvollziehen zu können. Da jedoch nur wenige Projekt so ausführlich dokumentiert sind, und Industrieprojekte und deren Dokumentation von den Unternehmen typischerweise nicht zugänglich gemacht werden, ist ein solcher Vergleich im Rahmen dieser Arbeit nicht möglich gewesen.

Daher entfallen die Tests zur Systemverbesserung und Vorhersage von Verhaltensänderungen im Bereich der Regelimplikation, da diese den Zugriff auf ein reales System voraussetzen. Außerdem entfällt der Test zur Vorhersage von Verhalten, der die Beobachtung eines realen Systems über einen längeren Zeitraum voraussetzt. Um den Bereich der Regelimplikation nicht vollständig zu ignorieren, wurden Regeln und Strategie, die zum Stoffumfang der Softwaretechniklehre gehören, anhand des Modells nachvollzogen. Diese finden sich in den aufgestellten Hypothesen wieder.

Um Struktur- und Parameterverifikation durchzuführen, wurden zwei Ansätze verfolgt. Zum einen wurde ein von Studierenden durchgeführtes Projekt, das im Rahmen eines Praktikums zur Softwaretechnik regelmäßig an der Universität Ulm stattfindet, mit dem Simulationsmodell nachgebildet. Zum anderen wurden ausgewählte Szenarien aus der Literatur mit dem Modell nachgestellt. Die dazugehörige Beschreibung findet sich in Form von Anwendungsbeispielen in Kapitel 4. Das studentische Projekt konnte mit dem Simulationsmodell adäquat abgebildet werden. Auch die Umsetzung der Szenarien aus der Literatur war in den meisten Fällen problemlos möglich.

Die Konsistenz der Modelldimensionen wurde bei der Konstruktion des Modells soweit wie möglich berücksichtigt. Da viele Parameter, sowohl an den Schnittstellen, als auch innerhalb des Simulationsmodells existieren, die im realen System nicht oder nur schwer beobacht- und messbar sind, beispielsweise die Motivation eines Mitarbeiters, gibt es hierfür oft keine etablierte Metrik. Für

entsprechende Parameter wurden eigene Wertebereiche und Metriken definiert, die im Rahmen der entsprechenden Teilmodelle in Kapitel 2 besprochen werden.

Die Modellgrenzen wurden wie in Abschnitt 1.4 beschrieben festgelegt. Das Simulationsmodell orientiert sich an dieser Definition. Im Rahmen der Tests auf die Adäquatheit der Modellgrenzen ist es jedoch sinnvoll zu prüfen, ob diese definierten Grenzen für ein Planspielmodell geeignet sind, d. h. ob essentielle Aspekte fehlen oder Teile des Modells in Planspielen nicht benötigt werden. Die direkteste Methode, um dies zu überprüfen, ist entsprechende Planspiele mit dem Simulationsmodell zu bauen. Im Rahmen dieser Arbeit ist ein Planspiel, das auf dem Simulationsmodell aufbaut, entstanden. Es wird in Kapitel 4.3 im Zuge der Anwendungsbeispiele beschrieben. Auch die in Abschnitt 4.4 beschriebenen weiteren Anwendungsbeispiele weisen darauf hin, dass die Modellgrenzen sinnvoll gewählt wurden. Um noch besser gesicherte Ergebnisse zu bekommen, ist es jedoch notwendig weitere Planspiele mit dem Simulationsmodell zu entwickeln.

Für die Evaluation des Simulationsmodells wurden Effekte und Zusammenhänge identifiziert, die wichtiger Bestandteil der Softwaretechniklehre sind. Die Auswahl dieser Effekte wurde in Abschnitt 3.1 beschrieben. Es hat sich herausgestellt, dass sich diese mit dem Simulationsmodell abbilden lassen, was ein weiterer Hinweis darauf ist, dass die Modellgrenzen gut gewählt wurden.

Der restliche Teil der Evaluation untersucht das Verhalten des Simulationsmodells mittels der in Abschnitt 3.2 dafür beschriebenen Tests. Außer dem Test zur Vorhersage von Verhalten, der ein reales System voraussetzt, das über länger Zeit hin beobachtet werden kann, und dem Vergleich mit empirischen Daten realer Projekte, die nicht zur Verfügung standen, werden alle dort aufgeführten Tests zur Evaluation des Simulationsmodells bezüglich des Modellverhaltens verwendet.

Grundlage für die restlichen Tests bilden die in Abschnitt 3.1 beschriebenen Effekte und Zusammenhänge, und die daraus formulierten Hypothesen. Um diese zu überprüfen, wurde eine Vielzahl an Experimenten durchgeführt. In einem Experiment werden ein oder mehrere auf die jeweilige Hypothese bezogene Szenarien erstellt. Jedes Szenario wird mittels des Simulationsmodells ausgeführt und das Simulationsergebnis, sowie der Simulationsverlauf aufgezeichnet. Anhand dieser Daten kann die entsprechende Hypothese bestätigt oder verworfen werden. Dazu werden die Daten mittels Begutachtung, Korrelationsanalyse und statistischer Hypothesentests ausgewertet.

Grundsätzliche Gestaltung der Experimente

Die Szenarien eines Experiments werden mittels eines Monte-Carlo-Verfahrens [Madachy, 2007] erstellt. Dabei werden die Parameter des Szenarios über eine Zufallsfunktion erzeugt. Variiert werden dabei gegebenenfalls die Struktur, d. h. der Aufbau des Produkts, die Zusammensetzung des Projektteams und das Prozessmodell, als auch die Parameter der jeweiligen Entitäten, wie beispielsweise das Fähigkeitsniveau der Mitarbeiter.

Für die Erzeugung der Testdaten wurden folgende Verfahren eingesetzt:

- **Extremwertuntersuchung.** Um Probleme mit Werten an den Parametergrenzen zu untersuchen, werden die Parameter einzeln oder in Kombination mit ihren Extremwerten belegt. Die durch eine anschließende Ausführung und Aufzeichnung des Simulationsverlaufs erhaltenen Daten können analysiert werden, um sowohl technische, als auch konzeptionelle Probleme im Modell zu identifizieren.

- **Untersuchung eines einzelnen Parameters.** Um den Einfluss der Belegung eines Parameters auf das Simulationsergebnis zu testen, wird ein einzelner Parameter variiert, während die anderen Parameter des Szenarios, sogenannte Rahmenparameter, gleich bleiben. Die Variation des Parameters kann dabei schrittweise oder mittels eine Zufallsfunktion der Gleichverteilung geschehen.

  Um ein belastbares Ergebnis des Experiments zu erhalten, muss sichergestellt werden, dass dieses nicht von der Wahl der Rahmenparameter abhängt. Werden die Rahmenparameter konstant gewählt, ist dies nicht gewährleistet, da so nur eine bestimmte Konfiguration geprüft wird, aus der sich keine Allgemeingültigkeit der Ergebnisse ableiten lässt. Daher müssen auch die Rahmenparameter zufällig gewählt werden. Dabei besteht jedoch die Gefahr, dass beobachtete

Effekte nicht durch die Variation des beobachteten Parameters, sondern der Rahmenparameter zustande kommen. Außerdem verursachen die Rahmenparameter zusätzliche Effekte, die den zu messenden Effekt überlagern und ihn so verzerren.

Um dieses Problem zu lösen, werden zwei Ansätze verwendet. Die Gesamtmenge der Experimente wird in mehrere Sets aufgeteilt, deren Rahmenparameter innerhalb des Sets konstant, und über die Sets hinweg zufällig gewählt werden. Die Parametervariation der so erzeugten Daten hängt damit deutlich stärker vom untersuchten Parameter als von den Rahmenparametern ab. Zudem können die Sets jeweils einzeln ausgewertet, und die Ergebnisse der Auswertung akkumuliert werden. Beim zweiten Ansatz werden die Rahmenparameter zufällig entsprechend der Normalverteilung um den neutralen Wert des jeweiligen Parameters, der im Folgenden als Standardwert bezeichnet wird, gewählt. So entsteht ein jeweils zufälliges Szenario, bei dem Kombinationen extremer Parameterwerte, die meist einen großen Einfluss auf das Verhalten haben, unterdurchschnittlich häufig auftreten.

- **Variation eines Parameterpaars.** Da mittels der Variation eines einzelnen Parameters, auch bei zufälliger Wahl der Rahmenparameter nicht ausgeschlossen werden kann, dass ein bestimmtes Verhalten von der Wechselwirkung zweier Parameter abhängt, ist es sinnvoll anstatt eines Parameters, auch zwei Parameter gleichzeitig zu variieren.

  Die Idee dazu stammt aus dem Ansatz der Pairwise-Methode zum Testen von Softwaresysteme, die auf der Beobachtung basiert, dass die meisten Fehler in Softwaresysteme durch das Zusammenspiel von nur zwei Parametern ausgelöst werden [Flores, 2011]. Übertragen auf das Simulationsmodell bedeutet das, dass durch eine ungeschickte Wahl zweier Parameter, ein fehlerhaftes Verhalten entstehen kann. Um ein solches Verhalten aufzudecken werden daher Parameter in Kombination variiert.

  Führt die Variation eines einzelnen Parameters zu den gleichen Beobachtungen wie die Variation dieses Parameters in Kombination einer Variation eines beliebigen anderen Parameters, ist dies außerdem ein starker Hinweis darauf, dass dieser Parameter alleine für den beobachteten Effekt verantwortlich ist.

- **Vollständig zufällige Belegung aller Parameter.** Um Parameterkombinationen noch intensiver zu testen, können alle Parameter eines Szenarios zufällig gewählt werden. Eine Gleichverteilung der Zufallswerte würde hier jedoch zu viele Kombinationen aus Extremwerten erzeugen. Solche Kombinationen treten in der Realität nur selten auf, und spielen daher weder für Richtlinien und Regeln, noch für Lehrinhalte eine Rolle. Daher werden die Parameterwerte mittels einer Zufallsfunktion basierend auf der Normalverteilung gewählt, um Kombinationen extremer Belegungen zu reduzieren.

  Werden alle Parameter zufällig belegt, sind aufgrund der hohen Variation einige Zusammenhänge und Effekte nicht mehr beobachtbar, da sich diese gegenseitig überlagern. Die so gestalteten Experimente dienen daher nicht dem Nachweis bestimmter Effekte, sondern sollen Effekte aufzeigen, die den Ergebnissen anderer Experimente entgegenstehen.

- Zusätzlich zur Gestaltung der Parameterwahl, haben auch die **Managementstrategien** teilweise eine erhebliche Auswirkung auf den Projektverlauf. Die Gestaltung des Projektplans ermöglicht es, verschiedene Strategien umzusetzen, die typischerweise durch den Projektleiter bestimmt werden. Reicht ein statisch vorgegebener Projektplan nicht aus, können Projektleiter als zusätzliche Agenten umgesetzt werden, die auf den Projektverlauf reagieren und entsprechende Maßnahmen ergreifen.

Um die Evaluation besser handhaben zu können, und eine geeignete Grundlage für eine Automatisierung zu erreichen, wurde eine Liste der relevanten Modellparameter und Messwerte definiert, die gegebenenfalls im Rahmen der Betrachtung konkreter Hypothesen um weitere, hypothesenspezifische Parameter ergänzt wird.

Dabei wird zwischen *Eingabe-* und *Ausgabeparametern* unterschiedene. Eingabeparameter beschreiben Werte, die vom Experiment vorgegeben sind, d. h. sie gelten zu Beginn des Simulationslaufs oder

werden durch das Szenario während des Simulationslaufs sichergestellt. Ausgabeparameter werden während oder am Ende des Simulationslaufs gemessen, oder aus den aufgezeichneten Daten des Simulationsverlaufs berechnet. Einige Parameter werden zwar zu Beginn der Simulationsausführung festgelegt, verändern sich jedoch während des Simulationsverlaufs. Solche Parameter werden als *dynamische Parameter* bezeichnet. Wird die Veränderung eines solchen Parameters betrachtet, wird dies als *Parameterdelta* oder Änderung bezeichnet. Gemessen wird dabei die Differenz zwischen dem Parameterwert vor und nach der Simulationsausführung.

Die genaue Definition der allgemein verwendeten Parameter, und die Vorschriften nach denen sie aus der Simulation extrahiert bzw. berechnet werden, findet sich in Anhang A.4.

Auswertung der Daten

Anhand der durch die Experimente erzeugten Daten können Eigenschaften und Effekte des Simulationsmodells gezeigt werden. Dazu kommen vor allem folgende drei Techniken zum Einsatz:

- Durch eine **Begutachtung** werden die Ergebnisse von Hand analysiert. Anhand der aufgezeichneten Simulationsverläufe und Simulationsergebnisse lässt sich häufig direkt erkennen, ob ein bestimmtes Verhalten vorliegt. Insbesondere können so Verhaltensanomalien und unerwartetes Verhalten erkannt werden. Des Weiteren kann die Parametersensitivität und das Verhalten unter Extrembedingungen qualitativ überprüft werden. Eine Begutachtung kann nicht automatisiert werden, was ein wesentlicher Nachteil dieser Technik ist.

- Mittels einer **Korrelationsanalyse** kann gezeigt werden, dass ein statistischer Zusammenhang zwischen zwei Parametern besteht. Viele der nachzuweisenden Effekte sind als Abhängigkeit zwischen Parametern formuliert, wie beispielsweise: Mitarbeiter mit besseren Fähigkeiten sind produktiver. Eine solche Aussage lässt sich überprüfen, indem eine positive Korrelation zwischen dem Parameter für die Mitarbeiterfähigkeit und der gemessenen Produktivität gezeigt wird.

  Um die Korrelation zweier Parameter aus den Messdaten zu ermitteln, wird der Rangkorrelationskoeffizient nach Spearman berechnet. Dieser hat den Vorteil, dass er keine lineare Beziehung zwischen den Variablen, und keine Normalverteilung der Daten voraussetzt, und außerdem robust gegen Ausreißer ist. Das Signifikanzniveau ist auf $p \leq 0.05$ festgelegt. Da die Signifikanz eines Ergebnisses maßgeblich von der Stichprobengröße abhängt, und diese durch die Anzahl der Simulationsläufe eines Experiments beliebig erhöht werden kann, weisen die meisten Experimente eine Signifikanz von annähernd 0 auf. Werte unterhalb von 0.001 werden im Folgenden als $p \ll 0.05$ angegeben. Wichtiger ist deshalb die Bestimmung der Effektstärke in Form des Korrelationskoeffizienten, der angibt, wie eindeutig der Zusammenhang zwischen den Parametern ist. Nach [Cohen, 1988] lassen sich Korrelationen anhand ihres Korrelationskoeffizienten r in die Klassen *schwacher* ($r \approx 0.1$), *mittlerer* ($r \approx 0.3$), und *starker* ($r \approx 0.5$) Zusammenhang einordnen. Diese Einteilung wird übernommen und streng ausgelegt, sodass der eine Korrelation erst als schwach gilt, wenn ihr Koeffizient mindestens 0.1 beträgt, ein mittlerer Zusammenhang ab einem Korrelationskoeffizienten von mindestens 0.3 gilt, und ein starker Zusammenhang einen Koeffizienten von mindestens 0.5 verlangt.

  Korrelationsanalysen können automatisiert auf den Messdaten berechnet und ausgewertet werden. Da die zu prüfenden Hypothesen bei den Experimenten hinterlegt sind, kann automatisch entschieden werden, ob diese angenommen oder abgelehnt werden müssen. Dadurch werden Regressionstests, die bei Änderungen am Modell durchgeführt werden müssen, deutlich einfacher.

  Einige Experimente werden in mehreren Sets durchgeführt, wobei innerhalb eines Sets nur ein einzelner Parameter verändert wird, während alle anderen Parameter gleichbleiben. So lässt sich der Effekt besser auf den einzigen veränderten Parameter einschränken. Typischerweise werden mehrere Sets berechnet, um zu zeigen, dass der Effekt unabhängig von der Wahl der nicht veränderten Parameter ist. Um für mehrere Sets einen gemeinsamen Korrelationswert zu erhalten, wird die Korrelation der Sets einzeln berechnet, und anschließend das Mittel aus allen signifikanten Ergebnissen berechnet. Der Ausschluss nicht signifikanter Ergebnisse ist hierbei

notwendig, da durch die automatisierte Erstellung von Szenarien Situationen entstehen können, in denen Projekte nicht korrekt funktionieren, und die so in der Realität nicht auftreten, da der Projektleiter in einer solchen Situation passende Veränderungen vornehmen würde. Werden Koeffizienten solcher Projekte in das Ergebnis miteinbezogen, kann es dadurch erheblich verfälscht werden. Das ist beispielsweise der Fall, wenn Mitarbeiter nicht die erforderlichen Fähigkeiten besitzen, um Fortschritt erzeugen zu können, oder wenn Defekte aufgedeckt werden sollen, obwohl keine im Produkt vorhanden sind.

Durch den Ausschluss nicht signifikanter Ergebnisse können so theoretisch Korrelationswerte entstehen, die nur auf einem oder wenigen Sets basieren, und somit keine hohe Aussagekraft haben. Entsprechend ist es wichtig, das Verhältnis zwischen der Gesamtzahl und der Anzahl an ausgeschlossenen Sets mitzuberücksichtigen.

- In einigen Fällen wird ein **Vergleich** zwischen zwei Klassen von Szenarien benötigt. Beispielsweise wird untersucht, ob ein Tausch der Aufgaben zweier Mitarbeiter zu einem bestimmten Zeitpunkt die Produktivität beeinträchtigt. Dazu werden mehrere Szenarien geschaffen, und diese in die jeweiligen Klassen eingeordnet. Die Simulationsergebnisse beider Klassen werden anschließend mit dem Wilcoxon-Test verglichen, um zu ermitteln, ob signifikante Unterschiede bestehen. Dieser Test hat gegenüber dem t-Test den Vorteil, dass er keine Normalverteilung der Daten fordert. Als Kriterium für einen signifikanten Unterschied wird $p \leq 0.05$ angenommen. Als Effektstärke wird die Differenz der Mittelwerte beider Gruppen bei unabhängigen Stichproben, bzw. das Mittel der paarweisen Differenzen bei paarweisen Stichproben verwendet. Wie auch Korrelationsanalysen lassen sich so durchgeführte Vergleiche leicht automatisieren.

3.4. Durchgeführte Experimente und Ergebnisse

In diesem Abschnitt werden die im Rahmen der Evaluation des Simulationsmodells durchgeführten Experimente, deren Auswertung und die erhaltenen Ergebnisse vorgestellt. Neben der ständigen Begutachtung der entstandenen Simulationsverläufe während der Entwicklung des Simulationsmodells und der Erstellung der Beispielszenarien, die unter anderem in Kapitel 4 beschrieben sind, konzentriert sich die Evaluation auf die in Abschnitt 3.1 genannten Hypothesen. Zusätzlich zu den Experimenten, die im Rahmen der Hypothesenevaluation verwendet werden, wurden weitere Experimente durchgeführt, um zusätzliche Messdaten mit dem Simulationsmodell zu generieren. Diese werden in Abschnitt 3.4.2 beschrieben. Dadurch werden auch Bereiche berücksichtigt, die durch die Hypothesen nicht abgedeckt sind. Vorrangiges Ziel der weiteren Experimente ist es, Anomalien und Unstimmigkeiten aufzudecken. Dazu werden auch die Ergebnisse aller Experimente auf Widersprüche geprüft.

Im Folgenden wird häufig auf Standardwerte für Szenarioparameter verwiesen. Eine Auflistung dieser Werte findet sich in Anhang A.1.

3.4.1. Prüfung der aufgestellten Hypothesen

Im Folgenden wird die Prüfung der oben gelisteten Hypothesen im Detail beschrieben. Dabei werden sowohl die Experimente erläutert, die zur Generierung der Daten verwendet werden, als auch die Auswertung dieser Daten.

Die meisten Hypothesen werden anhand zweier Experimente untersucht. Davon wird ein Experiment in Sets durchgeführt. Dabei wird der zu untersuchende Parameter innerhalb eines Sets variiert, während alle andere Parameter gleichbleiben. Es werden typischerweise mehrere Sets berechnet, um auszuschließen, dass die Rahmenparameter des Sets zufälligerweise zu dem jeweiligen Ergebnis führen. Mit dieser Art Experimente kann der Einfluss des Parameters direkt gezeigt werden. Im zweiten Experiment werden alle Parameter in jedem Simulationslauf neu gewählt. Mit dieser Art von Experimenten lässt sich zum einen zeigen, dass die Ergebnisse der Experimente im Set nicht zufälligerweise von der Wahl der Setparameter abhängen. Zum anderen lässt sich der Einfluss des untersuchten Parameters im Verhältnis zu den anderen Parametern betrachten, und so seine Einflussstärke besser einschätzen.

Hypothese 1: Zusammenhang zwischen Fähigkeit und Produktivität

Je höher die Fähigkeiten eines Mitarbeiters für eine Aktivität sind, umso produktiver ist er dabei.

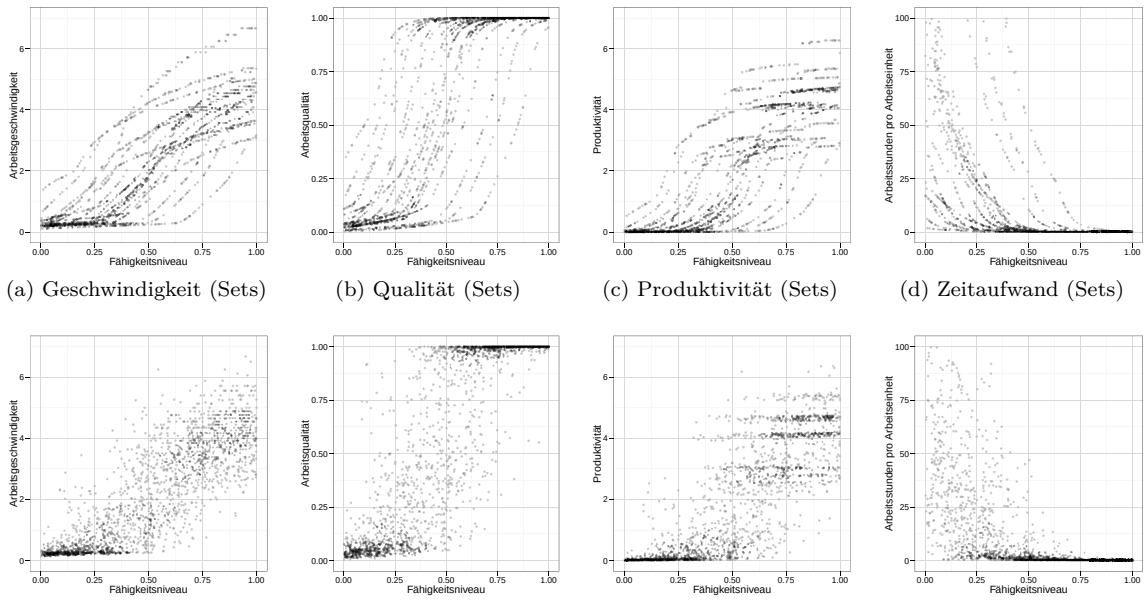
Um diese Hypothese, die sich häufig in der Literatur findet, wie beispielsweise in [Boehm, 1981], akzeptieren zu können, muss gezeigt werden, dass ein positiver Zusammenhang zwischen den Fähigkeiten eines Mitarbeiters und seiner Arbeitsleistung besteht. Erwartet wird eine positive Korrelation, sowohl zwischen Fähigkeit und Arbeitsgeschwindigkeit, als auch zwischen Fähigkeit und Qualität der Arbeitsergebnisse.

Konzeption der Experimente Das dazugehörige Experiment untersucht Projekte, deren Parameter zufällig gemäß der Normalverteilung um ihren Standardwert gewählt werden. Betrachtet wird ein einzelner Mitarbeiter, der ein einzelnes Artefakt bearbeitet. Gemessen werden:

- der durchschnittliche am Artefakt erzeugte Fortschritt pro Stunde,
- die durchschnittliche Qualität des erzeugten Fortschritts,
- die durchschnittliche Produktivität, als Quotient des geschaffenen Fortschritts, abzüglich der erzeugten Defekte, und der dafür benötigen Zeit,
- die benötigten Arbeitsstunden pro fertiggestellter Arbeitseinheit.

Die Hypothese ist zu akzeptieren, wenn für erzeugten Fortschritt, dessen Qualität und die Produktivität eine positive Korrelation, und für die benötigten Arbeitsstunden eine negative Korrelation, jeweils in Bezug auf die Fähigkeit des Mitarbeiters gezeigt werden kann.

Dazu werden zwei Experimente mit jeweils 2.000 Simulationsläufe berechnet. Das erste Experiment besteht aus 20 Sets mit jeweils 100 Simulationsläufen. Innerhalb eines Sets werden nur die Fähigkeit des Mitarbeiters bezüglich der durchzuführenden Aktivität verändert, während alle anderen Parameter des Projekts beibehalten werden. Diese werden je Set zufällig gemäß einer Gleichverteilung gewählt. Im zweiten Experiment werden auch alle anderen Projektparameter in jedem Simulationslauf zufällig neu gewählt, wobei hier die Normalverteilung um den jeweiligen Standardwert des Parameters verwendet wird.



(a) Geschwindigkeit (Sets) (b) Qualität (Sets) (c) Produktivität (Sets) (d) Zeitaufwand (Sets)

(e) Geschwindigkeit (alle unabhängig) (f) Qualität (alle unabhängig) (g) Produktivität (alle unabhängig) (h) Zeitaufwand (alle unabhängig)

Abbildung 3.3.: Zusammenhang zwischen Fähigkeitsniveau und Produktivität

Ergebnisse Abbildung 3.3 zeigt eine Visualisierung der Messergebnisse. Tabelle 3.1 enthält die Ergebnisse der Korrelationsanalyse. Sowohl die mittels Sets erzeugten Daten, als auch die Daten aus dem Experiment mit den unabhängigen Szenarien, weisen die geforderten signifikanten Korrelationskoeffizienten auf. Die Hypothese ist somit zu akzeptieren.

	Sets	alle unabhängig
Geschwindigkeit	$+0.79$	$+0.82$
Qualität	$+0.78$	$+0.84$
Produktivität	$+0.79$	$+0.81$
Zeitaufwand	-0.64	-0.68

Tabelle 3.1.: Korrelationskoeffizienten der Messergebnisse zu Hypothese 1

Hypothese 2: Zusammenhang zwischen Artefaktwissen und Produktivität

Je größer das Artefaktwissen eines Mitarbeiters ist, umso höhere ist seine Produktivität.

Um effektiv arbeiten zu können, müssen die Mitarbeiter mit dem Produkt, an dem sie arbeiten, vertraut sein [Abdel-Hamid, 1991]. Das Simulationsmodell beschreibt diese Vertrautheit mit dem Produktwissen, das sich wiederum aus dem Wissen zu den einzelnen Artefakten des Produkts zusammensetzt. Je höher dieses Artefaktwissen eines Mitarbeiters ist, umso höher muss auch seine Produktivität sein. Wie stark sich das Produktwissen auf die Produktivität auswirkt, ist dabei vor allem von der durchgeführten Aktivität abhängig. Für eine Inspektion ist beispielsweise ein detaillierteres Wissen über den Prüfling notwendig, als es bei einem Black-Box-Test der Fall ist, bei dem lediglich die Schnittstellen relevant sind.

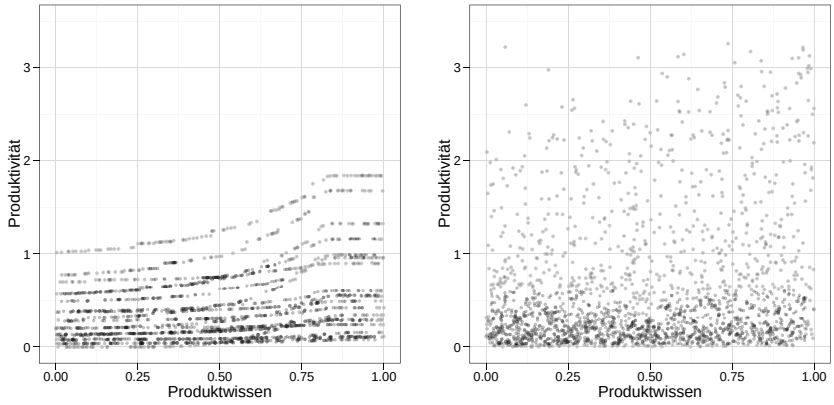
Konzeption der Experimente Um zu zeigen, dass das Simulationsmodell diesen Effekt korrekt abbildet, werden zwei Experimente durchgeführt. Dabei wird die Produktivität eines Mitarbeiters, der an einem Artefakt arbeitet gemessen. Das zu bearbeitende Artefakt weist zu Beginn der Simulation eine relative Vollständigkeit zwischen 0 und 1 auf, die zufällig entsprechend der Normalverteilung um einen Erwartungswert von 0.5, und mit einer Standardabweichung von 0.2 gewählt wird. Das relative Artefaktwissen des Mitarbeiters über den schon vorhandenen Artefaktteil wird zufällig gemäß einer Gleichverteilung auf dem Intervall $[0, 1]$ gewählt. Die restlichen Rahmenparameter des Szenarios werden zufällig gemäß der Normalverteilung um ihren jeweiligen Standardwert belegt.

Das erste Experiment wird in 20 Sets à 100 Simulationsläufen durchgeführt, wobei innerhalb eines Sets lediglich das Artefaktwissen variiert wird. Das zweite Experiment enthält 2.000 voneinander unabhängige Simulationsläufe. Hier werden die Rahmenparameter in jedem Simulationslauf zufällig neu gewählt.

Die Hypothese ist zu akzeptieren, wenn in beiden Experimenten eine positive Korrelation zwischen Artefaktwissen und Produktivität gezeigt werden kann.

Ergebnisse Abbildung 3.4 zeigt die Ergebnisse der beiden Experimente. In beiden Experimenten ist eine positive Korrelation zu erkennen. Für die Sets beträgt der Rangkorrelationskoeffizient $r = +0.99$, für die unabhängigen Durchläufe $r = +0.18$. Beide Korrelationen sind mit $p \ll 0.05$ signifikant. Die Hypothese ist entsprechend zu akzeptieren.

Es ist auffällig, dass sich die Korrelationskoeffizienten der beiden Experimente stark unterscheiden. Während die einzelnen Sets keinen Störgrößen unterliegen, verzerren diese das Ergebnis bei den unabhängigen Durchläufen erheblich. Grund dafür ist, dass die Produktivität nicht nur vom Produktwissen, sondern auch von weiteren Parametern, wie beispielsweise den Fähigkeiten des Bearbeiters abhängt. Die unabhängigen Durchläufe dienen deshalb vor allem dazu, die Evaluation gegen falsche Ergebnisse, die durch eine schlechte Wahl der Sets entstehen, abzusichern. Widersprüche zwischen den unterschiedlichen Experimenten würden auf ein solches Problem hinweisen.



(a) Sets (b) alle unabhängig

Abbildung 3.4.: Produktivität in Abhängigkeit des Produktwissens

Hypothese 3: Zusammenhang zwischen Schwierigkeit und Produktivität

Je höher die Schwierigkeit eines Artefakts ist, umso geringer ist die Produktivität der daran arbeitenden Mitarbeiter.

Im Simulationsmodell ergibt sich die Schwierigkeit einer Aufgabe aus der im dazugehörigen Artefakt angegeben Schwierigkeit. Durch diesen Wert kann beispielsweise die Komplexität des Artefakts dargestellt werden. Die Schwierigkeit einer Aufgabe hat direkte Auswirkungen auf Geschwindigkeit und Qualität, mit der sie durchgeführt werden kann. Somit ist für Artefakte mit höherem Schwierigkeitsgrad eine niedrigere Produktivität bei deren Erstellung zu erwarten.

Konzeption der Experimente Um dies zu zeigen, werden zwei Experimente konzipiert, bei denen ein Mitarbeiter ein Artefakt bearbeitet. Die Schwierigkeit dieses Artefakts wird variiert. Im ersten Experiment werden 20 Sets à 100 Simulationsläufe berechnet. Innerhalb eines Sets wird die Schwierigkeit zufällig gemäß einer Gleichverteilung aus dem Intervall $[0, 1]$ gewählt, während alle anderen Parameter gleichbleiben. Diese Rahmenparameter werden einmal pro Set zufällig gemäß der Normalverteilung um ihren Standardwert gewählt. Im zweiten Experiment werden 2.000 Simulationsläufe berechnet, wobei in jedem Simulationslauf alle Parameter neu gewählt werden.

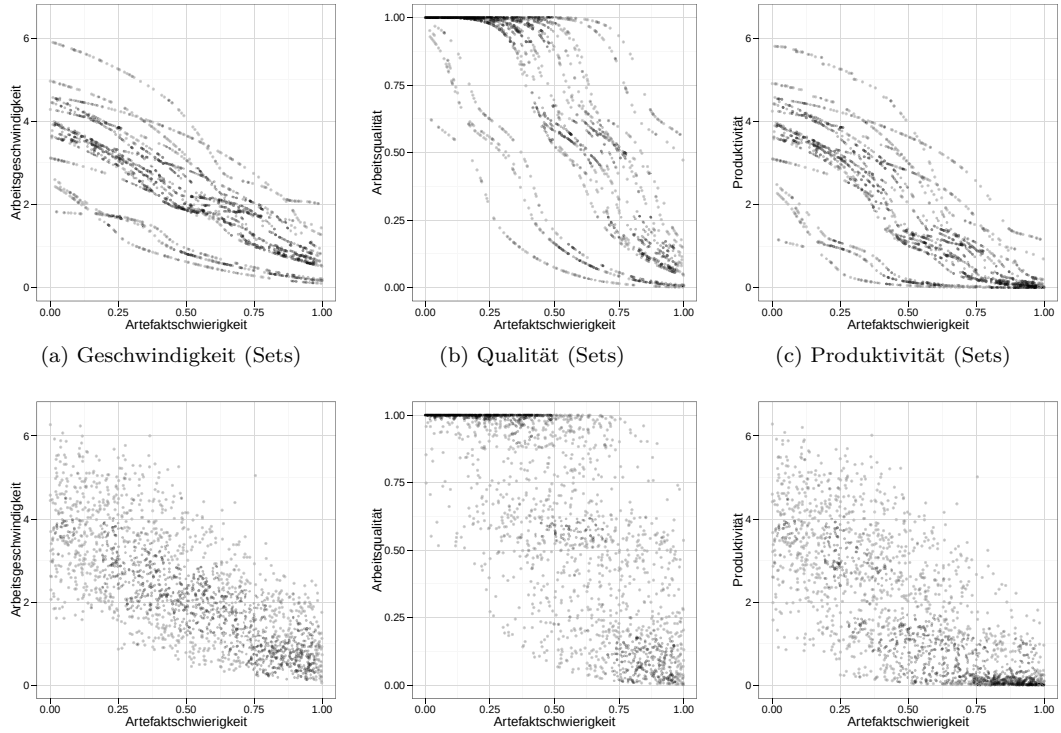
Gemessen wird die Arbeitsgeschwindigkeit, die Qualität des Arbeitsergebnisses und die Produktivität als Quotient aus geschaffenem Artefaktumfang und der dafür benötigten Zeit.

Die Hypothese ist zu akzeptieren, wenn in beiden Experimenten zwischen Artefaktschwierigkeit und den gemessenen Parametern eine negative Korrelation nachgewiesen werden kann.

Ergebnisse Abbildung 3.5 zeigt die Messergebnisse der Experimente. In allen 6 Messungen ist eine negative Korrelation zu erkennen. Tabelle 3.2 enthält die dazugehörigen Rangkorrelationskoeffizienten, die sich mit einer Signifikanz von $p \ll 0.05$ aus den Messdaten der Experimente ergeben.

	Geschwindigkeit	Qualität	Produktivität
Sets	-0.99	-0.99	-0.99
alle unabhängig	-0.76	-0.79	-0.77

Tabelle 3.2.: Korrelationskoeffizienten der Messergebnisse zu Hypothese 3



(a) Geschwindigkeit (Sets) (b) Qualität (Sets) (c) Produktivität (Sets)

(d) Geschwindigkeit (alle unabhängig) (e) Qualität (alle unabhängig) (f) Produktivität (alle unabhängig)

Abbildung 3.5.: Produktivität in Abhängigkeit der Artefaktschwierigkeit

Hypothese 4: Zusammenhang zwischen Produktivität und Projektdauer

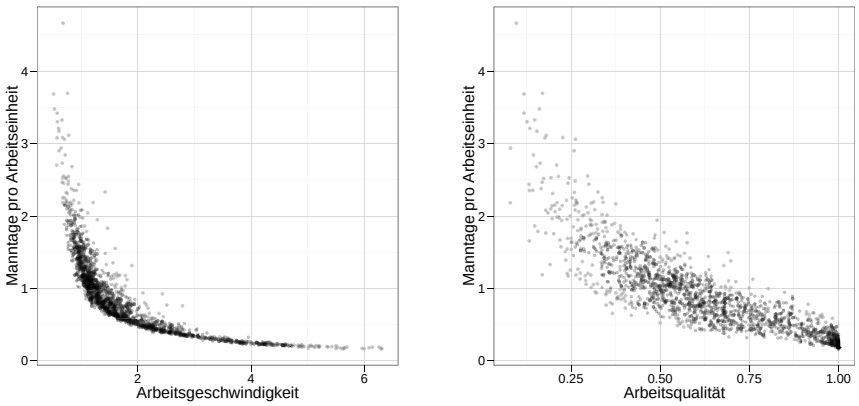
Je höher die Produktivität der Mitarbeiter ist, umso geringer ist die benötigte Zeit für die Fertigstellung des Projekts.

Die in dieser Hypothese beschriebene Aussage ist trivial, da sich die Produktivität als geschaffener Projektumfang pro benötigte Zeit darstellen lässt. Umso wichtiger ist es, dass das Simulationsmodell diesen Effekt korrekt darstellt. Um dies zu zeigen, wird der Zusammenhang zwischen Arbeitsgeschwindigkeit bzw. Arbeitsqualität und der benötigten Zeit für die Erstellung des Produkts untersucht. Eine Diskrepanz zwischen Arbeitsleistung und Projektdauer könnte beispielsweise entstehen, wenn Mitarbeiter den durch die höhere Arbeitsleistung erhaltenen Zeitvorteil in projektfremde Tätigkeiten investieren. Auch Fehler in der Gestaltung und Umsetzung des Simulationsmodells, insbesondere der Verhaltensmodelle, könnten zu einem entsprechenden fehlerhaften Verhalten führen.

Konzeption der Experimente Um ein fehlerhaftes Verhalten auszuschließen, wird ein einzelnes Experiment durchgeführt. Da der Zusammenhang in diesem Fall sehr klar ist, werden keine Sets berechnet. Das Experiment wird in 2.000 unabhängigen Simulationsläufen durchgeführt, bei denen alle Parameter zufällig gemäß der Normalverteilung um ihren Standardwert für jeden Simulationslauf neu gewählt werden. Die Größe des Teams wird zufällig zwischen 1 und 5 Projektbeteiligten gewählt. Die Anzahl der zu bearbeitenden Artefakte wird in Abhängigkeit der Teamgröße gewählt.

Gemessen wird die durchschnittliche Produktivität bei der Aufgabenbearbeitung in Form von Arbeitsgeschwindigkeit und Arbeitsqualität, sowie die Projektlaufzeit pro Arbeitseinheit und Mitarbeiter. Eine solche Umrechnung ist notwendig, da sowohl der zu entwickelnde Produktumfang als auch die Größe des Teams zufällig gewählt wird, und das beim Ergebnis zu berücksichtigen ist.

Ergebnisse Abbildung 3.6 zeigt die Ergebnisse der Messungen. Wird Produktivität als geschaffener Umfang pro Zeiteinheit interpretiert, ergibt daraus eine Hyperbelfunktion mit benötigte Zeit = $^{\text{Arbeitsumfang}}/_{\text{Produktivität}}$. Diese Funktion ist in beiden Schaubildern erkennbar, ein entsprechender Zusammenhang zwischen Produktivität und Projektdauer ist somit gegeben. Der dazugehörige Rangkorrelationskoeffizient von $r = -0.95$ für die Abhängigkeit zwischen Arbeitsgeschwindigkeit und Projektlaufzeit, und $r = -0.89$ für die Abhängigkeit zwischen Arbeitsqualität und Projektlaufzeit, beide mit einer Signifikanz von $p \ll 0.05$, bestätigen den in der Hypothese formulierten Zusammenhang. Die Hypothese ist somit zu akzeptieren.



(a) Geschwindigkeit (b) Qualität

Abbildung 3.6.: Zusammenhang zwischen Produktivität und Projektlaufzeit

Hypothese 5: Zusammenhang zwischen Aufgabensympathie und Produktivität

Je höher die Sympathie eines Mitarbeiters für eine Aufgabe ist, umso höher ist seine Produktivität bei der Bearbeitung dieser Aufgabe.

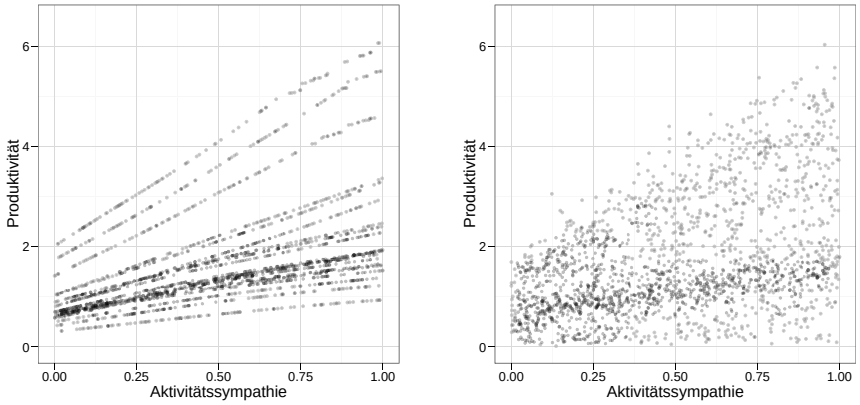
Mitarbeiter, die Aufgaben bearbeiten, die ihnen Spaß machen und die zu ihren Interessen passen, sind deutlich produktiver als Mitarbeiter deren Aufgaben nicht zu ihrer Motivation passen [Boehm, 1981]. Das Simulationsmodell bildet Spaß, Interesse und Motivation für eine Aufgabe über die Aktivitätssympathie/-antipathie ab. Entsprechend ist zu zeigen, dass eine höhere Sympathie zu einer höheren Produktivität führt.

Konzeption der Experimente Um das zu untersuchen, werden zwei Experimente durchgeführt. Die Rahmenparameter des jeweiligen Szenarios werden zufällig gemäß der Normalverteilung um ihren Standardwert gewählt. Ein Experiment wird als 20 Sets à 100 Durchläufe konzipiert, hier werden die Rahmenparameter einmal pro Set neu gewählt. Das zweite Experiment wird als 2.000 unabhängige Durchläufe konzipiert; hier werden die Rahmenparameter in jedem Durchlauf neu gewählt. Die Aktivitätssympathie des beobachteten Mitarbeiters wird gemäß der Gleichverteilung zufällig in jedem Durchlauf aus dem Intervall $[0, 1]$ neu gewählt.

Gemessen wird die Produktivität als Quotient aus dem geschaffenen defektfreien Fortschritt und der dafür benötigten Zeit.

Die Hypothese ist zu akzeptieren, wenn gezeigt werden kann, dass eine positive Korrelation zwischen Aktivitätssympathie und Produktivität besteht.

Ergebnisse Abbildung 3.7 zeigt die Ergebnisse der Experimente. Bei allen Messungen ist ein positiver Zusammenhang zwischen Sympathie und Produktivität zu erkennen. Die dazugehörigen Korrelationskoeffizienten betragen $r = +1.00$ (Sets) und $r = +0.41$ (alle unabhängig) und sind mit $p \ll 0.05$ signifikant. Die Hypothese ist damit zu akzeptieren.



(a) Sets (b) alle unabhängig

Abbildung 3.7.: Produktivität in Abhängigkeit der Aktivitätssympathie

Hypothese 6: Zusammenhang zwischen Adaptionsfähigkeit und Produktivität

Je höher die Adaptionsfähigkeit eines Mitarbeiters ist, umso produktiver ist er.

Die Adaptionsfähigkeit gibt an, wie schnell ein Mitarbeiter neues Wissen aufnehmen, verarbeiten und speichern kann. Nach [Abdel-Hamid, 1991] ist zu erwarten, dass Mitarbeiter mit einer höheren Aufnahmefähigkeit produktiver sind, als Mitarbeiter, die Schwierigkeiten mit der Adaption neuen Wissens haben.

Konzeption der Experimente Um den Zusammenhang zwischen Adaptionsfähigkeit und Produktivität zu zeigen, wird die Erstellung eines Artefakts betrachtet. Das daran arbeitende Team besteht aus 2 bis 5 Mitarbeitern. Die Teamgröße, sowie die Belegung der restlichen Parameter des Szenarios werden zufällig gewählt. Für die Wahl der Teamgröße wird dabei eine Gleichverteilung verwendet, alle anderen Parameter werden entsprechend der Normalverteilung um ihren Standardwert gewählt.

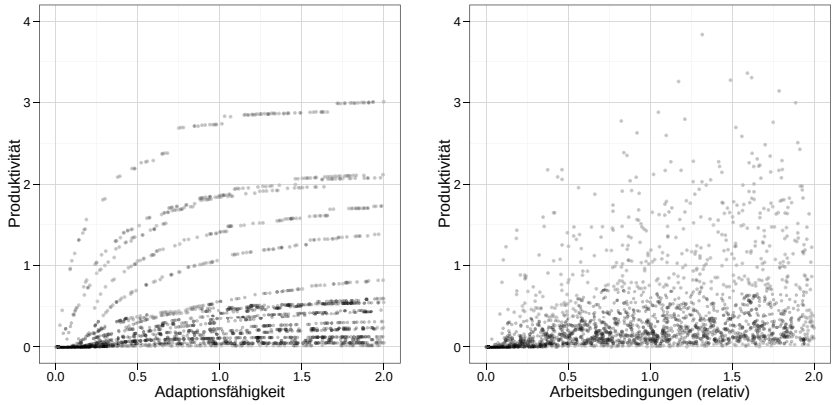
Es werden zwei Experimente durchgeführt. Das erste Experiment besteht aus 20 Sets mit jeweils 30 Simulationsläufen. Hier werden die Rahmenparameter pro Set gewählt. Innerhalb der Sets wird die Adaptionsfähigkeit entsprechend einer Gleichverteilung zufällig zwischen 0 und 2 in jedem Simulationslauf neu gewählt. Im zweiten Experiment werden 2.000 unabhängige Simulationsläufe berechnet, wobei sowohl alle Rahmenparameter als auch die Adaptionsfähigkeit für jeden Simulationslauf neu gewählt werden.

In beiden Experimenten wird die Produktivität in Form des pro Stunde erzeugten Fortschritts, abzüglich des Umfangs der dabei erzeugten Defekte, pro Mitarbeiter gemessen.

Die Hypothese wird akzeptiert, wenn in beiden Experimenten eine signifikant positive Korrelation zwischen Adaptionsfähigkeit und Produktivität gezeigt werden kann.

Ergebnisse Abbildung 3.8 zeigt eine Visualisierung der Messergebnisse. In beiden Experimenten ist eine positive Korrelation zwischen der Adaptionsfähigkeit und der Produktivität zu erkennen. Der Korrelationskoeffizient der Sets beträgt im Durchschnitt $r = +0.99$ bei einer Signifikanz von $p \ll 0.05$, was zeigt, dass eine höhere Adaptionsrate zu einer höheren Produktivität führt. Die Stärke der Zunahme hingegen ist auch von den Rahmenparametern abhängig, wie sich anhand von Abbildung 3.8a erkennen lässt.

Der Korrelationskoeffizient im zweiten Experiment beträgt $r = +0.43$ bei $p \ll 0.05$. Dies zeigt, dass mindestens ein Rahmenparameter einen deutlich höheren Einfluss auf die Produktivität hat. Werden sowohl der Schwierigkeitsgrad des Artefakts, als auch die Fähigkeiten der Mitarbeiter von der Variation der Parameter ausgeschlossen, so lässt sich auch im zweiten Experiment ein Korrelationskoeffizient von $r > 0.8$ nachweisen. Das zeigt, dass vor allem Schwierigkeitsniveau und Fähigkeiten die Produktivität stärker beeinflussen als es die Adaptionsrate tut.



(a) Sets (b) alle unabhängig

Abbildung 3.8.: Zusammenhang zwischen Adaptionsfähigkeit und Produktivität

Hypothese 7: Einfluss der Voraussetzungen auf die Produktivität

Je besser die Voraussetzungen für einen Arbeitsschritt sind, umso höher ist die Produktivität in diesem Arbeitsschritt.

Die Voraussetzungen für einen Arbeitsschritt, werden im Simulationsmodell durch die Kontextartefakte des zu bearbeitenden Artefakts modelliert. Weitere Voraussetzungen müssen gegebenenfalls von den Mitarbeitern selbst mitgebracht werden. Dazu gehören vor allem deren Fähigkeiten. Diese Eigenschaften der Mitarbeiter werden durch die Hypothesen H1, H2, H5 und H6 betrachtet. Im Rahmen dieser Hypothese werden lediglich die Kontextartefakte als Voraussetzungen untersucht.

Die durch die Kontextartefakte bereitgestellten Voraussetzungen ergeben sich gleichermaßen aus deren Qualität und Vollständigkeit. Der Einfluss auf die Bearbeitung des Artefakts wird über das in der Kontextverknüpfung angegebene Gewicht gesteuert.

Konzeption der Experimente Um zu zeigen, dass die Produktivität von den Kontextartefakten abhängt, werden vier Experimente konzipiert. Dabei wird ein einzelner Mitarbeiter betrachtet, der ein einzelnes Artefakt bearbeitet, das im Kontext eines weiteren Artefakts steht. Die Kontextverknüpfung hat ein Gewicht von 0.5.

Zwei Experimente berechnen 20 Sets zu je 100 Durchläufen. Ein Experiment variiert dabei die Qualität des Kontextartefakts, das andere dessen relative Vollständigkeit. Beide Eigenschaften werden dabei in jedem Durchlauf zufällig entsprechend einer Gleichverteilung aus dem Wertebereich $[0, 1]$ gewählt. Alle anderen Parameter werden für jedes Set zufällig gemäß der Normalverteilung um ihren Standardwert gewählt. Zwei weitere Experimente sind gleich konzipiert, jedoch werden hier alle Parameter in jedem Durchlauf neu gewählt.

Gemessen werden die Arbeitsleistung d. h. Arbeitsgeschwindigkeit und Arbeitsqualität, sowie die Produktivität als Quotient aus geschaffenem Artefaktumfang und der dafür benötigten Zeit.

Die Hypothese ist zu akzeptieren, wenn für alle gemessenen Eigenschaften eine signifikante positive Korrelation zwischen der Kontextqualität bzw. der Kontextvollständigkeit und der jeweiligen Eigenschaft gezeigt werden kann.

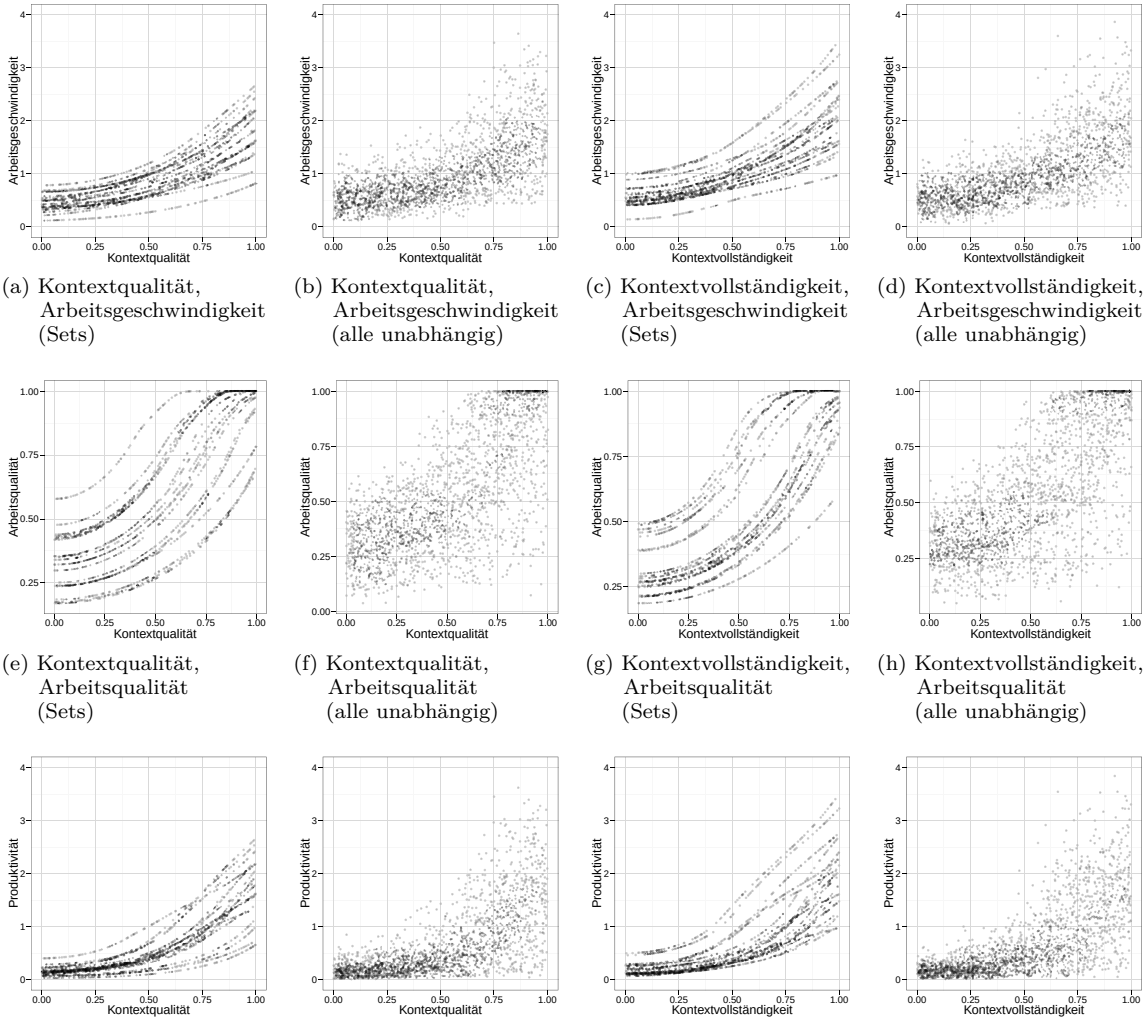
Ergebnisse Abbildung 3.9 zeigt die Ergebnisse der Experimente. Eine Spalte enthält jeweils die drei Messungen zu einem Experiment. Bei allen Messungen ist die geforderte positive Korrelation zu erkennen. Die dazugehörigen Koeffizienten der Rangkorrelation sind in Tabelle 3.3 aufgeführt. Für alle Korrelationskoeffizienten gilt $p \ll 0.05$. Die Hypothese ist somit zu akzeptieren.

Bei den Ergebnissen ist auffällig, dass sich die Kontextqualität und Kontextvollständigkeit sehr ähnlich verhalten. Die zu erkennenden Unterschiede sind auf die zufällig gewählten Rahmenparameter

zurückzuführen. Es macht keinen Unterschied, ob ein Teil des Kontextartefakts fehlt, oder ob dieser Teil in einer unbrauchbaren Qualität vorliegt.

	Kontextqualität		Kontextvollständigkeit	
	Sets	alle unabhängig	Sets	alle unabhängig
Geschwindigkeit	+0.99	+0.69	+0.99	+0.70
Qualität	+0.99	+0.68	+0.99	+0.68
Produktivität	+0.99	+0.72	+0.99	+0.72

Tabelle 3.3.: Korrelationskoeffizienten der Messergebnisse zu Hypothese 7



(a) Kontextqualität, Arbeitsgeschwindigkeit (Sets)
(b) Kontextqualität, Arbeitsgeschwindigkeit (alle unabhängig)
(c) Kontextvollständigkeit, Arbeitsgeschwindigkeit (Sets)
(d) Kontextvollständigkeit, Arbeitsgeschwindigkeit (alle unabhängig)
(e) Kontextqualität, Arbeitsqualität (Sets)
(f) Kontextqualität, Arbeitsqualität (alle unabhängig)
(g) Kontextvollständigkeit, Arbeitsqualität (Sets)
(h) Kontextvollständigkeit, Arbeitsqualität (alle unabhängig)
(i) Kontextqualität, Produktivität (Sets)
(j) Kontextqualität, Produktivität (alle unabhängig)
(k) Kontextvollständigkeit, Produktivität (Sets)
(l) Kontextvollständigkeit, Produktivität (alle unabhängig)

Abbildung 3.9.: Arbeitsgeschwindigkeit, Arbeitsqualität und Produktivität in Abhängigkeit von Kontextqualität und Kontextvollständigkeit

Hypothese 8: Zusammenhang zwischen Motivation und Produktivität

Je höher die Motivation eines Mitarbeiters ist, umso höher ist seine Produktivität.

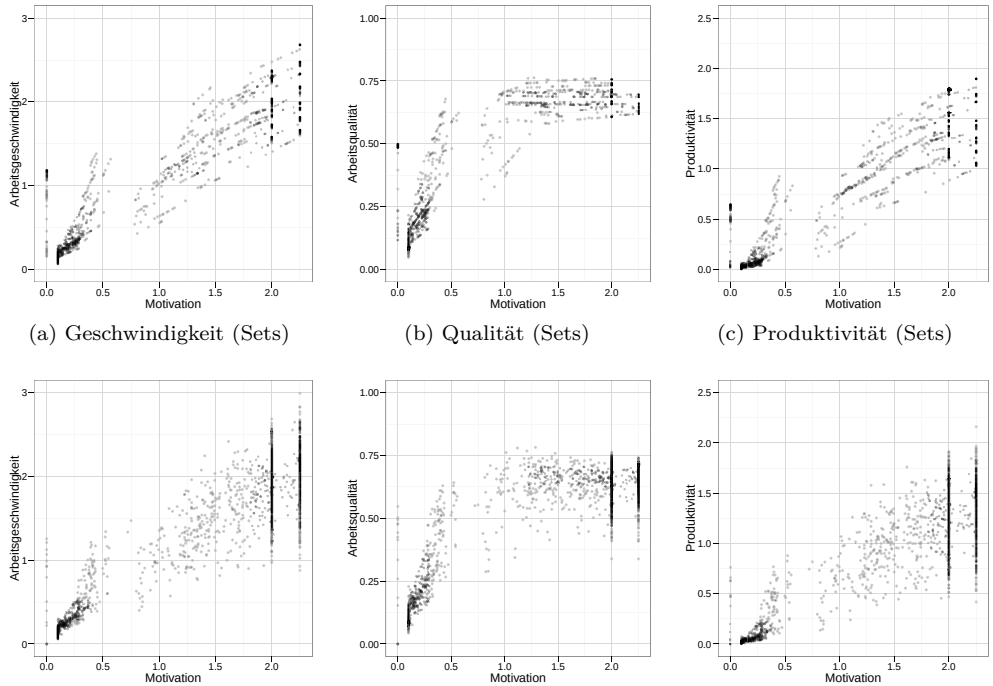
Die Motivation hat einen erheblichen Einfluss auf die Produktivität der Mitarbeiter [Boehm, 1981] [Tvedt, 1996]. Entsprechend wichtig ist es, dass das Simulationsmodell diesen Aspekt korrekt umsetzt.

Konzeption der Experimente Um zu zeigen, dass die Produktivität von der Motivation abhängt, wird ein Mitarbeiter beobachtet, der ein Artefakt bearbeitet. Dabei werden die direkte Arbeitsleistung in Form von Arbeitsgeschwindigkeit und Arbeitsqualität, sowie die Produktivität als Quotient aus geschaffenem Artefaktumfang und der dafür benötigten Zeit gemessen.

In jedem Simulationslauf wird die Motivation des beobachteten Mitarbeiters variiert. Da die Motivation mittels des Motivationsmodells aus unterschiedlichen Faktoren hervorgeht, lässt sie sich nicht direkt festlegen. Um die Motivation des Mitarbeiters zu beeinflussen, werden die einzelnen Einflussfaktoren auf die Motivation (vgl. Motivationsmodell in Abschnitt 2.4.5) gesetzt. Dazu wird für jeden Durchlauf ein zufälliger Wert aus dem Intervall $[0, 1]$ gewählt, auf den alle Einflussfaktoren gesetzt werden. Mit diesen Werten wird das Motivationsmodell aktualisiert und der daraus resultierende Motivationswert für den jeweiligen Durchlauf gemessen. Da sich das Motivationsmodell nicht linear verhält, und außerdem weitere Einflussfaktoren, wie der Zustand des Motivationsmodells und der Anspruch an die einzelnen Faktoren zufällig gewählt werden, ergibt sich hier für die Motivation keine Gleichverteilung. Da bei der Auswertung der Experimente nicht von einer bestimmten Verteilung ausgegangen wird, beeinflusst das die Qualität der Ergebnisse jedoch nicht.

Alle anderen Rahmenparameter werden zufällig gemäß der Normalverteilung um ihren Standardwert gewählt. Es wird ein Experiment mit 20 Sets à 100 Durchläufen, und ein weiteres Experiment mit 2.000 unabhängigen Durchläufen durchgeführt.

Die Hypothese ist zu akzeptieren, wenn für beide Experimente jeweils eine signifikante positive starke Korrelation ($r \geq +0.5$) zwischen Motivation und Arbeitsgeschwindigkeit, Arbeitsqualität und Produktivität beobachtet werden kann.



(a) Geschwindigkeit (Sets) (b) Qualität (Sets) (c) Produktivität (Sets)

(d) Geschwindigkeit (alle unabhängig) (e) Qualität (alle unabhängig) (f) Produktivität (alle unabhängig)

Abbildung 3.10.: Produktivität in Abhängigkeit der Motivation

Ergebnisse Abbildung 3.10 zeigt die Visualisierung der Messergebnisse. Hier ist zu beobachten, dass sich die Motivationswerte in manchen Bereichen häufen. Diese Häufungen entsprechen den unterschiedlichen Motivationszuständen, die das Modell einnehmen kann. Tabelle 3.4 enthält die dazugehörigen Rangkorrelationen. Alle Koeffizienten sind signifikant mit $p \ll 0.05$, und beschreiben in allen Fällen die geforderte Korrelation. Die Hypothese ist somit zu akzeptieren.

	Geschwindigkeit	Qualität	Produktivität
Sets	+0.99	+0.92	+0.99
alle unabhängig	+0.89	+0.71	+0.82

Tabelle 3.4.: Korrelationskoeffizienten der Messergebnisse zu Hypothese 8

Hypothese 9: Einfluss von Arbeitsbedingungen auf die Motivation

Es existiert ein Punkt, an dem eine Steigerung der Arbeitsbedingungen zu keinem Anstieg der Motivation mehr führt.

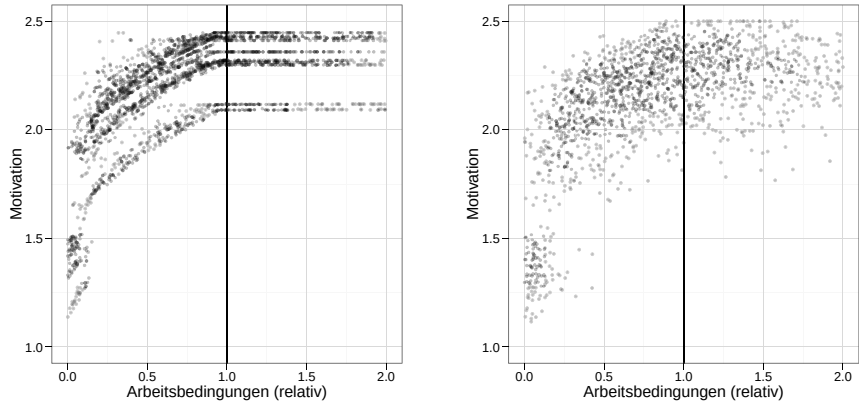
Arbeitsbedingungen sind ein wichtiger Hygienefaktor, der die Motivation, und damit die Arbeitsleistung beeinflusst [Herzberg, 1959]. Ab einem bestimmten Punkt führt eine weitere Verbesserung der Arbeitsbedingungen jedoch nicht mehr zu einer weiteren Steigerung der Motivation [Boehm, 1981]. Dieser Punkt wird im Simulationsmodell durch den Anspruch an die Arbeitsumgebung als Teil des Charakters jedes einzelnen Mitarbeiters festgelegt. Die Arbeitsbedingungen werden relativ zum Anspruch an sie betrachtet. Überschreiten die Arbeitsbedingungen den Anspruch, tritt keine weitere Verbesserung der Motivation mehr ein. Um diesen Effekt zu belegen, werden die folgenden Experimente durchgeführt.

Konzeption der Experimente In einem Szenario, dessen Rahmenparameter zufällig gemäß der Normalverteilung um ihren Standardwert gewählt werden, bearbeitet ein Mitarbeiter ein Artefakt. Die Arbeitsbedingungen werden dabei zufällig gemäß einer Gleichverteilung aus dem Intervall $[0, 1]$ gewählt. Ebenso zufällig und gleichverteilt wird der Anspruch des Mitarbeiters aus dem Intervall $[0, 1]$ gewählt, der angibt, welche Arbeitsbedingungen der Mitarbeiter erwartet. Aus den gewählten Arbeitsbedingungen und dem gewählten Anspruch werden die relativen Arbeitsbedingungen als Verhältnis zwischen Arbeitsbedingungen und Anspruch berechnet.

Ein Experiment wird in 20 Sets à 100 Simulationsläufe berechnet, wobei innerhalb eines Sets lediglich die Arbeitsbedingungen und der Anspruch an diese variiert werden, und alle anderen Rahmenparameter gleich bleiben. In einem weiteren Experiment werden in 2.000 unabhängigen Simulationsläufen jeweils alle Parameter variiert.

Gemessen wird die durchschnittliche Motivation während des gesamten Bearbeitungsvorgangs. Die Hypothese ist zu akzeptieren, wenn für Werte der relativen Arbeitsbedingung < 1 eine positive Korrelation zur Motivation gezeigt werden kann. Für Werte > 1 darf diese Korrelation nicht bestehen.

Ergebnisse Abbildung 3.11 zeigt die Ergebnisse der Experimente. Es ist zu beobachten, dass der geforderte Zusammenhang für Wert der relativen Arbeitsbedingung < 1 in beiden Experimenten vorhanden ist. Werden die Messungen in zwei entsprechende Datensätze aufgeteilt, so kann, ausgehende von der Hilfshypothese, dass eine Korrelation zwischen den relativen Arbeitsbedingungen und der Motivation besteht, für die Werte < 1 ein Rangkorrelationskoeffizient mit $r = +0.91$ bei $p \ll 0.05$ (Sets) bzw. $r = +0.64$ bei $p \ll 0.05$ (alle unabhängig) ermittelt werden. Für Werte > 1 gilt $r = 0.00$ bei $p = 0.96$ (Sets) bzw. $r = 0.01$ bei $p = 0.80$ (alle zufällig). Für Werte < 1 ist die Hilfshypothese zu akzeptieren, während für Werte > 1 von der Nullhypothese auszugehen ist. Damit ist die Gesamthypothese zu akzeptieren, da gezeigt werden konnte, dass mit dem Anspruch an die Arbeitsbedingungen der dort geforderte Punkt, an dem eine Steigerung der Bedingungen keine Wirkung mehr zeigt, existiert.



(a) Sets (b) alle unabhängig

Abbildung 3.11.: Motivation in Abhängigkeit der relativen Arbeitsbedingungen

Hypothese 10 und 11: Zusammenhang zwischen Teamgröße, Projektdauer und Effizienz

H10: *Je größer das Mitarbeiterteam ist, umso früher wird das Projekt fertiggestellt.*
H11: *Je größer das Mitarbeiterteam ist, umso niedriger ist die Produktivität pro Mitarbeiter.*

Die Arbeitskraft in einem Projekt wird unter anderem durch die Teamgröße beeinflusst. Jeder zusätzliche Mitarbeiter in einem Team erhöht direkt dessen Arbeitskraft. Da die Menge der Aufgaben, die in einem Projekt parallel bearbeitet werden können jedoch beschränkt ist, kann die zusätzliche Arbeitskraft ab einem gewissen Punkt nicht mehr produktiv eingesetzt werden. Zusätzlich kommt durch die Kommunikation, die zum Wissensaustausch der einzelnen Teammitglieder benötigt wird, ein zusätzlich überproportional wachsender Kostenfaktor hinzu, der die Arbeitskraft des Teams verringert.

Es ist daher zu erwarten, dass die Produktivität des Teams mit seiner Größe zunimmt, was gleichbedeutend damit ist, dass die benötigte Zeit bis zur Fertigstellung des Projekts abnimmt. Es ist außerdem zu erwarten, dass die Produktivität durch das Hinzufügen neuer Mitarbeiter nicht beliebig weit gesteigert werden kann. Außerdem ist, aufgrund des durch ein größeres Team deutlich erhöhten Kommunikationsbedarfs, eine niedrigere Effizienz des Teams zu erwarten.

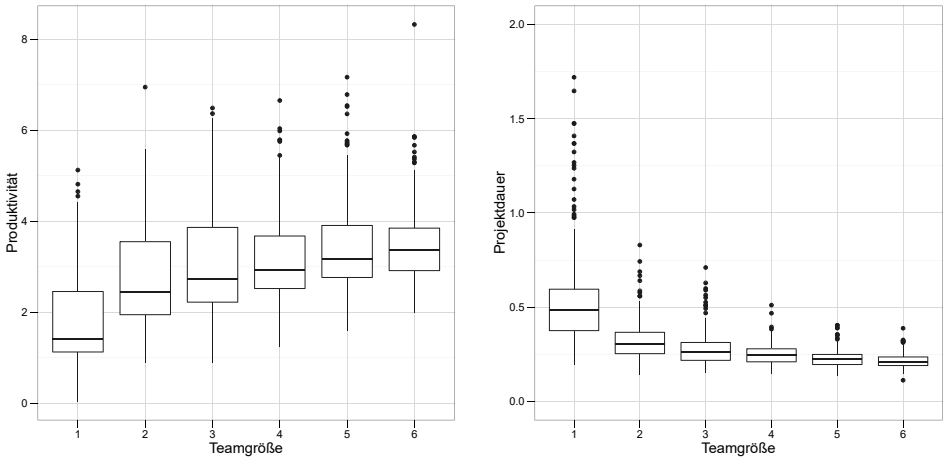
Konzeption der Experimente Um den Zusammenhang zwischen Teamgröße, Projektdauer, und Effizienz zu zeigen, wird ein Experiment mit 2.000 Durchläufen konzipiert. Ein Team mit einer Größe zwischen 1 und 6 Mitarbeitern bearbeitet gemeinsam das Produkt, das als einzelnes Artefakt konzipiert ist. Die Teamgröße wird für jeden Durchlauf zufällig gleichverteilt gewählt. Alle anderen Rahmenparameter werden für jeden Durchlauf zufällig normalverteilt um ihren Standardwert gewählt.

Gemessen wird die Produktivität als Quotient aus geschaffenem Produktumfang und der dafür benötigten Zeit, und die benötigte Zeit, um das Projekt fertigzustellen. Da der Projektumfang zwischen den einzelnen Durchläufen variiert, wird die benötigte Zeit durch den Projektumfang normiert, sodass die benötigte Dauer pro Arbeitseinheit gemessen wird. Außerdem wird die Effizienz in Form der durchschnittlichen Produktivität pro Mitarbeiter erfasst.

H10 muss akzeptiert werden, wenn eine positive Korrelation zwischen Teamgröße und Produktivität und eine negative Korrelation zwischen Teamgröße und Projektdauer gezeigt werden kann.

H11 muss akzeptiert werden, wenn eine negative Korrelation zwischen Teamgröße und Effizienz gezeigt werden kann.

Ergebnisse Abbildung 3.12 zeigt die Auswertung der Messergebnisse zur Produktivität und Projektdauer in Abhängigkeit der Teamgröße. Die Produktivität gibt die durchschnittlich geschaffenen Arbeitseinheiten pro Stunde an. Die Projektdauer beschreibt, wie viel Zeit für das Erstellen einer Arbeitseinheit benötigt wird, und ist damit das Inverse der Produktivität.



(a) Produktivität (b) Projektdauer

Abbildung 3.12.: Produktivität in Abhängigkeit der Teamgröße

Hier ist zu erkennen, dass die durchschnittliche Produktivität mit der Teamgröße ansteigt. Dies wird auch durch die positive Rangkorrelation von $r = +0.46$ belegt. Passend dazu ist zu erkennen, dass die benötigte Zeit mit der Teamgröße sinkt. Die dazugehörige Rangkorrelation beträgt $r = -0.66$.

Abbildung 3.13 zeigt die Effizienz in Abhängigkeit der Teamgröße. Die Effizienz gibt die pro Stunde erstellten Arbeitseinheiten pro Mitarbeiter an. Hier ist zu erkennen, das die Effizienz, im Gegensatz zur Produktivität des Teams, sinkt. Die dazugehörige Rangkorrelation beträgt $r = -0.73$.

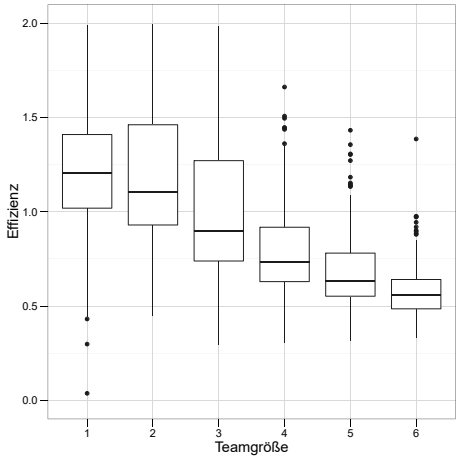


Abbildung 3.13.: Effizienz in Abhängigkeit der Teamgröße

Alle Messungen weisen die geforderten Korrelationen auf. Beide Hypothesen sind somit zu akzeptieren.

Auffällig bei allen Messungen ist die schnelle Sättigung, die schon bei einer relativ geringen Teamgröße auftritt. Ein Hinzufügen weiterer Teammitglieder hat schon ab einer Teamgröße von 3 keinen nennenswerten Vorteil mehr für die Produktivität des Teams. Grund hierfür ist die Gestaltung des Produkts in den hier beschriebenen Experimenten, das nur aus einem einzigen Artefakt besteht, und sich damit schlecht in mehrere unabhängige Aufgaben aufteilen lässt. In anderen Simulationsläufen hat sich gezeigt, dass bei mehr voneinander unabhängigen, oder nur schwach gekoppelten Artefakten deutlich größere Teams effizient arbeiten können.

Hypothese 12: Mehr oder bessere Mitarbeiter?

Es ist effektiver, wenige Mitarbeiter mit hohen Fähigkeiten zu nutzen, als viele Mitarbeiter mit niedrigen Fähigkeiten.

Diese Aussage lässt sich nicht direkt zeigen, da nicht klar ist, um welchen Grad sich die Fähigkeiten der wenigen Mitarbeiter im Vergleich zu den Fähigkeiten der vielen Mitarbeiter unterscheidet. Um zu zeigen, dass dieser Effekt im Simulationsmodell existiert, wird gezeigt, dass es einen Faktor gibt, um den die Fähigkeiten bei einer Vergrößerung des Teams verringert werden müssen, um eine damit einhergehende Verringerung der Produktivität beobachten zu können.

Konzeption der Experimente Die Abhängigkeit der Fähigkeit l der Teammitglieder wird in Abhängigkeit von der Teamgröße $|B|$ mit

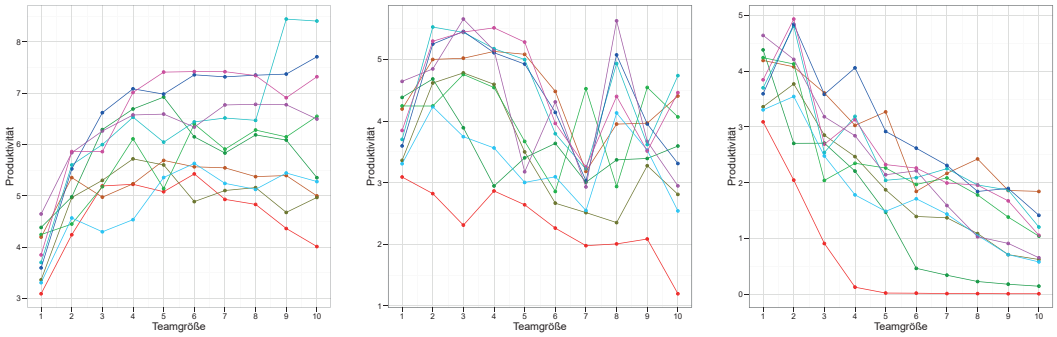
$$l = 1 - \log_2 |B| \cdot k \tag{3.1}$$

festgelegt. k steuert dabei die Stärke, um welche die Fähigkeiten bei einer Erhöhung der Teamgröße verringert werden. Eine Verdopplung der Teamgröße bewirkt eine Senkung der Fähigkeiten um k. Es werden unterschiedliche Werte für k geprüft, um herauszufinden, ab welchem Faktor eine Produktivitätssteigerung in einen Produktivitätsverlust übergeht.

Die Experimente dazu bestehen jeweils aus 100 Sets à 10 Durchläufen, wobei, begonnen mit einem Mitarbeiter, in jedem Durchlauf die Teamgröße um einen weiteren Mitarbeiter erhöht wird. Bearbeitet wird ein einzelnes Artefakt, dessen Umfang für jedes Set zufällig entsprechend der Normalverteilung mit einem Erwartungswert von 500U und einer Standardabweichung von 100U gewählt wird. Die Schwierigkeit des Artefakts wird ebenfalls für jedes Set zufällig gemäß der Normalverteilung mit einem Erwartungswert von 0.7 und einer Standardabweichung von 0.1 gewählt. Alle anderen Rahmenparameter werden für jedes Set zufällig gemäß der Normalverteilung um ihren jeweiligen Standardwert gewählt.

Gemessen wird pro Durchlauf die Produktivität des gesamten Teams, d. h der Quotient aus bewirktem Fortschritt und dafür benötigter Zeit bezogen auf das gesamte Projekt.

Ergebnisse Abbildung 3.14 zeigt die Ergebnisse der Experimente für $k = 0.0$, $k = 0.1$ und $k = 0.2$. Die Messpunkte eines Sets sind jeweils in einer gemeinsamen Farbe eingezeichnet. Der Übersichtlichkeit halber werden nur die Ergebnisse der ersten 10 Sets dargestellt.



(a) $k = 0.0$ (b) $k = 0.1$ (c) $k = 0.2$

Abbildung 3.14.: Produktivität in Abhängigkeit der Teamgröße (Ergebnisse der ersten 10 Sets)

Es lässt sich erkennen, dass für $k = 0.2$ ein eindeutiger Produktivitätsverlust stattfindet, wenn das Team auf Kosten der Fähigkeiten seiner Mitarbeiter vergrößert wird. Dieser Eindruck wird durch eine signifikante Rangkorrelation von $r = -0.96$ gestützt.

Als Kontrollgruppe dient das Experiment mit $k = 0.0$ bei dem die Fähigkeiten nicht verändert werden. Hier lässt sich erkennen, dass mit der Teamgröße auch die Produktivität zunimmt, was sich in einer signifikanten Rangkorrelation von $r = +0.82$ widerspiegelt.

Für $k = 0.1$ lässt sich keine einheitliche Zunahme oder Abnahme der Produktivität erkennen. Der Effekt hängt hier hauptsächlich von den Rahmenparametern des Sets ab. Das lässt darauf schließen, dass die negativen Effekte durch die niedrigeren Fähigkeiten, den positiven durch das größere Team überwiegen, wenn bei einer Verdopplung des Teams die Fähigkeiten um mehr als 0.1 verringert werden.

Der nicht monotone Verlauf am einigen Stellen innerhalb der Sets wird durch die Unterschiede der Mitarbeiter innerhalb des Teams verursacht. Diese sind bis auf deren Fähigkeiten zufällig gewählt. Dadurch kann es passieren, dass ein weiterer Mitarbeiter die Produktivität entgegen des Trends verändert, da er in seinen übrigen Parametern deutliche Vor- oder Nachteile gegenüber dem bisherigen Team hat.

Die Auswertung der Experimente zeigt, dass die Hypothese in der oben besprochenen Abwandlung akzeptiert werden kann. Es existiert im Simulationsmodell ein Wert für die Verringerung der Fähigkeiten in Abhängigkeit der Teamgröße, ab dem der beschriebene Effekt nachweisbar ist.

Hypothese 13: Brooks' Law

Werden zu einem späten Zeitpunkt im Projekt neue Mitarbeiter hinzugefügt, verzögert sich das Projektende.

Wenn zu einem späten Zeitpunkt im Projekt neue Projektmitarbeiter zu einem Projektteam hinzugefügt werden, so kommt es dadurch zu einer Mehrbelastung der bestehenden Belegschaft aufgrund der Einarbeitung der neuen Mitarbeiter. Gleichzeitig sind die neuen Mitarbeiter in der ersten Zeit wenig produktiv und können so den Mehraufwand, der durch ihre Einarbeitung entsteht, nicht kompensieren. Dadurch sinkt die Gesamtproduktivität des Teams in der ersten Zeit. [Brooks, 1995]

Konzeption der Experimente Betrachtet werden Projekte, bei denen ein Team gemeinsam ein Produkt entwickelt, dass nicht näher spezifiziert ist, und deshalb als einzelnes Artefakt modelliert wird. Das Projekt wird durchgeführt, bis das zu erstellende Artefakt vollständig ist. Das Team besteht zu Anfang aus zwei Entwicklern. Sobald das Artefakt eine relative Vollständigkeit von c erreicht hat, werden dem Team zwei weitere Mitarbeiter hinzugefügt. Alle anderen Parameter des Projekts, der Mitarbeiter, des Artefakts und der zu verwendenden Fortschrittsaktivität werden zufällig entsprechend der Normalverteilung bestimmt. Der Erwartungswert entspricht dabei dem jeweiligen Standardwert.

Aus der zu prüfenden Hypothese lässt sich ableiten, dass es im Projektverlauf einen zeitlichen Punkt geben muss, vor dem das Hinzufügen neuer Mitarbeiter die Produktivität erhöht. Werden Mitarbeiter hingegen nach diesem Zeitpunkt hinzugefügt, sinkt die Produktivität.

Um das zu prüfen, wird das Verhältnis zwischen der Produktivität vor und nach dem Hinzufügen neuer Mitarbeiter gemessen und in Abhängigkeit des Zeitpunkts an dem das Hinzufügen stattgefunden hat dargestellt. Dazu werden 2.000 Simulationsläufe durchgeführt, bei denen zu einem zufällig gewählten Zeitpunkt zwischen $c = 0.1$ und $c = 0.9$ eine Verdopplung des Teams stattfindet. Die anderen Parameter des Szenarios werden für jeden Simulationslauf zufällig neu gewählt.

Ergebnisse Abbildung 3.15 zeigt das so erhaltene Ergebnis der Messung. Hier ist zu erkennen, dass eine Verdopplung der Mitarbeiterzahl zu Beginn des Projekts die Produktivität um ca.80% steigert. Die Differenz zu einer Leistungszunahme von 100% ist durch den zusätzlich entstehenden Kommunikationsaufwand zu erklären. Je später die Verdopplung des Teams stattfindet, umso geringer fällt die Produktivitätssteigerung aus. Bei $c \approx 0.75$ findet keine Steigerung mehr statt. Werden die Mitarbeiter im letzten Viertel des Projekts hinzugefügt, sinkt die Produktivität des Projektteams, was wiederum zu der von Brooks vorhergesagten Verzögerung des Projektendes führt.

Das Diagramm zeigt auch, dass der Effekt nicht in allen Fällen auftritt. Es gibt eine Reihe von Situationen, in denen das Hinzufügen neuer Mitarbeiter auch zu einem späten Zeitpunkt noch eine entsprechende Steigerung der Produktivität bewirkt. Das ist beispielsweise der Fall, wenn die neu hinzugekommenen Mitarbeiter deutlich bessere Fähigkeiten mitbringen, als die vorhandene Belegschaft besitzt, oder sie über eine hohe Auffassungsgabe verfügen und daher schnell lernen.

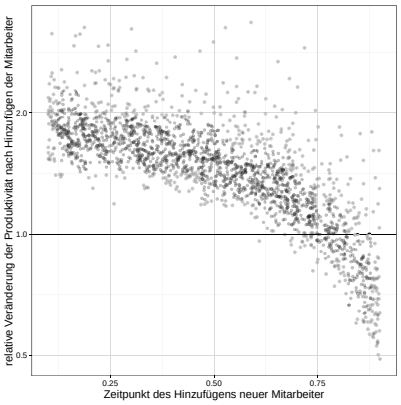
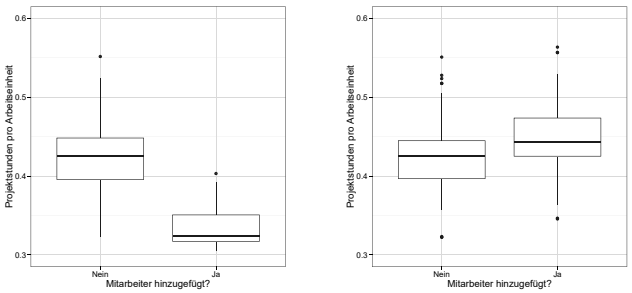


Abbildung 3.15.: Änderung der durchschnittlichen Produktivität durch das Hinzufügen neuer Mitarbeiter (Erstveröffentlichung in [Nassal, 2017])

Konzeption weiterer Experimente Der von Brooks beschriebene Effekt lässt sich auch direkt anhand der Simulationsergebnisse zeigen. Dazu werden in zwei weiteren unabhängigen Experimenten jeweils die Projektlaufzeiten zweier Projekte verglichen, die sich lediglich darin unterscheiden, dass im ersten Projekt die Mitarbeiterzahl erhöht wird, im zweiten jedoch nicht. Die Differenz zwischen den Projektlaufzeiten ergibt die Zeit, die durch das Hinzufügen der neuen Mitarbeiter gewonnen bzw. verloren wird. Es werden zwei Gruppen mit jeweils 300 Projekten simuliert. In der ersten Gruppe findet das Hinzufügen der Mitarbeiter im Zeitraum $c \in [0.0, 0.75)$, und in der zweiten Gruppe im Zeitraum $c \in [0.75, 1.0]$ statt. Die übrigen Rahmenbedingungen entsprechen denen der vorherigen Experimente. Die Intervallgrenze bei 0.75 ergibt sich aus den Ergebnissen des ersten Experiments.

Ergebnisse Abbildung 3.16 zeigt die Ergebnisse in Form einer Kastengrafik. Da auch der Umfang der Projekte zufällig gewählt wird, ist die auf der Y-Achse aufgetragene Projektdauer auf eine einzelne Arbeitseinheit bezogen, um die Messungen vergleichbar zu machen. Aus der linken Abbildung ist so zu entnehmen, dass die durchschnittlich benötigte Erstellungszeit pro Arbeitseinheit des Produkts, durch das frühe Hinzufügen von Mitarbeitern zum Projekt im Mittel von 0.42h auf 0.32h fällt, wohingegen sie beim späten Hinzufügen auf 0.45h ansteigt. Da hier die gesamte Projektzeit betrachtete wird, fällt die Änderung beim späten Hinzufügen geringer aus, da die neue Situation nur über einen vergleichsweise kurzen Zeitraum am Ende des Projekts besteht, und damit nur einen geringen Anteil am Gesamtergebnis hat.



(a) frühes Hinzufügen von Mitarbeitern

(b) spätes Hinzufügen von Mitarbeitern

Abbildung 3.16.: Veränderung der Projektdauer durch das Hinzufügen von neuen Mitarbeitern

Die Signifikanz der Ergebnisse kann mittels eines gepaarten Wilcoxon-Tests mit $p \ll 0.05$ gezeigt werden. Ein Paar besteht dabei jeweils aus dem Messwert eines Projekts, mit und ohne Hinzufügen neuer Mitarbeiter. Die Effektstärke entspricht den Differenzen der oben genannten Mittelwerte.

Hypothese 14: Zusammenhang zwischen Defektzahl und Aufdeckungsrate

Je mehr Defekte in einem Dokument sind, umso höher ist die Rate mit der Defekte darin aufgedeckt werden.

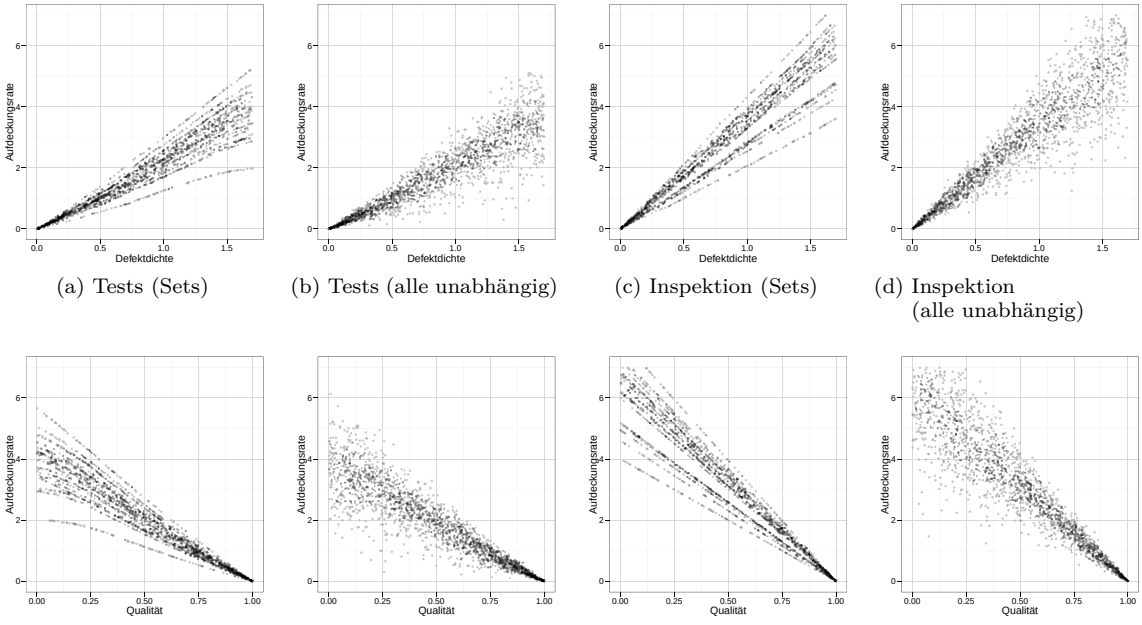
Zwischen der Defektdichte in einem Stück Software, und der Rate mit der die Defekte gefunden werden, besteht ein direkter Zusammenhang [McGraw, 2006]. Diese Tatsache wird in vielen Verfahren zur Restfehlerschätzung ausgenutzt, um abzuschätzen, wie viele Defekte noch nicht gefunden wurden. Je mehr Defekte in einem Dokument sind, umso größer ist die Wahrscheinlichkeit, dass einer davon zufällig oder durch strukturierte Qualitätssicherungsmaßnahmen aufgedeckt wird.

Konzeption der Experimente Um zu untersuchen, ob im Simulationsmodell ein Zusammenhang zwischen der Defektdichte und der Geschwindigkeit, mit der die Defekte im Zuge der Qualitätssicherungsaktivitäten aufgedeckt werden besteht, werden vier Experimente durchgeführt.

In allen Experimenten wird ein fertiggestelltes Artefakt vorgegeben, das jedoch Defekte enthält. Die Defektdichte des Artefakts wird in allen Experimenten für jeden Simulationslauf neu gewählt. Dazu wird die Qualität des Artefakts zufällig gemäß einer Gleichverteilung gewählt, und solange Defekte in das Artefakt eingebracht, bis diese Qualität erreicht ist. Dazu wird das entsprechende Elementarmodell (vgl. Abschnitt 2.4.4) verwendet, dass anhand eines Umfangs und eines Qualitätsniveaus entsprechende Defekte erzeugt.

Ein Mitarbeiter bearbeitet das Artefakt über einen Zeitraum von 1 Woche mit einer Qualitätssicherungsaktivität. Gemessen wird der Umfang der aufgedeckten Defekte pro Stunde. Zwei Experimente werden mit einer Testaktivität, die anderen beiden mit einer Inspektionsaktivität durchgeführt. Je ein Experiment jeder Gruppe wird in 20 Sets à 100 Simulationsläufen, das jeweils andere Experiment in 2.000 unabhängigen Simulationsläufen durchgeführt. Alle Rahmenparameter werden zufällig gemäß der Normalverteilung um ihren jeweiligen Standardwert, entweder pro Set oder pro Durchlauf, gewählt.

Die Hypothese ist zu akzeptieren, wenn für alle Experimente positive Korrelationen zwischen Defektdichte und Aufdeckungsrate, beziehungsweise negative Korrelationen zwischen Artefaktqualität und Aufdeckungsrate nachgewiesen werden können.



(a) Tests (Sets) (b) Tests (alle unabhängig) (c) Inspektion (Sets) (d) Inspektion (alle unabhängig)

(e) Tests (Sets) (f) Tests (alle unabhängig) (g) Inspektion (Sets) (h) Inspektion (alle unabhängig)

Abbildung 3.17.: Zusammenhang zwischen Defektanzahl und Aufdeckungsgeschwindigkeit

Ergebnisse Tabelle 3.5 zeigt die, auf Basis der durch die Experimente erzeugten Daten, berechneten Rangkorrelationskoeffizienten. Alle Korrelationen weisen eine Signifikanz von $p \ll 0.5$ auf. Abbildung 3.17 enthält die dazugehörigen Streudiagramme. Da die geforderten Korrelationen gezeigt werden können, ist die Hypothese zu akzeptieren.

	Testen		Inspektion	
	Sets	alle unabhängig	Sets	alle unabhängig
Defektdichte	$+0.99$	$+0.93$	$+0.94$	$+0.99$
Qualität	-0.99	-0.99	-0.94	-0.99

Tabelle 3.5.: Korrelationskoeffizienten der Messergebnisse zu Hypothese 14

Hypothese 15: Lernen durch Qualitätssicherungsaktivitäten

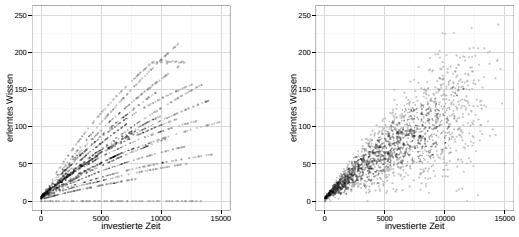
Mitarbeiter, die an Qualitätssicherungsaktivitäten teilnehmen, erweitern dadurch ihr Artefaktwissen.

Auch für Qualitätssicherungsaktivitäten wird Wissen über den Prüfgegenstand und seinen Kontext benötigt. Dieses Wissen muss, falls nicht schon vorhanden, erworben werden, um die Qualitätssicherungsaktivitäten effektiv durchführen zu können. Eine Qualitätssicherungsaktivität geht daher auch häufig mit dem Erwerb von Produktwissen einher. Insbesondere Inspektionsaktivitäten, bei denen meistens nicht nur die Schnittstellen des Prüflings, sondern sein gesamter Inhalt begutachtet wird, fördert das Produktwissen [Tvedt, 1996].

Konzeption der Experimente Um zu zeigen, dass die Mitarbeiter im Simulationsmodell während der Durchführung von Qualitätssicherungsaktivitäten Wissen über die betroffenen Artefakte erwerben, werden zwei Experimente durchgeführt. Dabei wird ein Artefakt mittels einer Qualitätssicherungsaktivität untersucht, das vollständig erstellt, jedoch defektbehaftet ist. Dazu wird die Qualität des Artefakts zufällig aus dem Intervall $[0, 1]$ gemäß einer Normalverteilung um einen Erwartungswert von 0.4 und einer Standardabweichung von 0.1 gewählt. Alle weiteren Rahmenparameter werden zufällig gemäß der Normalverteilung im ihren jeweiligen Standardwert gewählt. Variiert wird die Dauer, die in die Qualitätssicherungsaktivität investiert wird. Gemessen wird die Zunahme des Artefaktwissens des bearbeitenden Mitarbeiters in Abhängigkeit der in die Qualitätssicherung investierten Zeit.

Ein Experiment wird in 20 Sets à 100 Simulationsläufen durchgeführt, wobei die Rahmenparameter innerhalb eines Sets konstant bleiben; variiert wird nur die investierte Zeit. Im zweiten Experiment werden die Rahmenparameter in jedem der 2.000 Simulationsläufe neu gewählt. Die Hypothese ist zu akzeptieren, wenn das erworbene Produktwissen positiv mit der in die Qualitätssicherung investierten Zeit korreliert.

Ergebnisse Die geforderte Korrelation lässt sich mit einer hohen Signifikanz von $p \ll 0.05$ in beiden Experimenten nachweisen. Der Korrelationskoeffizient beträgt für das in Sets durchgeführte Experiment $r = +0.99$ und für die unabhängigen Simulationsläufe $r = 0.83$. Damit ist die Hypothese zu akzeptieren. Abbildung 3.18 zeigt die dazugehörigen Diagramme.



(a) Sets (b) alle unabhängig

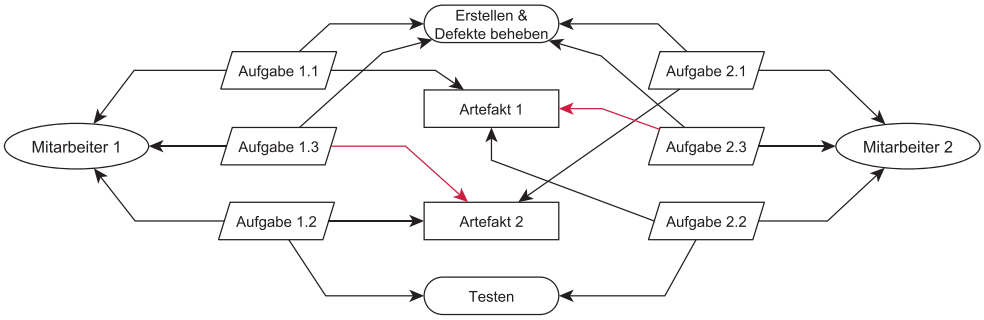
Abbildung 3.18.: Zusammenhang zwischen erlerntem Produktwissen und investierter Zeit

Hypothese 16: Autorenvorteil beim Beheben von Defekten

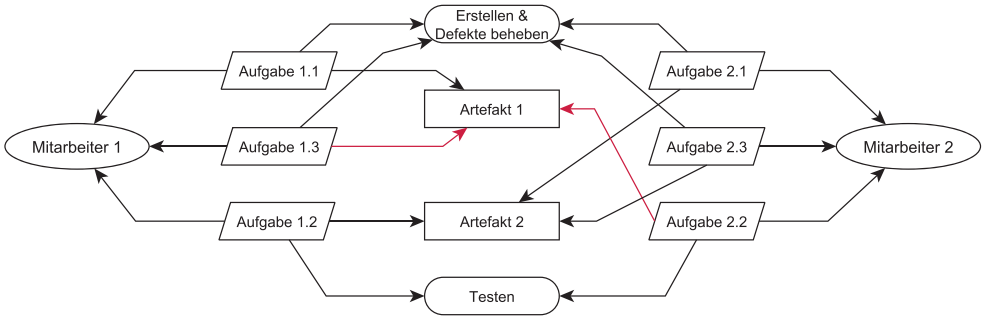
Der Autor eines Dokuments kann die darin gefundene Defekte schneller beheben als der Mitarbeiter, der sie gefunden hat.

Defekte werden am effizientesten vom Autor des Dokuments behoben [Drappa, 2000b]. Dieser kennt sich am besten damit aus und kann daher die Defekte, die beim Testen aufgedeckt wurden, schnell und zuverlässig beheben.

Konzeption der Experimente Um zu zeigen, dass dieser Effekt im Simulationsmodell erzeugt wird, werden zwei Szenarien miteinander verglichen, die in Abbildung 3.19 beschrieben sind. Die beiden Szenarien unterscheiden sich lediglich durch die rot markierten Beziehungen.



(a) Defektbehebung durch den Tester



(b) Defektbehebung durch den Autor

Abbildung 3.19.: Rahmenszenarien zur Überprüfung von Hypothese 16

Die Szenarien enthalten jeweils drei Aufgaben, die im Projektplan entsprechend definiert sind. Im Rahmen der ersten Aufgabe erstellt ein Mitarbeiter ein Artefakt. Mit der zweiten Aufgabe wird dieses Artefakt durch einen weiteren Mitarbeiter getestete, um die darin enthaltenen Defekte aufzudecken. Durch die dritte Aufgabe werden die aufgedeckten Defekte behoben. Im ersten Szenario übernimmt das der Mitarbeiter, der sie im Rahmen des Testens aufgedeckt hat. Im zweiten Szenario übernimmt der Autor die Behebung der Defekte. Um Effekte zu unterbinden, die beispielsweise dadurch entstehen können, dass der testende Mitarbeiter in der ersten Projekthälfte keine Aufgabe hat, wird der Aufbau jeweils gespiegelt. Die beiden Mitarbeiter bearbeiten so jeweils ein eigenes Artefakt, testen anschließend das ihres Kollegen, und beheben dann, je nach Szenario, entweder die Defekte im eigenen Artefakt oder die im Artefakt des Kollegen.

Es werden 500 Szenariopaare simuliert, die sich lediglich darin unterscheiden, ob die Defekte im eigenen Artefakt oder dem des Kollegen behoben werden. Alle Rahmenparameter werden für jedes Szenariopaar zufällig, entsprechend der Normalverteilung um ihren Standardwert, gewählt.

Gemessen wird die durchschnittliche Produktivität beider Mitarbeiter während der letzten Aufgabe, die der Behebung der Defekte dient. Dazu wird der Umfang der behobenen Defekte in ein Verhältnis zur dafür benötigten Zeit gesetzt.

Ergebnisse Abbildung 3.20 zeigt die aus der Simulation beider Szenarien hervorgehenden Messergebnisse. Werden die Defekte durch den Tester behoben, so liegt die Produktivität im Durchschnitt bei 1.8 Arbeitseinheiten pro Stunde. Im Vergleich dazu liegt die Produktivität bei einer Behebung der Defekte durch den Autor im Durchschnitt bei 3.1 Arbeitseinheiten pro Stunde.

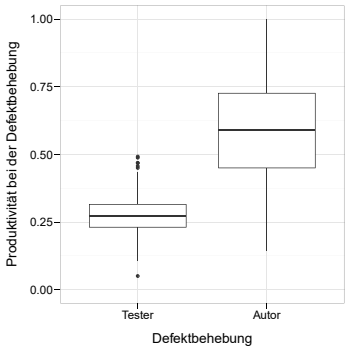


Abbildung 3.20.: Vergleich Defektbehebung

Die Signifikanz dieses Unterschieds lässt sich mittels des gepaarten Wilcoxon-Test nachweisen. Der Test ergibt eine Signifikanz von $p \ll 0.05$ und eine Effektstärke von -1.2, die den durchschnittlichen Unterschied der Produktivität in den Szenariopaaren angibt.

Hypothese 17: Produktivitätsvorteil kontinuierlicher Qualitätssicherung

Begleitende Qualitätssicherung führt zu einer höheren Produktivität als Qualitätssicherung ausschließlich am Ende eines Projekts.

Mit dieser Hypothese wird untersucht, wie sich unterschiedliche Strategien bezüglich der Qualitätssicherung auf die Produktivität auswirken. Betrachtet wird ein wasserfallartiges Vorgehen, bei dem die Qualitätssicherungsaktivitäten am Ende des Entwicklungsprozesses stehen, und ein Vorgehen, bei dem begleitend zu Entwicklungsprozess kontinuierlich Qualitätssicherungsaktivitäten vorgenommen werden, wie sie beispielsweise in [Balzert, 2008] gefordert wird.

Konzeption der Experimente Im ersten Experiment wird von einem Mitarbeiter ein einzelnes Artefakt bearbeitet. Alle Rahmenparameter werden zufällig entsprechend der Normalverteilung um ihren Standardwert gewählt. Es werden 1.000 Paare simuliert, bei denen die Rahmenparameter jeweils identisch gewählt werden. Im ersten Durchlauf wird das wasserfallartige Vorgehen umgesetzt, im zweiten Durchlauf die kontinuierliche Qualitätssicherungsstrategie.

Um das entsprechende Vorgehen zu simulieren, wird ein zusätzlicher Agent eingefügt, der den Projektplan während der Simulation verwaltet. Für das wasserfallartige Vorgehen wird zuerst die Fortschrittsaktivität eingefügt, die ausgeführt wird, bis das Artefakt vollständig erstellt wurde. Anschließend wird die Qualitätssicherungsaktivität eingesetzt um Defekte aufzudecken, bis das Qualitätsniveau über 0.8 liegt. Um die gefundenen Defekte zu beheben, wird anschließend wieder die Fortschrittsaktivität eingesetzt, bis alle Defekte behoben sind. Da während der Behebung neue Defekte in das Artefakt eingebracht worden sein können, wird gegebenenfalls erneut die Qualitätssicherungsaktivität ausgeführt. Der Wechsel zwischen Fortschrittsaktivität und Qualitätssicherungsaktivität wird solange fortgeführt, bis das Artefakt vollständig ist, und gleichzeitig eine Qualität von mindesten 0.7 aufweist.

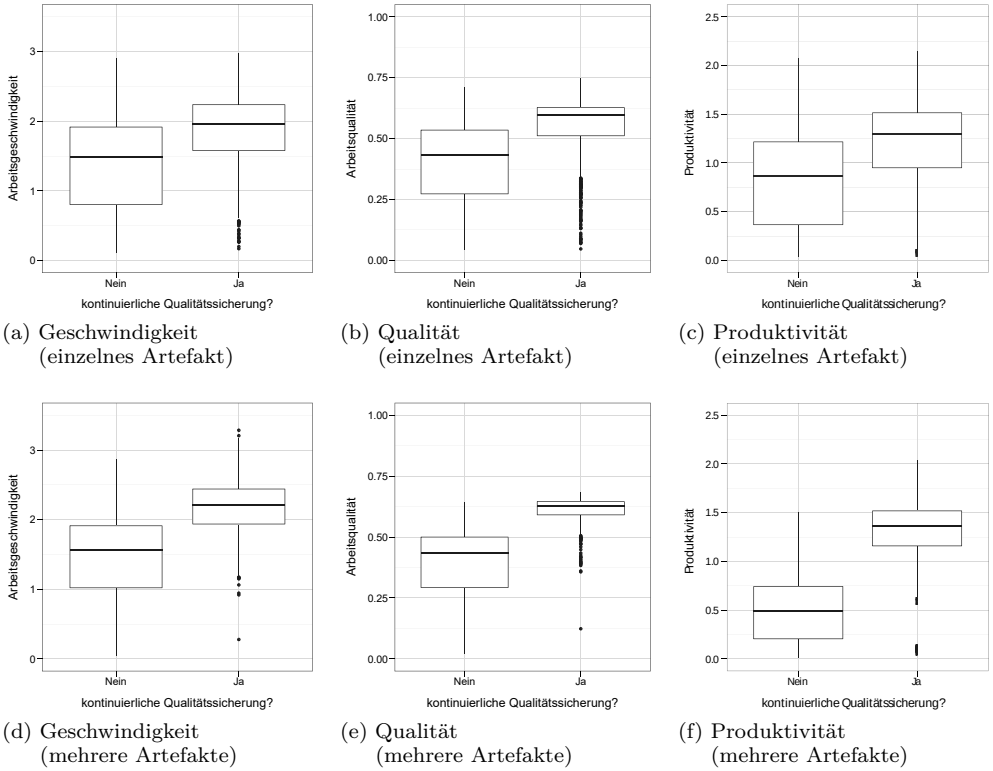
Bei der kontinuierlichen Qualitätssicherung wird die Qualität des Artefakts überwacht. Sinkt diese unter 0.6, wird die Fortschrittsaktivität unabhängig vom aktuellen Fortschritt des Artefakts unterbrochen, und von der Qualitätssicherungsaktivität solange ersetzt, bis das Artefakt eine Qualität von über 0.8 erreicht. Das Szenario endet, sobald das Artefakt vollständig ist, und eine Qualität von mindesten 0.7 aufweist.

Laut [Balzert, 2008] ist eine entwicklungsbegleitende Qualitätssicherung vor allem notwendig, da die verschiedenen Entwicklungsschritte aufeinander aufbauen, und Defekte eines vorangegangenen Schritts die darauffolgenden Schritte negativ beeinflussen. Mittels eines einzelnen Artefakts lässt sich ein solches Szenario nicht untersuchen. Deshalb wird in einem zweiten Experiment das Produkt um weitere Artefakte erweitert. Anstelle eines einzelnen Artefakts werden nun mehrere durch Vorbedingungen und Kontextbeziehungen abhängige Artefakte entwickelt. Ein solches Produkt umfasst 1 bis 15 Artefakte. Die Anzahl der Artefakte, sowie die Abhängigkeiten werden für jedes Paar zufällig gewählt. Das Szenario endet, wenn alle Artefakte vollständig mit einer Qualität von mindesten 0.7 entwickelt wurden.

Gemessen wird die Arbeitsleistung in Form von Geschwindigkeit und Qualität, sowie die Produktivität als Quotient aus geschaffenem Produktumfang und der dafür benötigten Zeit.

Die Hypothese ist zu akzeptieren, wenn gezeigt werden kann, dass bei einer kontinuierlichen Qualitätssicherung eine signifikant höhere Geschwindigkeit, Qualität und Produktivität erzielt werden wird.

Ergebnisse Abbildung 3.21 zeigt die Ergebnisse beider Experimente in Form einer Kastengrafik.



(a) Geschwindigkeit (einzelnes Artefakt)
(b) Qualität (einzelnes Artefakt)
(c) Produktivität (einzelnes Artefakt)
(d) Geschwindigkeit (mehrere Artefakte)
(e) Qualität (mehrere Artefakte)
(f) Produktivität (mehrere Artefakte)

Abbildung 3.21.: Produktivität mit und ohne kontinuierliche Qualitätssicherung

Mittels des paarweisen Wilcoxon-Tests lässt sich zeigen, dass die beiden Strategien einen signifikanten Unterschied aufweisen. Tabelle 3.6 zeigt die Effektstärke in Form des durchschnittlichen Unterschieds zwischen den beiden Strategien. Für alle Werte gilt eine Signifikanz von $p \ll 0.05$.

	Geschwindigkeit	Qualität	Produktivität
einzelnes Artefakt	0.45	0.14	0.37
mehrere Artefakte	0.70	0.21	0.84

Tabelle 3.6.: Effektstärken zu Hypothese 17

Die Auswertung zeigt, dass eine kontinuierliche Qualitätssicherung für alle Messkategorien eine höheres, und damit besseres Ergebnis liefert. Prozentual bezogen auf das wasserfallartige Vorgehen, lässt sich bei der kontinuierlichen Qualitätssicherung eine Steigerung der Produktivität um 42% für das einzelne Artefakt, und eine Steigerung um 166% für das Produkt bestehend aus mehreren Artefakten feststellen. Der Unterschied zwischen dem einzelnen Artefakt und mehrere zusammenhängender Artefakte lässt sich zuverlässig über die Gewichte der Kontexteinflüsse steuern, wie weitere Experimente mit unterschiedlichen Gewichten gezeigt haben. Je stärker die einzelnen Entwicklungsschritte aufeinander aufbauen, umso stärker ist auch der Vorteil, den eine entwicklungsbegleitende Qualitätssicherung mit sich bringt.

Hypothese 18: Zusammenhang zwischen Teamgröße und der Effizienz bei Inspektionen

Inspektionen sind mit einer Gruppe bestehend aus 4 Mitarbeitern am effizientesten.

Ein wesentlicher Bestandteil von Inspektionen ist die Diskussion über die gefundenen Auffälligkeiten. Der Aufwand für die Kommunikation steigt überproportional mit der Anzahl der Kommunikationspartner. Untersuchungen, wie in [Weller, 1993] beschrieben, zeigen, dass eine gute Teamgröße für Inspektionen 4 Mitarbeitern umfasst, da hier ein guter Kompromiss zwischen dem Zusammenwirken der unterschiedlichen Fähigkeiten und Blickwinkeln und dem Kommunikationsaufwand besteht.

Konzeption der Experimente Um zu zeigen, dass dies auch für die Inspektionsaktivität des Simulationsmodells gilt, wird ein entsprechendes Experiment konzipiert. Das Szenario dazu enthält ein vollständiges Artefakt im zufällig gewählten Umfang zwischen 100 und 1.000 Arbeitseinheiten, das eine zufällig gewählte Qualität zwischen 0 und 1 mit den entsprechenden Defekten aufweist. Dieses Artefakt wird mittels einer Inspektionsaktivität von einem Inspektionsteam untersucht. Alle weiteren Rahmenparameter werden zufällig gemäß der Normalverteilung um ihren jeweiligen Standardwert gewählt. Für jede Rahmenparameterbelegung werden 10 Durchläufe simuliert, wobei, begonnen mit einem Mitarbeiter, das Inspektionsteam in jedem weiteren Durchlauf um einen weiteren Mitarbeiter vergrößert wird. Es werden 200 solcher Simulationsreihen durchgeführt.

Gemessen werden die Summe der aufgedeckten Defekte pro Stunde und die Effizienz des Teams als Quotient aus dieser Summe und der Teamgröße.

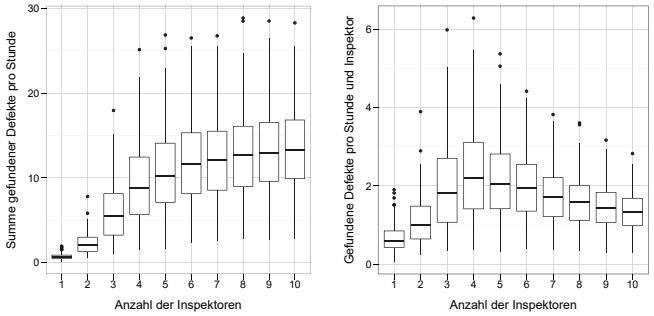
Die Hypothese ist zu akzeptieren, wenn für die Effizienz ein Maximum bei einer Teamgröße von 4 Mitarbeitern gezeigt werden kann.

Ergebnisse Abbildung 3.22 zeigt die Auswertung des Experiments. Hier ist zu erkennen, dass die Summe der aufgedeckten Defekte monoton mit der Teamgröße steigt, jedoch gegen einen Grenzwert zu streben scheint. Grund dafür ist der mit der Teamgröße überproportional zunehmende Kommunikationsaufwand. Die rechte Abbildung zeigt, dass das Effizienzmaximum bei 4 Mitarbeitern liegt. Mittels des gepaarten Wilcoxon-Tests lässt sich zeigen, dass dieses Maximum signifikant ist. Dazu werden jeweils aus einer Simulationsreihe das Szenario mit 3 und 4 bzw. 4 und 5 Mitarbeitern als Paar betrachtet. Der Test zeigt, dass die Szenarien mit 3 bzw. 5 Mitarbeitern eine signifikant niedrigere Effizienz ($p \ll 0.05$) aufweisen als die Szenarien mit 4 Mitarbeitern.

	durchschnittlich gefundene Defekte pro Stunde									
	1	2	3	4	5	6	7	8	9	10
insgesamt	0.67	2.26	5.87	9.08	10.75	11.81	12.19	12.73	13.12	13.40
pro Inspektor	0.67	1.13	1.95	2.27	2.15	1.97	1.74	1.59	1.46	1.34

Tabelle 3.7.: Arbeitsleistung bei Inspektionen

Tabelle 3.7 enthält die zu Abbildung 3.22 gehörenden Mittelwerte. In [Weller, 1993] findet sich die Aussage, dass ein Inspektionsteam bestehend aus vier Mitarbeitern doppelt so effektiv ist, wie ein Team mit 3 Mitarbeitern. Betrachtete man die am Simulationsmodell gemessenen Werte, so lässt sich feststellen, dass ein Team mit 4 Mitarbeitern im Mittel 55% mehr Defekte aufdeckt als ein Team mit 3 Mitarbeitern. Die Hypothese ist somit zu akzeptieren, das Simulationsmodell kann, vorausgesetzt



(a) Arbeitsleistung (b) Effizienz

Abbildung 3.22.: Effizienz bei Inspektionen in Abhängigkeit der Größe des Inspektionsteams

man hält an den Zahlen nach Weller fest, an dieser Stelle jedoch gegebenenfalls noch verbessert werden, um den quantitativen Aspekt besser abzubilden.

Hypothese 19 und 20: Einfluss von Zeitdruck auf die Produktivität

H19: *Je höher der Zeitdruck auf einen Mitarbeiter ist, umso niedriger ist seine Produktivität, wenn der Zeitdruck über einen langen Zeitraum aufrechterhalten bleibt.*
H20: *Je höher der Zeitdruck auf einen Mitarbeiter ist, umso höher ist seine Produktivität, wenn der Zeitdruck nur über einen kurzen Zeitraum aufrechterhalten bleibt.*

Zeitdruck beeinflusst die Arbeitsleistung der Mitarbeiter. Während eine Erhöhung des Zeitdrucks für kurze Zeitintervalle die Produktivität der Mitarbeiter erhöht, wirkt sich ein andauernder Zeitdruck langfristig negativ auf die Produktivität aus [DeMarco, 2002]. Grund dafür ist, dass die Mitarbeiter eine kurzfristige Mehrbelastung durch Energiereserven gut kompensieren können, eine dauerhafte Überbelastung sich jedoch typischerweise negativ auf Gesundheit und Motivation auswirkt.

Konzeption der Experimente Um diesen Effekt am Simulationsmodell zu zeigen, werden vier Experimente durchgeführt. In allen Experimenten wird ein einzelner Mitarbeiter beobachtet, der ein Artefakt bearbeitet. Alle Rahmenparameter werden entsprechend der Normalverteilung um ihren Standardwert zufällig gewählt.

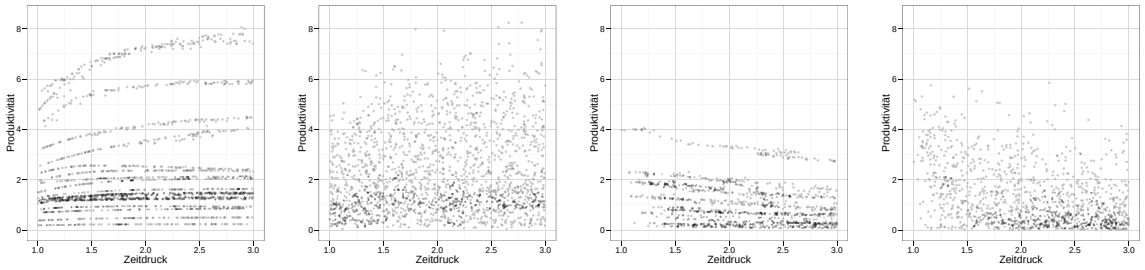
In jedem Durchlauf wird der Zeitdruck auf den Mitarbeiter zufällig gemäß einer Gleichverteilung aus dem Intervall $[1, 3]$ gewählt. 1 bedeutet, der Mitarbeiter hat genau so viel Zeit wie er benötigt, 3 bedeutet, er benötigt dreimal so viel Zeit wie ihm zur Verfügung steht. Der Zeitdruck entsteht durch die Gestaltung des Projektplans. Mittels eines zusätzlichen Agenten, der den Projektplan in jedem Berechnungsschritt prüft und gegebenenfalls anpasst, wird sichergestellt, dass unabhängig von der Arbeitsleistung des Mitarbeiters ein konstanter Zeitdruck herrscht, der dem zufällig gewählten Wert entspricht.

Die Experimente werden in zwei Gruppen aufgeteilt. Jede Gruppe enthält ein Experiment mit 20 Sets à 100 Durchläufen, wobei die Rahmenparameter innerhalb eines Sets gleich gewählt werden, und ein Experiment mit 2.000 Durchläufen, wobei in jedem Durchlauf alle Rahmenparameter zufällig neu gewählt werden. In der ersten Gruppe wird die Produktivität über einen Zeitraum von 10 Tagen betrachtet. In der zweiten Gruppe wird der Zeitdruck über 365 Tage hinweg betrachtet.

Gemessen wird die Produktivität des Mitarbeiters als Quotient aus geschaffenem Artefaktumfang und der dafür benötigten Zeit im betrachteten Zeitintervall. H19 ist zu akzeptieren, wenn in der zweiten Gruppe eine negative Korrelation zwischen Zeitdruck und Produktivität beobachtet werden kann. H20 ist zu akzeptieren, wenn in der ersten Gruppe eine positive Korrelation zwischen Zeitdruck und Produktivität zu beobachten ist.

Ergebnisse Abbildung 3.23 zeigt die Ergebnisse der Experimente. Für das kurze Zeitintervall ist einen Anstieg der Produktivität in Abhängigkeit des Zeitdrucks zu sehen, während für das lange

Zeitintervall eine Verringerung der Produktivität in Abhängigkeit des Zeitdrucks zu beobachten ist. Diese Beobachtungen wird von dazu passenden Korrelationskoeffizienten gestützt. Für das kurze Zeitintervall ergibt sich eine Rangkorrelation von $r = +0.88$ für die einzelnen Sets, und $r = +0.10$ für jeweils neu gewählte Rahmenparameter in jedem Durchlauf. Für das lange Zeitintervall beträgt der Korrelationskoeffizient für die Sets $r = -0.93$, und für die voneinander unabhängigen Durchläufe $r = -0.48$.



(a) 10 Tage (Sets) (b) 10 Tage (alle unabhängig) (c) 365 Tage (Sets) (d) 365 Tage (alle unabhängig)

Abbildung 3.23.: Produktivität in Abhängigkeit des Zeitdrucks

Die niedrigen Korrelationskoeffizienten, die sich ergeben, wenn alle Rahmenparameter in jedem Durchlauf vollständig neu gewählt werden, weisen auf Parameter hin, die einen stärkeren Einfluss als der Zeitdruck auf die Produktivität haben. Führt man die Experimente in einer veränderten Form durch, wobei der Energiewert, die Artefaktschwierigkeit und die Fähigkeiten des Mitarbeiters über die Durchläufe hinweg unverändert bleiben, so erhält man auch für kurze Zeitintervalle einen deutlich höheren Korrelationskoeffizienten von $r = +0.22$.

Betrachtet man die Qualität der Arbeitsleistung, so lässt sich zusätzlich feststellen, dass auch diese vom Zeitdruck abhängig ist. Entsprechende Korrelationskoeffizienten mit $r < -0.9$, sowohl für lange als auch für kurze Zeiträume mit erhöhten Zeitdruck bestätigen diese Beobachtung.

Damit sind beide Hypothesen zu akzeptieren. Die hier beschriebenen Experimente zeigen dieses Verhalten nur für zwei konkrete Zeiträume. Betrachtet man die Produktivität in Abhängigkeit der Zeit, so lässt sich feststellen, dass diese mit Erhöhung des Zeitintervalls, in dem ein bestimmter Zeitdruck herrscht, abnimmt. Abbildung 3.24 zeigt exemplarisch den Produktivitätsverlauf in Abhängigkeit der Länge des Zeitintervalls, für das der erhöhte Zeitdruck aufrechterhalten wird. Um die Daten dafür zu erhalten, wurde in einem weiteren Experiment das betrachtete Zeitintervall bei einem konstanten Zeitdruck variiert. Es lässt sich erkennen, dass auch bei keinem Zeitdruck die Produktivität mit der Zeit leicht abnimmt. Grund dafür ist der fehlende Erholungsurlaub, der in diesem Experiment nicht berücksichtigt wird. So ergibt sich eine leicht negative Energiebilanz, die sich entsprechend auf die Produktivität auswirkt (vgl. Abschnitt 2.4.1).

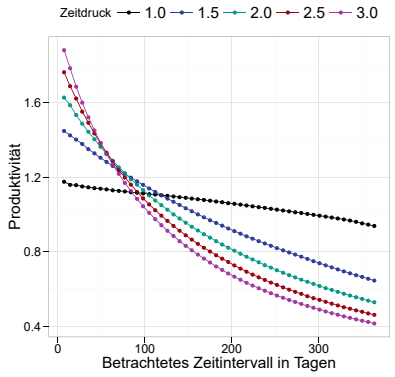


Abbildung 3.24.: Produktivität in Abhängigkeit der Wirkdauer von Zeitdruck

Hypothese 21: Einfluss von fehlendem Zeitdruck auf die Produktivität

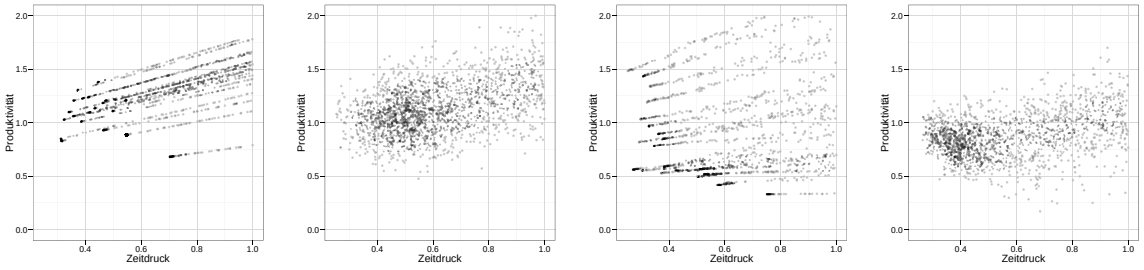
Je mehr freie Zeit ein Mitarbeiter hat, umso niedriger ist seine Produktivität.

Wird einem Mitarbeiter für eine Aufgabe mehr Zeit zur Verfügung gestellt als er dafür benötigt, so reduziert er typischerweise seine Arbeitsleistung bis auf durchschnittlich 60% [Abdel-Hamid, 1991] [Navarro, 2006]. Die übrige Zeit verwendet er für projektfremde und private Tätigkeiten.

Konzeption der Experimente Um zu zeigen, dass das Simulationsmodell diesen Effekt aufweist, werden vier Experimente konzipiert, die im Wesentlichen denen zur Untersuchung der Auswirkungen erhöhtem Zeitdrucks gleichen. Ein einzelner Mitarbeiter bearbeitet ein einzelnes Artefakt über einen bestimmten Zeitraum. Untersucht wird ein Zeitraum von 30 Tagen und ein Zeitraum von 365 Tagen in jeweils zwei Experimenten. Jeweils eines davon wird in 20 Sets à 100 Simulationsläufen, das andere in 2.000 unabhängigen Simulationsläufen durchgeführt. Variiert wird die freie Zeit, die dem Mitarbeiter zur Verfügung steht. Dazu wird der Zeitdruck anhand einer Gleichverteilung zufällig aus dem Intervall $[0, 1]$ gewählt. Alle anderen Rahmenparameter werden gemäß der Normalverteilung um ihren Standardwert zufällig gewählt.

Gemessen wird die Produktivität des Mitarbeiters als Quotient aus geschaffenem Artefaktumfang und der dafür benötigten Zeit im betrachteten Zeitintervall. Die Hypothese ist zu akzeptieren, wenn eine positive Korrelation zwischen Zeitdruck und Produktivität gezeigt werden kann.

Ergebnisse Abbildung 3.25 zeigt die Ergebnisse der Experimente. Für alle Experimente sind signifikant positive Korrelationen zwischen Zeitdruck und Produktivität festzustellen. Die Rangkorrelationskoeffizienten betragen für die Experimente mit Sets $r = +0.94$ für den kurzen Zeitraum und $r = +0.97$ für den langen Zeitraum. Für unabhängige Simulationsläufe ergibt sich $r = +0.39$ für den kurzen Zeitraum und $r = +0.30$ für den langen Zeitraum.



(a) 30 Tage (Sets) (b) 30 Tage (alle unabhängig) (c) 365 Tage (Sets) (d) 365 Tage (alle unabhängig)

Abbildung 3.25.: Produktivität in Abhängigkeit des fehlenden Zeitdrucks

Für die Experimente, die in Sets durchgeführt wurden, d. h. deren einzelne Messpunkte untereinander vergleichbar sind, kann der prozentuale Unterschied für die Produktivität bei einem Zeitdruck von 1 und bei einem Zeitdruck von 0 berechnet werden. Dieser beträgt für alle Sets ungefähr 60% und liegt damit in der oben geforderten Größenordnung.

Hypothese 22: Zusammenhang zwischen Kommunikation und Produktivität

Mitarbeiter, die an einer gemeinsamen Aufgabe arbeiten sind produktiver, wenn sie miteinander reden.

Mangelnde Kommunikation führt zu einer niedrigeren Produktivität und geringeren Softwarequalität [Curtis, 1988]. Arbeiten mehrere Mitarbeiter gemeinsam an einem Projekt, so ist es notwendig, dass sie sich untereinander austauschen, um Wissen und Erfahrungen zu teilen. Das Simulationsmodell erzeugt diesen Effekt, was mit den hier beschriebenen Experimenten gezeigt werden kann.

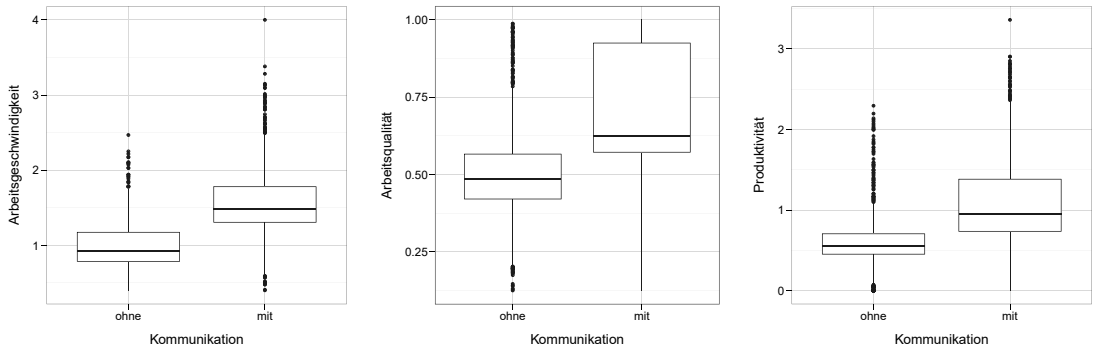
Konzeption der Experimente Dazu wird ein zufälliges Szenario gewählt, bei dem ein Team aus 2 bis 5 Mitarbeitern gemeinsam an einem Produkt arbeiten. Alle Parameter werden zufällig gemäß der Normalverteilung um ihren Standardwert gewählt. Es werden 1.000 solcher Parametersets erzeugt. Jedes der Parametersets wird einmal regulär simuliert. Beim zweiten Simulationslauf wird die Kommunikation im Team deaktiviert.

Der Anteil der Kommunikation wird durch das Verhalten der Mitarbeiter bestimmt. Derzeit enthält das dafür zuständige Verhaltensmodell keinen Parameter, über den sich dieser Anteil einstellen lässt. Dieses Experiment beschränkt sich deshalb darauf, die im Modell vorgesehene Kommunikation mit einer vollständigen Deaktivierung der Kommunikation zu vergleichen.

Gemessen wird die Arbeitsleistung in Form von Arbeitsgeschwindigkeit und Arbeitsqualität, sowie die Produktivität als Quotient aus geschaffenem Artefaktumfang und der dafür benötigten Zeit pro Mitarbeiter.

Die Hypothese ist zu akzeptieren, wenn nachgewiesen werden kann, dass sowohl Arbeitsgeschwindigkeit, als auch Arbeitsqualität, sowie die Produktivität mit Kommunikation höher ist als ohne.

Ergebnisse Abbildung 3.26 zeigt die Ergebnisse des Experiments. Wie der paarweise Wilcoxon-Test zeigt, ergeben sich signifikant höhere Werte ($p \ll 0.05$) für die Simulationsläufe bei denen die Mitarbeiter kommuniziert haben. Die Arbeitsgeschwindigkeit liegt mit Kommunikation im Mittel um 58% höher, die Qualität um durchschnittlich 0.2 Punkte und die Produktivität um 78%. Die Hypothese ist damit zu akzeptieren.



(a) Geschwindigkeit (b) Qualität (c) Produktivität

Abbildung 3.26.: Produktivität mit und ohne Kommunikation

Hypothese 23: Vorteile beim mehrfachen Erstellen von Artefakten

Wird ein Artefakt von einem Mitarbeiter zweimal erstellt, ist der Zeitaufwand für die erste Erstellung größer als für die zweite.

In manchen Situationen werden Artefakte mehrfach erstellt. Geht ein Artefakt beispielsweise durch einen Hardwareausfall verloren, so muss es erneut erstellt werden. In solchen Situationen ist es von Vorteil, dies den gleichen Autor machen zu lassen, da er sich mit dem Artefakt schon auskennt. Im Folgenden wird gezeigt, dass das Simulationsmodell diesen Effekt erzeugt.

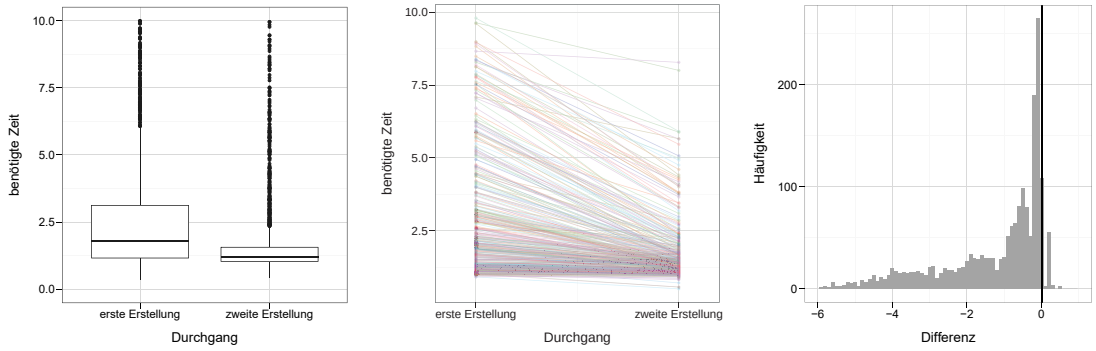
Konzeption der Experimente Es wird ein Experiment gestaltet, bei dem ein Artefakt durch einen zu beobachtenden Mitarbeiter zweimal erstellt wird. Dazu wird der Fortschritt des Artefakts, auf 0.0 zurückgesetzt, sowie alle Defekte aus dem Artefakt entfernt, sobald dieses das erste Mal vollständig ist. Die Eigenschaften des Mitarbeiters werden dabei nicht verändert. Insbesondere sein Artefaktwissen und seine Fähigkeiten bleiben erhalten. Das Szenario endet, sobald die Erstellung des Artefakts ein zweites Mal abgeschlossen wurde. Die Dauer der ersten und zweiten Erstellung wird getrennt gemessen. Da die Artefaktgröße zufällig gewählt wird, muss die Dauer in einen Wert pro Arbeitseinheit

umgerechnet werden. Aus den beiden Zeitmessungen wird die Differenz gebildet, um den zeitlichen Unterschied direkt auszudrücken.

Es werden 2.000 Simulationsläufe berechnet. Alle Rahmenparameter werden in jedem Simulationslauf gemäß der Normalverteilung um ihren Standardwert zufällig gewählt.

Die Hypothese ist zu akzeptieren, wenn ein signifikanter Unterschied zwischen dem ersten und zweiten Erstellungsdurchgang nachgewiesen werden kann, und der Mittelwert des zweiten Durchgangs kleiner ist als der des ersten.

Ergebnisse Abbildung 3.27 zeigt die Visualisierung der Messergebnisse. Links werden die gemessenen Zeiten des ersten und zweiten Durchgangs in Form eines Kastendiagramms gegenübergestellt. Hier ist zu erkennen, dass der zweite Durchgang im Mittel weniger Zeit in Anspruch genommen hat. Die mittlere Abbildung zeigt die einzelnen Messpunkte, die jeweils mit einer Linie verbunden sind. Es lässt sich erkennen, dass der erste Messpunkt bis auf wenige Ausnahmen einen höheren Wert annimmt als der zweite Messpunkt. Rechts ist die Häufigkeitsverteilung der Differenzen dargestellt. Hier lässt sich erkennen, dass der überwiegende Teil negativ ist, d. h. der zweite Durchgang der Artefakterstellung weniger Zeit in Anspruch genommen hat als der erste.



(a) Mittelwerte und Verteilung (b) direkter Vergleich der Erstellungszeiten (c) Häufigkeitsverteilung der Differenzen

Abbildung 3.27.: Produktivität mit und ohne Kommunikation

Die in Abbildung 3.27 gezeigten Ergebnisse lassen sich durch den Wilcoxon-Test bestätigen. Dieser zeigt, mit einer Signifikanz von $p \ll 0.05$, dass die für eine Arbeitseinheit benötigte Erstellungszeit im Mittel von 3.16h auf 1.91h gefallen ist, was einer Dauer von 60% des ersten Durchgangs entspricht. Die Hypothese ist somit zu akzeptieren.

Hypothese 24: Zusammenhang zwischen Artefaktumfang und Produktivität

Die Produktivität ist vom Umfang der Artefakte unabhängig.

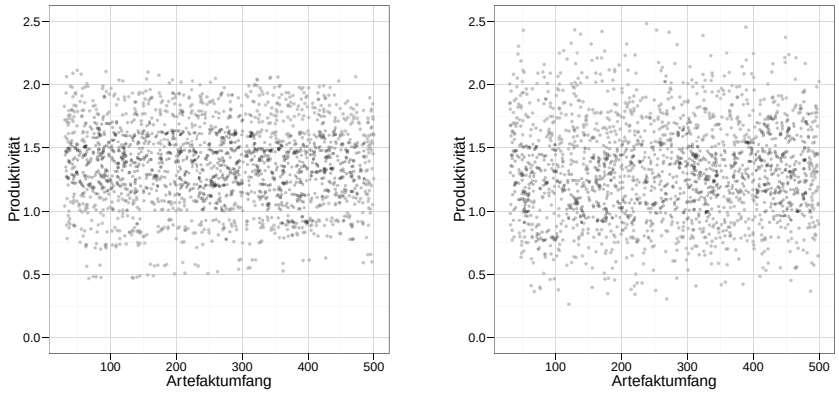
Eine gute Modularität hilft bei der Bewältigung der Komplexität. Eine niedrigere Komplexität wiederum führt zu einer höheren Produktivität [Tvedt, 1996]. Es ist daher wichtig, das zu erstellende Produkt gut zu strukturieren. Im Simulationsmodell wird dies jedoch nicht durch die Aufteilung in Artefakte realisiert, sondern durch die in den Artefakten angegebenen Schwierigkeitsgrade. Das ist notwendig, um dem Spielentwickler die Entscheidung zu überlassen, wie er sein Produkt aufbaut. Beispielsweise ob er das Pflichtenheft als einzelnes Artefakt behandelt, oder ob er es in seine einzelnen Teile, bis hin zu den einzelnen Anforderungen aufteilt. Nur so kann gewährleistet werden, dass das Simulationsmodell flexibel für die unterschiedlichsten Szenarien eingesetzt werden kann. Dementsprechend ist es notwendig, dass der Umfang der einzelnen Artefakte keinen Einfluss auf die Produktivität hat, mit dem diese entwickelt werden, da sonst die Granularität mit der das Produkt gestaltet wird nicht mehr frei wählbar wäre.

Konzeption der Experimente Um zu zeigen, dass der Artefaktumfang keinen Einfluss auf die Produktivität hat, wird ein entsprechendes Experiment durchgeführt, bei dem ein Mitarbeiter Artefakte unterschiedlichen Umfangs bearbeitet. Der Umfang wird dabei zufällig gemäß der Gleichverteilung aus dem Intervall $[30, 500]$ gewählt. Alle Rahmenparameter werden zufällig gemäß der Normalverteilung um ihren Standardwert gewählt. Das Experiment wird in 20 Sets à 100 Simulationsläufen durchgeführt. Ein weiteres Experiment wird in 2.000 unabhängigen Simulationsläufen durchgeführt.

Gemessen wird die Produktivität als Quotient aus geschaffenem Umfang und der dafür benötigten Zeit. Die Hypothese ist zu akzeptieren, wenn keine Korrelation zwischen Artefaktumfang und Produktivität festgestellt werden kann.

Ergebnisse Abbildung 3.28 zeigt die Ergebnisse der Experimente. Weder bei der Durchführung in Sets, noch bei einer Durchführung als unabhängige Simulationsläufe ist ein Zusammenhang zwischen Artefaktumfang und Produktivität zu erkennen.

Die Korrelationsanalyse zwischen Artefaktumfang und Produktivität ergibt einen Rangkorrelationskoeffizienten von $r = -0.019$ mit $p = 0.38$ (Sets) bzw. $r = +0.0025$ mit $p = 0.91$ (alle unabhängig). Somit muss an dieser Stelle die Nullhypothese angenommen werden, d. h. es kann nicht von einem Zusammenhang zwischen Artefaktumfang und Produktivität ausgegangen werden.



(a) Sets (b) alle unabhängig

Abbildung 3.28.: Produktivität in Abhängigkeit des Artefaktumfangs

Hypothese 25: Vorteile parallelen Arbeitens

Je mehr Aufgaben parallel bearbeitet werden können, um so kürzer ist die Projektlaufzeit, vorausgesetzt, es stehen genügend geeignete Mitarbeiter zur Verfügung.

Ein gutes Projektmanagement zeichnet sich mitunter dadurch aus, dass die Arbeitslast geschickt in voneinander unabhängige Aufgaben aufgeteilt wird, die parallel und verteilt bearbeitet werden können. Durch eine gute Parallelisierung der Arbeitsvorgänge kann die Projektdauer gesenkt werden [Tvedt, 1996]. Es ist wichtig, dass das Simulationsmodell das parallele Bearbeiten von Aufgaben korrekt umsetzt.

Konzeption der Experimente Um dies zu zeigen, wird ein Experiment gestaltet, bei dem ein Produkt aus mehreren unabhängigen Artefakten entwickelt wird. Jedes Artefakt ermöglicht dabei eine unabhängige Aufgabe. Es werden Anzahlen von Artefakten zwischen 1 und 8 untersucht und miteinander verglichen. Dazu werden je Anzahl 100 Simulationsläufe mit Rahmenparametern, die gemäß der Normalverteilung um ihren Standardwert zufällig gewählt werden, berechnet. Um sicherzustellen, dass genügend Mitarbeiter verfügbar sind, umfasst das Projektteam 8 Projektbeteiligte, die die Aufgaben unter sich aufteilen. Je nach Aufgabenzahl werden Aufgaben auch von mehreren Mitarbeitern gemeinsam bearbeitet. Alle Mitarbeiter sind somit immer ausgelastet.

Gemessen wird die Projektdauer in Stunden pro Arbeitseinheit. Die Hypothese ist zu akzeptieren, wenn eine negative Korrelation zwischen der Anzahl an parallel zu bearbeitenden Aufgaben und der dafür benötigten Zeit gezeigt werden kann.

Ergebnisse Abbildung 3.29 zeigt die Ergebnisse des Experiments. Hier ist zu beobachten, dass die benötigte Zeit bei einer Verdopplung der Anzahl paralleler Aufgaben um ca. 40% sinkt. Der dazugehörige Rangkorrelationskoeffizient beträgt $r = -0.94$ bei einer Signifikanz von $p \ll 0.05$. Die Hypothese ist somit zu akzeptieren.

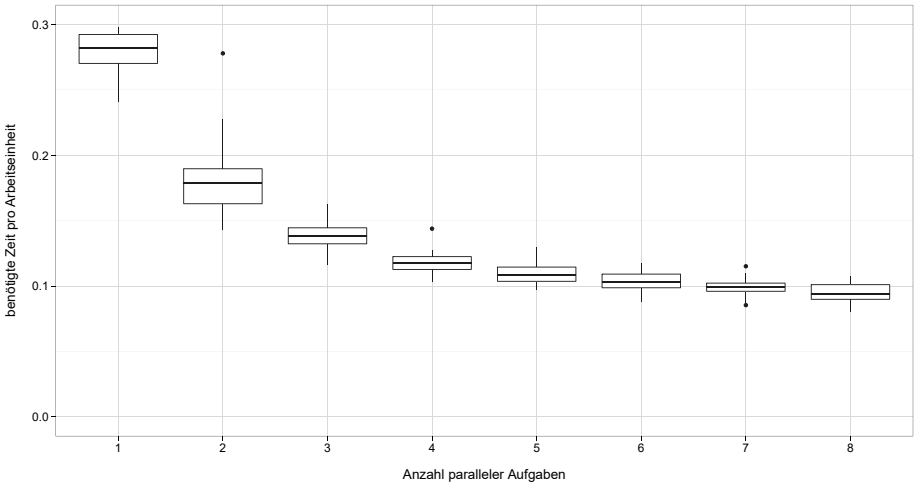


Abbildung 3.29.: Benötigte Zeit in Abhängigkeit der Anzahl paralleler Aufgaben

Um zusätzlich den Einfluss der Teamgröße auf den Zeitgewinn zu untersuchen, wird ein weiteres Experiment mit unterschiedlichen Teamgrößen zwischen 1 und 8 Teammitgliedern durchgeführt. Abbildung 3.30 zeigt die dazu gehörigen Ergebnisse. Tabelle 3.8 enthält die dazugehörigen Medianwerte, die relativ zum Wert für jeweils eine parallele Aufgabe angegeben sind. Hier ist zu erkennen, dass der oben beschriebene Effekt nur solange auftritt, bis die Anzahl der parallel zu bearbeitenden Aufgaben der Teamgröße entspricht. Es hat keinen Vorteil mehr Aufgaben bereitzustellen, als Mitarbeiter verfügbar sind, die diese Aufgaben bearbeiten.

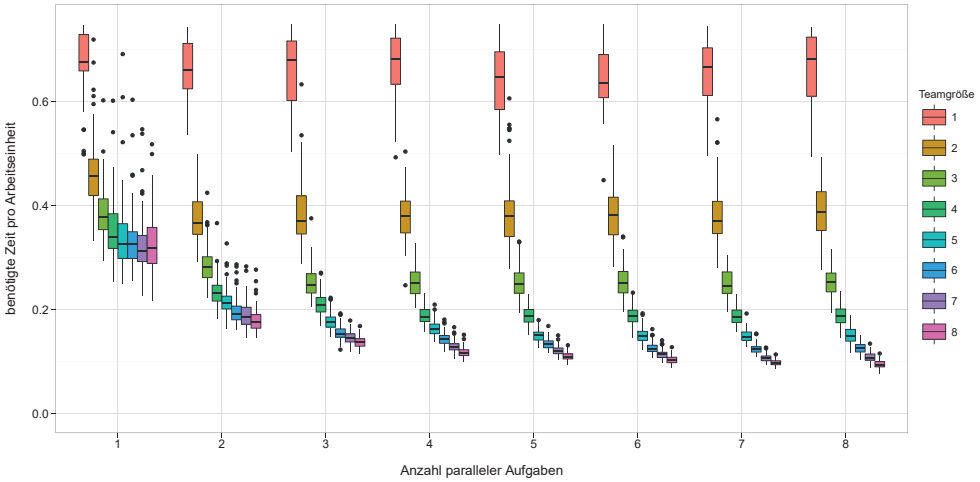


Abbildung 3.30.: Benötigte Zeit in Abhängigkeit Anzahl paralleler Aufgaben und der Teamgröße

Teamgröße	Anzahl paralleler Aufgaben							
	1	2	3	4	5	6	7	8
1	1.0	0.96	0.98	0.99	0.99	0.97	0.93	0.98
2	1.0	0.80	0.80	0.83	0.83	0.83	0.81	0.84
3	1.0	0.74	0.65	0.65	0.65	0.65	0.64	0.67
4	1.0	0.67	0.61	0.54	0.54	0.55	0.54	0.55
5	1.0	0.65	0.54	0.50	0.46	0.45	0.45	0.45
6	1.0	0.59	0.47	0.44	0.41	0.38	0.38	0.38
7	1.0	0.59	0.46	0.40	0.38	0.36	0.34	0.34
8	1.0	0.55	0.43	0.36	0.34	0.32	0.30	0.29

Tabelle 3.8.: Benötigte Zeit in Abhängigkeit Anzahl paralleler Aufgaben und der Teamgröße

3.4.2. Weitere Analyse von Parameterabhängigkeiten

Die gezeigten Experimente prüfen, ob die wichtigsten Effekte im Simulationsmodell korrekt abgebildet werden. Sie können jedoch nicht alle möglichen Zusammenhänge, die potentiell zwischen den einzelnen Parametern bestehen, betrachten.

Um unerwartetes Verhalten und Verhaltensanomalien aufzudecken, werden eine Reihe von zusätzlichen Experimenten durchgeführt. Dabei wird der Einfluss der relevanten Szenarioparameter auf das Simulationsergebnis untersucht. Im Vergleich zur Prüfung der Hypothesen, steht dabei nicht der Nachweis von erwünschten Effekten im Vordergrund, sondern die Identifikation von unerwünschten, oder falschen Effekten, die auf Fehler im Simulationsmodell hindeuten.

Dazu werden die Zusammenhänge aller relevanter Parameter des Modells untersucht. Diese Parameter werden in Eingabe- und Ausgabeparameter unterteilt. *Eingabeparameter* sind Eigenschaften des Szenarios, die so das Simulationsmodell steuern. *Ausgabeparameter* sind Eigenschaften des Szenarios, des Simulationsverlaufs oder des Simulationsergebnisses, die sich durch die Ausführung der Simulation ergeben.

Die untersuchten Parameter sind in Tabelle 3.9 und Tabelle 3.10 aufgeführt. Eine detaillierte Beschreibung der Parameter, ihrer Wertebereiche und wie sie gesetzt bzw. gemessen werden, findet sich in Anhang A.4.

Mitarbeiter	Produkt	Aktivitäten	Projekt
• Energieniveau • Motivation • Fähigkeiten • Artefaktwissen • Charaktereigenschaften: – Adaptionsfähigkeit – Rednerfähigkeit – physische Ausdauer – Zeitdruckschranke – Lernfähigkeit – Wechselfähigkeit – Vergessensrate – Aktivitätssympathie – Anspruch (Hyg.) – Anspruch (Mot.) – Mitarbeitersympathie	• Schwierigkeit • durchschnittlicher Artefaktumfang • Einfluss zusätzlichen Wissens	• Fortschritt – Quantitätsmodifikator – Qualitätsmodifikator • Qualitätssicherung – Quantitätsmodifikator – Qualitätsmodifikator • Kommunikation – Quantitätsmodifikator	• Teamgröße • Zeitdruck

Tabelle 3.9.: Liste der untersuchten Eingabeparameter

Mitarbeiter	Produkt	Aktivitäten	Projekt
• Energieänderung • Motivation • Fähigkeitsänderung • Wissensänderung	• Fortschritts-änderung • Qualitätsänderung	• Fortschritt – Geschwindigkeit – Qualität • Qualitätssicherung – Geschwindigkeit – Qualität • Kommunikation – Geschwindigkeit	• Zeit pro Arbeitseinheit • Zeit pro Arbeitseinheit und Mitarbeiter • Produktivität • Produktivität pro Mitarbeiter

Tabelle 3.10.: Liste der untersuchten Ausgabeparameter

Variation eines einzelnen Parameters

Um die Zusammenhänge zu evaluieren, wird zu jedem Eingabeparameter ein eigenes Experiment mit 20 Sets à 50 Simulationsläufen durchgeführt. Der zu untersuchende Parameter wird dabei innerhalb des Sets variiert, indem er für jeden Simulationslauf neu aus dem jeweils für ihn gültigen Wertebereich zufällig ausgewählt wird. Das Projektteam besteht aus 1 bis 6 Mitarbeitern. Die genaue Anzahl wird zufällig pro Set gewählt. Die restlichen Parameter des Szenarios werden zufällig gemäß der Normalverteilung um ihren Standardwert gewählt.

Die Struktur des zu erstellenden Produkts wird ebenfalls zufällig erzeugt. Es besteht aus 1 bis 20 Artefakten; die konkrete Anzahl wird zufällig gewählt. Die Abhängigkeiten zwischen den Artefakten, sowie die Vorbedingungen, als auch die Kontextabhängigkeiten, werden ebenfalls zufällig gewählt. Die relative Vollständigkeit der einzelnen Artefakte wird zufällig und gleichverteilt aus dem Intervall $[0, 1]$ gewählt. So werden nicht nur Projekte in der Anfangsphase, sondern aus allen Phasen der Entwicklung betrachtet.

Die Aufgabe des Projektleiters übernimmt ein zusätzlicher Agent, der den Projektplan erstellt und kontinuierlich aktualisiert. Er entspricht im Wesentlichen dem Agenten, der bei den Experimenten zur kontinuierlichen Qualitätssicherung (vgl. Hypothese 17) eingesetzt wurde. Er kontrolliert die Qualität der zu bearbeitenden Artefakte und entscheidet dynamisch, ob Fortschritts- oder Qualitätssicherungsaktivitäten angewendet werden sollen. Die zu bearbeitenden Artefakte werden von ihm zufällig, jedoch unter Berücksichtigung ihrer Vorbedingungen ausgewählt, und auf die einzelnen Mitarbeiter verteilt. Dabei wird darauf geachtet, eine möglichst hohe Parallelisierung der Aufgaben zu erreichen.

Der untersuchte Simulationszeitraum umfasst 5 Monate. Wird ein Projekt vorher abgeschlossen, endet der Zeitraum entsprechend früher.

Nach Ausführung jeden Experiments wird der Rangkorrelationskoeffizient zwischen dem jeweiligen untersuchten Eingabeparameter und aller gemessenen Ausgabeparameter berechnet.

Auswertung und Ergebnisse Aus den durch die Experimente gewonnenen Daten wurde für jedes Parameterpaar der Rangkorrelationskoeffizient berechnet, und ein entsprechendes Streudiagramm erzeugt. Abbildung 3.31 zeigt ein Beispiel für ein solches Diagramm. Anhand der Streudiagramme wurden alle berechneten Korrelationen von Hand kontrolliert. Hierbei wurden insbesondere die Fälle behandelt, bei denen sich zwar ein signifikanter Koeffizient ergeben hat, jedoch kein Effekt erkennbar ist. Solche Ergebnisse entstehen, da die Rangkorrelation keine Intervalle zwischen den Messwerten berücksichtigt. Um eine Korrelation zu erreichen, genügt es, dass die Messwerte die richtige Rangfolge aufweisen. Der tatsächliche Unterschied zwischen den einzelnen Werten wird dabei nicht miteinbezogen.

Vor Beginn der Experimente wurde für jeden potentiellen Zusammenhang eine entsprechende Einschätzung gemacht. Dabei wurde festgelegt, ob ein positiver ($r \geq +0.1$), negativer ($r \leq -0.1$), oder kein Zusammenhang ($-0.1 < r < +0.1$) zwischen zwei Parametern erwartet wird. Die dafür geforderte Signifikanz liegt bei $p \leq 0.05$. Alle Ergebnisse der Experimente wurden mit dieser Einschätzung

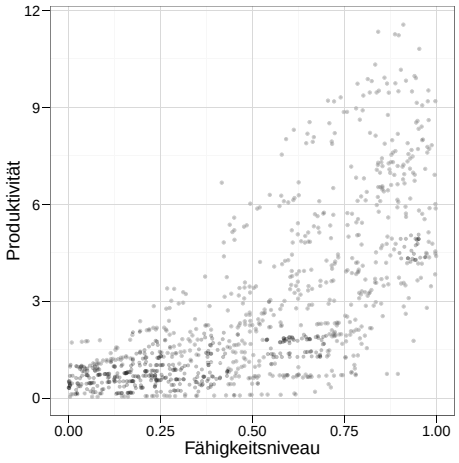


Abbildung 3.31.: Streudiagramm über den Zusammenhang zwischen Fähigkeit und Produktivität im Rahmen der weiteren Analyse der Parameterabhängigkeiten

verglichen. Tabelle 3.11 zeigt die Ergebnisse der Experimente im Vergleich zur Einschätzung. Bestätigte positive bzw. negative Zusammenhänge sind mit einem farbigen Punkt markiert. Fehlende oder unerwartete Zusammenhänge werden durch ein Dreieck gekennzeichnet.

Im Wesentlichen decken sich die Ergebnisse der Experimente mit den anfänglich gemachten Einschätzungen. Das Simulationsmodell verhält sich so, wie es vorgesehen ist. Die Überprüfung, ob diese Annahmen korrekt sind, ist nicht Aufgabe dieses Evaluationsschritts, sondern wird hauptsächlich mit der Überprüfung der aufgestellten Hypothesen abgedeckt. Bei der Analyse der Ergebnisse haben sich jedoch auch einige Auffälligkeiten ergeben:

- Es wurde erwartet, dass sich vorhandenes bzw. fehlendes Artefaktwissen nicht nur auf die Arbeitsleistung der Fortschrittsaktivitäten, sondern auch auf die der Qualitätssicherungsaktivitäten auswirkt. Ein solcher Zusammenhang hat sich jedoch zunächst nicht gezeigt. Im ursprünglichen Experiment wurden Testaktivitäten zur Qualitätssicherung verwendet. Die verwendeten Black-Box-Tests benötigen laut ihrer Definition nur einen geringen Grad an Artefaktwissen, da hier lediglich die Schnittstellen relevant sind. In einem ergänzenden Experiment wurden Inspektionen zur Qualitätssicherung eingesetzt, die mehr Artefaktwissen voraussetzen. In diesen Experimenten hat sich der positive Zusammenhang zwischen Artefaktwissen und Arbeitsgeschwindigkeit bzw. Qualität bei Qualitätssicherungsaktivitäten wie erwartet gezeigt.

- Eine ähnliche Situation, wie im vorherigen Punkt besprochen, hat sich für den Zusammenhang zwischen Adaptionsfähigkeit und Arbeitsleistung bei Qualitätssicherungsaktivitäten ergeben. Die Adaptionsfähigkeit wirkt sich auf das erworbene Artefaktwissen aus. Dieses wirkt sich wiederum auf die Arbeitsleistung aus, sodass hier ein entsprechender Zusammenhang zu erwarten ist. Auch hier hat sich der erwartete Effekt für die Qualitätssicherungsaktivitäten erst in einem zusätzlichen Experiment mit Inspektions- statt Testaktivitäten gezeigt.

- Es hat sich ein unerwarteter positiver Zusammenhang zwischen Adaptionsfähigkeit und dem Erwerb von Fähigkeiten gezeigt. Die Adaptionsfähigkeit beschränkt sich auf Artefaktwissen. Trotzdem wirkt sie sich indirekt auf den Fähigkeitserwerb aus, da ein höheres Produktwissen eine bessere Arbeitsleistung, und damit auch einen höheren Lernfortschritt verursacht. Zusätzlich wird durch eine höhere Adaptionsfähigkeit weniger Zeit für das Erlernen fehlenden Produktwissens benötigt, die wiederum in Aktivitäten mit Lerneffekt investiert werden kann.

- Analog zur Adaptionsfähigkeit wirkt sich auch die Rednerfähigkeit auf den Erwerb von Produktwissen aus. Entsprechend ergibt sich hier ein indirekter Zusammenhang zwischen Rednerfähigkeit und Fähigkeitsänderung. Ein Zusammenhang mit der Arbeitsleistung bei

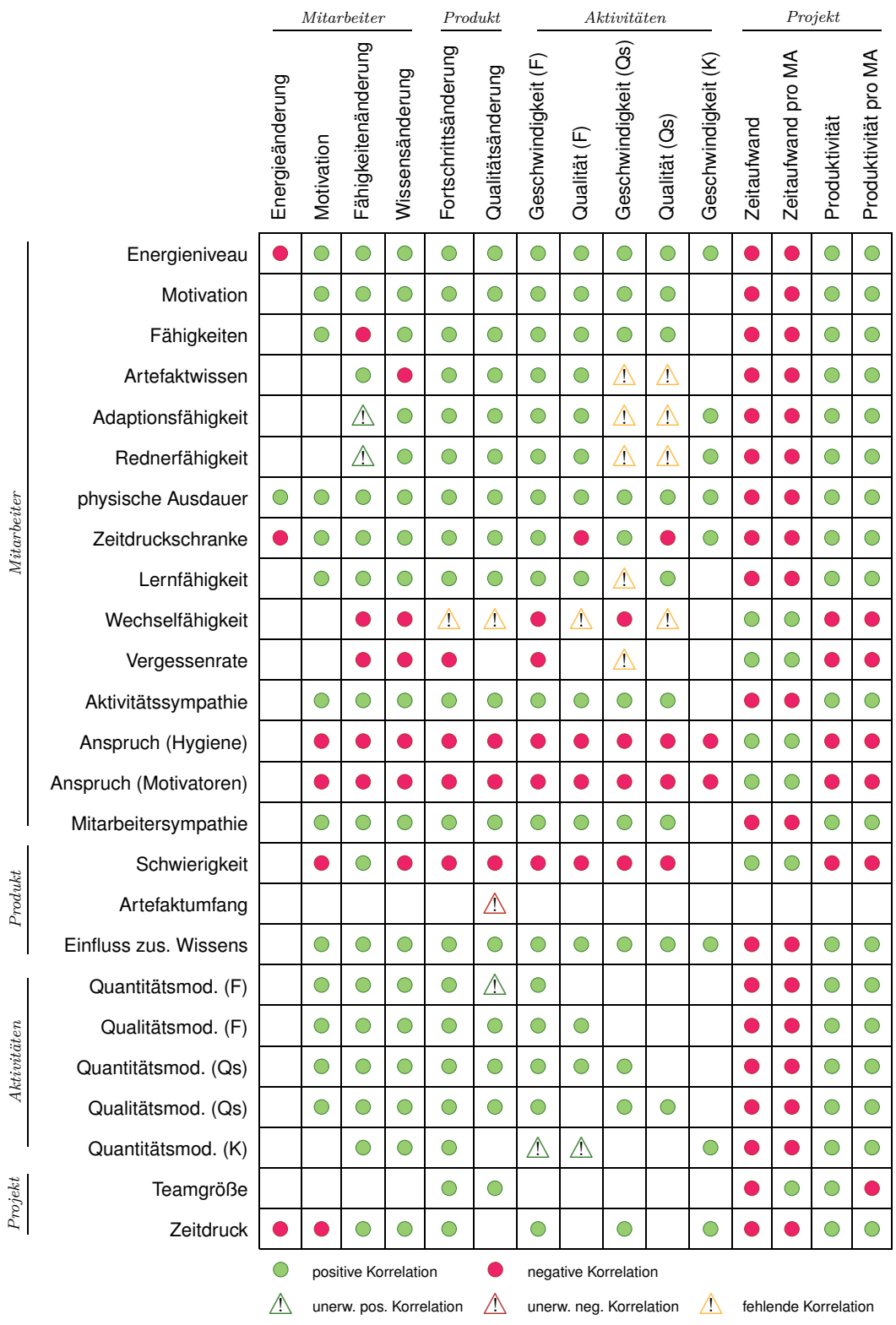


Tabelle 3.11.: Ergebnis der Zusammenhangsanalyse der unterschiedlichen Parameter

Qualitätssicherungsaktivitäten hat sich ebenfalls nur bei Inspektions- nicht jedoch bei der Testaktivität gezeigt.

- Zwischen Lernfähigkeit und der Geschwindigkeit von Qualitätssicherungsaktivitäten wurde ein indirekter Effekt vermutet, der jedoch nicht beobachtet werden konnte. Eine höhere Lernfähigkeit steigert die Aktivitätsfähigkeit, die sich wiederum auf die Arbeitsgeschwindigkeit auswirkt. Für Fortschrittsaktivitäten ist dieser Effekt sichtbar. Die beobachtete Projektzeit von 5 Monaten reicht jedoch nicht aus, um einen entsprechend großen Lernfortschritt zu erzeugen, der auch in der Geschwindigkeit der Qualitätssicherungsaktivitäten sichtbar wird. Da der Anteil dieser Aktivitäten im Vergleich zu den Fortschrittsaktivitäten geringer ist, ist für Fortschrittaktivitäten der Lernfortschritt in gleicher Zeit entsprechend höher. Für Projektlaufzeiten ab ca. 1 Jahr wird der Effekt auch für Qualitätssicherungsaktivitäten sichtbar.

- Durch die Untersuchung der Wechselfähigkeit wurde ein kleiner Fehler im Simulationsmodell aufgedeckt, der die Berechnung der Wechselzeiten zwischen verschiedenen Aufgaben betrifft. Dieser wurde korrigiert und alle Experimente anschließend erneut durchgeführt. Tabelle 3.11 enthält die Ergebnisse nach der Fehlerbehebung. Da eine niedrigere Wechselfähigkeit zu einem schnelleren Aufgabenwechsel führt, war zu erwarten, dass sich ein höherer Fähigkeitswert negativ auf Fortschritt und Qualität auswirkt. Das hat sich jedoch zunächst für vier Messwerte nicht gezeigt. Grund dafür ist der verwendete Projektmanagement-Agent, der die Aufgaben möglichst optimal bezüglich unnötiger Aufgabenwechsel verteilt. In einem weiteren Experiment wurde diese Optimierung entfernt. In diesem Experiment konnten auch die vier fehlenden Zusammenhänge beobachtet werden.

- Ein Zusammenhang zwischen Vergessensrate und Geschwindigkeit der Qualitätssicherungsaktivitäten konnte nicht gezeigt werden. Der Effekt ist zu schwach, da die Testaktivitäten nur geringes Produktwissen benötigen.

- Bei der Untersuchung der Mitarbeitersympathie hat sich zunächst nur ein Zusammenhang mit der Motivation gezeigt. Der Projektmanagement-Agent verteilt die einzelnen Mitarbeiter möglichst gut auf die einzelnen Aktivitäten. Dadurch arbeiten selten Mitarbeiter gemeinsam an einer Aufgabe. In einem weiteren Experiment wurden alle Aufgaben gemeinsam bearbeitet. Die erwarteten Effekte konnten so gezeigt werden. Die in Tabelle 3.11 gezeigten Ergebnisse stammen aus diesem zweiten Experiment.

- Nach Hypothese 24 hat der Umfang der Artefakte keinen Einfluss auf die Simulation. Entsprechend werden hier keinerlei Zusammenhänge erwartet. Trotzdem konnte ein eindeutiger Zusammenhang zwischen Umfang und Qualität des Produkts nachgewiesen werden. Die Qualität des Gesamtprodukts wird als Durchschnitt über alle Artefakt, gewichtet mit deren jeweiligem Umfang gemessen. Dadurch verändert sich diese bei einem großen Produkt vergleichsweise gering. Während bei einem kleinen Produkt der gleiche Umfang an hinzugekommenen oder entfernten Defekten eine große Wirkung hat, ändert sich bei einem großen Produkt dadurch nur wenig, da im Vergleich zum kleinen Produkt nur ein geringer Teil überhaupt bearbeitet wird. Die unerwartete Korrelation ist somit ein Problem der Messmethode und nicht des Modells. Ein weiteres Experiment, in dem die absolute Veränderung der Defekte betrachtet wurde, zeigt, dass diese vom Umfang des Produkts unabhängig ist.

- Bei der Variation des Quantitätsmodifikators der Fortschrittsaktivitäten hat sich ein unerwarteter Zusammenhang zwischen dem Modifikator und der Qualitätsänderung bezogen auf das Gesamtprodukt gezeigt. Durch die höhere Bearbeitungsgeschwindigkeit kann verhältnismäßig mehr Zeit in die Qualitätssicherungsaktivitäten investiert werden, was wiederum zu einer höheren Produktqualität führt. Auf die Arbeitsleistung der Qualitätssicherungsaktivitäten an sich, hat der Modifikator keine Auswirkungen.

- Der Quantitätsmodifikator für Kommunikationsaktivitäten beeinflusst direkt das Produktwissen der Mitarbeiter. Da Produktwissen insbesondere die Fortschrittsaktivitäten beeinflusst,

besteht ein indirekter Zusammenhang zwischen dem Modifikator und der Leistung von Fortschrittsaktivitäten, der so nicht erwartet wurde.

Bis auf den Fehler im Rahmen der Berechnung der Aufgabenwechselzeiten, der entsprechend behoben wurde, hat die Analyse zwar einige unerwartete Auffälligkeiten aufgezeigt, jedoch beeinträchtigt keine davon die Validität des Simulationsmodells. Die Auswertung der Experimente hat außerdem gezeigt, dass viele Effekte vom gegebenen Szenario, insbesondere auch von der konkreten Projektführung durch den Projektleiter abhängig sind.

Es gilt zu berücksichtigen, dass die identifizierten Zusammenhänge nicht grundsätzlich und uneingeschränkt gelten. Die Aussagen beziehen sich lediglich auf die Mehrheit der möglichen Szenarien. Für nahezu jeden Zusammenhang gibt es Konfigurationen, für die dieser nicht gilt, oder sogar ein gegenteiliger Effekt gezeigt werden kann. Ein Beispiel hierfür sind die Zeitdruckeffekte, die im Rahmen von Hypothese 19 und 20 besprochen wurden. Ob sich Zeitdruck positiv oder negativ auf die Produktivität auswirkt, hängt maßgeblich von der Dauer ab, über die er aufrechterhalten wird.

Im Rahmen der in diesem Abschnitt beschriebenen Experimente wurden auch die Extremwerte der einzelnen Eingabeparameter überprüft. Hier haben sich keinerlei Besonderheiten oder Unstetigkeiten im Simulationsverlauf oder dessen Ergebnissen ergeben.

Variation eines Parameterpaars und vollständige Variation aller Parameter

Die zuvor ausgeführten Experimente haben gezeigt, dass einige Effekte nur in bestimmten Situationen auftreten. Um unerwünschte Effekte, die durch das Zusammenspiel zweier Parameter entstehen, aufzudecken, wurde eine zweite Reihe von Experimenten durchgeführt, um die Ergebnisse der vorherigen Untersuchung zu bestätigen. Die Experimente entsprechen dabei im Wesentlichen denen zur Variation eines einzelnen Parameters. Anstelle des einzelnen Parameters werden jedoch zwei Parameter gleichzeitig variiert, indem deren Belegung jeweils in jedem Simulationslauf zufällig neu gewählt wird. Zu jedem möglichen Parameterpaar wird ein Experiment mit 10 Sets à 50 Durchläufen durchgeführt.

In einem weiteren Experiment werden alle Parameter in jedem Simulationslauf zufällig neu gewählt werden. Es werden 10.000 Durchläufe berechnet. Mit diesem Experiment sollen bislang noch unentdeckte Auffälligkeiten aufgedeckt werden, die von mehr als zwei Parametern abhängig sind.

Zu jedem Experiment und zu jedem Parameterpaar wurde der Rangkorrelationskoeffizient berechnet und, falls $p < 0.05 \wedge |r| \geq 0.1$, mit den Ergebnissen aus Tabelle 3.11 verglichen. Dabei wurden 61 Widersprüche festgestellt. Hier ist abermals zu berücksichtigen, dass die Zusammenhänge lediglich für die Mehrheit der Szenarien gelten können. Bei der geringen Zahl von 10 berechneten Sets kommt es daher vor, dass zufälligerweise mehrere Szenarien gewählt wurden, die den Effekt nicht aufweisen oder sogar einen gegensätzlichen Effekt erzeugen. Alle Widersprüche wurden einzeln von Hand anhand der jeweiligen Streudiagramme überprüft. In allen Fällen hat sich dabei herausgestellt, dass die mittels der Korrelationsanalyse identifizierten widersprüchlichen Effekte rein rechnerisch sind und kein erkennbarer Effekt vorliegt. Es konnte somit kein Zusammenhang identifiziert werden, der den in Tabelle 3.11 beschriebenen Ergebnissen widerspricht.

3.5. Betrachtung der Validitätsgüte

Um sicherzustellen, dass die Ergebnisse der Evaluation auch belastbar sind, ist es wichtig, die Einhaltung unterschiedlicher Gütekriterien sicherzustellen. Im Folgenden werden die für die im Rahmen dieser Arbeit durchgeführten Evaluation relevanten Aspekte der Validität [Mitchell, 2004] diskutiert.

Konstruktvalidität

Die *Konstruktvalidität* klärt, ob die Gestaltung eines Experiments adäquat ist, und sich die gewünschten Eigenschaften anhand der gewählten Indikatoren messen lassen. Dabei ist es wichtig, dass das Experiment ein zu messendes Konstrukt in all seinen Aspekten erfasst.

Im Simulationsmodell liegen die betrachteten Aspekte entweder direkt als Parameter vor oder können durch einfache Berechnungen aus diesen abgeleitet werden. Da die Zahl der Parameter beschränkt ist, kann durch eine einfache Betrachtung der Parameter festgestellt werden, ob alle relevanten Aspekte durch das Experiment erfasst werden. Grundlage dafür bildet die Definition der Parameter, die im Rahmen der Entwicklung des Simulationsmodells festgelegt wurde. Um das Simulationsmodell möglichst einfach zu halten, werden die meisten Aspekte nur durch einzelne Parameters beschrieben. Die Gültigkeit ist daher leicht anhand der Beschreibung der einzelnen Experimente, insbesondere der dort gemessenen Parameter, in Verbindung mit der Definition des Datenmodells des Simulationsmodells sicherzustellen.

Ein weiterer wichtiger Indikator ist die Konsistenz zwischen unterschiedlich durchgeführten Messungen des gleichen Aspekts. Diese liegt vor, wenn unterschiedliche Methoden zur Messung eines Konstrukts in ihren Ergebnissen übereinstimmen. Im Rahmen der Evaluation des Simulationsmodells wurden eine Vielzahl unterschiedlicher Experimente auf verschiedene Arten durchgeführt. Die dort betrachteten Aspekte werden, soweit möglich, auf mehrere Arten gemessen. So werden beispielsweise in vielen Experimenten sowohl die Arbeitsleistung der einzelnen Aktivitäten, als auch die Produktivität über das gesamte Projekt hinweg betrachtet. Innerhalb jedes einzelnen Experiments wird gezeigt, dass die unterschiedlichen Messungen eines gemeinsamen Aspekts in ihren Ergebnissen übereinstimmen. Über alle Experimente hinweg wurde betrachtet, ob sich deren Ergebnisse widersprechen. Dazu werden alle vorliegenden Ergebnisse miteinander auf ihre Konsistenz hin überprüft. Insbesondere alle festgestellten Korrelationen wurden gesammelt und automatisiert miteinander verglichen, um Widersprüche aufzudecken.

Statistische Validität

Die *statistische Validität* erreicht einen hohen Grad, wenn Reliabilität und Teststärke der Messverfahren und angewendeten statistischen Verfahren groß ist. Außerdem beeinträchtigen auch Messfehler die statistische Validität.

Um die statistische Validität bei der Evaluation des Simulationsmodells sicherzustellen, wurden nur etablierte statistische Verfahren eingesetzt, die durch die empirische Forschung erprobt sind. Das Simulationsmodell wird damit genauso untersucht, wie es auch mit dem Original möglich wäre. Lediglich die Messung einiger Parameter müsste im Original über Umwege geschehen.

Durch die direkte Extraktion der Messwerte aus dem Simulationsmodell beschränkt sich die Fehlervarianz der Messungen auf numerische Ungenauigkeiten bei der Umrechnung von Simulationsparameter in Messergebnisse, die zu vernachlässigen sind. Die Messwerte werden nachträglich nicht verändert. Es findet weder eine Eliminierung von Ausreißern, noch eine andere Form der Filterung der Messergebnisse statt. Lediglich Datensätze mit numerisch ungültige Werte, wie sie beispielsweise bei einer Division durch 0 entstehen, werden aussortiert. Solche Probleme entstehen beispielsweise, wenn durch eine ungünstige Belegung der Rahmenparameter kein Arbeitsfortschritt im Szenario erzeugt wird, und damit auch keine Messwerte für die Arbeitsgeschwindigkeit, etc. vorliegen.

Zur Berechnung der Kenngrößen wurden Skripte der Sprache *R* verwendet, die speziell für statistische Berechnungen entwickelt wurde. Dabei wurde auf die dort integrierten Funktionen und die etablierte Bibliothek `Hmisc` zurückgegriffen, welche die Funktion `rcorr` für die Berechnung der Korrelationskoeffizienten enthält. Eigene zusätzliche Funktionen zur Berechnung statistischer Kenngrößen aus den Messdaten wurden nicht eingeführt. Fehler bei deren Berechnungen sind daher unwahrscheinlich.

Um die Korrelation zwischen Merkmalen zu berechnen, wurde die Rangkorrelation nach Spearman verwendet. Diese setzt im Gegensatz zur Korrelation nach Pearson keine Normalverteilung der Daten für die Signifikanzberechnung voraus. Da bei der Berechnung der Rangkorrelation die Informationen über den tatsächlichen Unterschied der Messwerte zueinander verloren gehen, wurden zusätzlich auch die Streudiagramme ausgewertet, um sicherzustellen, dass ein sichtbarer Effekt vorhanden ist.

Der zur Untersuchung der Differenz zwischen zwei Gruppen verwendete Wilcoxon-Vorzeichen-Rang-Test setzt im Gegensatz zum t-Test ebenfalls keine Normalverteilung der Daten voraus. Auch hier wurde immer auch die Effektstärke in Form des Mittelwerts der Differenz der paarweise in den beiden Gruppen gemessenen Parametern berücksichtigt, um sicherzustellen, dass ein sichtbarer Effekt vorhanden ist.

Interne Validität

Interne Validität liegt vor, wenn die gefundenen Effekte tatsächlich auf die dafür angenommenen Ursachen zurückzuführen sind. Dazu müssen alternative Erklärungen möglichst umfassend ausgeschlossen werden. Nur so kann gewährleistet werden, dass die Veränderung abhängiger Variable auf die Variation der unabhängigen Variablen zurückzuführen ist.

Die interne Validität lässt sich hauptsächlich durch den Ausschluss von Störvariablen erreichen. Ziel dabei ist es, die Experimente so zu gestalten, dass die zu erwartenden Effekte nur durch die untersuchten unabhängigen Variablen beeinflusst werden können. Im Gegensatz zu vielen empirischen Untersuchungen, findet die Evaluation des Simulationsmodells in einer stark kontrollierten Umgebung statt. Dadurch können die häufigsten Gefährdungsfaktoren der internen Validität, wie sie bei empirischen Experimenten auftreten, ausgeschlossen werden:

- Die unabhängigen Variablen können in den allermeisten Fällen, eine Ausnahme ist hier die Motivation der Mitarbeiter, unmittelbar und exakt in Form der Eingangsparameter gesteuert werden. Fast alle Experimente werden unter anderem in Sets durchgeführt. In einem Set werden alle möglichen Störvariablen konstant gehalten. Ihr Einfluss auf einen zu beobachtenden Effekt kann damit ausgeschlossen werden.

- Die Rahmenbedingungen bei den einzelnen Simulationsläufen lassen sich exakt reproduzieren. Weder das Simulationsmodell, noch die Parameter des Szenarios verändern sich innerhalb eines einzelnen Experiments. Dadurch wird eine hohe Vergleichbarkeit der Messungen erreicht. Parameter, die nicht Teil der zu untersuchenden Rahmenbedingungen sind, werden hier typischerweise zufällig gewählt.

- Die Messungen sind reine Beobachtungen, die den Simulationsverlauf nicht beeinflussen. Die Messungen haben keinen Einfluss auf den Simulationszustand. Eine Wechselwirkung zwischen Messung und zu messendem Konstrukt ist somit ausgeschlossen.

- Die Messungen sind vollständig reproduzierbar, wenn alle Rahmenbedingungen inklusive dem Seed des verwendeten Zufallsgenerators gespeichert werden. Durch die automatisierte Messung gibt es keinen menschlichen Fehlerfaktor. Die Messmethode ist für alle Simulationsdurchläufe exakt gleich.

- Durch eine vollständige Randomisierung der Parameter des Rahmenszenarios wird eine Vorselektion vermieden. Wird ein Experiment mit konstanten Rahmenparametern in Sets durchgeführt, werden immer mehrere Sets untersucht, um auszuschließen, dass eine ungünstig gewählte Stichprobe Effekte zufällig aufweist bzw. nicht aufweist. Typischerweise werden Untersuchungen in Sets durch ein vollständig randomisiertes Experiment ergänzt, um noch besser ausschließen zu können, dass Selektionseffekte das Ergebnis beeinflussen.

Externe Validität

Die *externe Validität* beschreibt die Allgemeingültigkeit der untersuchten Stichproben für die Gesamtheit, aus der diese gewählt wurden. Nur, wenn eine Übertragbarkeit der anhand der Stichprobe ermittelten Effekte auf die Gesamtheit möglich ist, sind die Ergebnisse nützlich.

Eine häufige Gefährdung der externen Validität liegt in der Wahl der Stichprobe. Werden die Probanden nicht aus der vollständigen Grundmenge, sondern aus einer ihrer Teilmengen gezogen, ist eine vollständige Übertragbarkeit nicht mehr gewährleistet. Um das zu vermeiden, wurden bei der Evaluation des Simulationsmodells die Probanden grundsätzlich zufällig aus der Grundmenge gezogen.

Die Grundmenge besteht dabei aus allen, durch die entsprechende Belegung ihrer Parameter definierten, betrachteten Szenarios. Das Intervall, welches die gültigen Werte eines Parameters angibt, wird in Anhang A.4 beschrieben. Für die meisten Eingangsparameter entspricht es dem im Modell für den jeweiligen Parameter definierten Intervall und deckt damit dessen gesamtes Belegungsspektrum ab. Einige Parameter, wie beispielsweise die Quantitätsmodifikatoren, sind nicht nach oben begrenzt. Hier

wurden sinnvolle Grenzen definiert. Die Evaluation liefert lediglich Aussagen über die so definierten Intervalle.

Für einige Experimente wurde die Grundmenge künstlich eingeschränkt, um spezielle, in den jeweiligen Hypothesen beschriebene Situationen herzustellen. Die so untersuchten Effekte sind entsprechend nur innerhalb dieser eingeschränkten Grundmenge validiert.

Die externe Validität verbessert sich, wenn die Ergebnisse eines Experiments an anderer Stelle ebenfalls gezeigt werden können. Das kann zum einen durch die Wahl einer anderen Stichprobe geschehen, wie durch die Untersuchung mehrere Sets, jedoch auch durch völlig neue Experimente, die andere Bedingungen und Einschränkungen besitzen. Im Rahmen der Evaluation des Simulationsmodells wurden eine Vielzahl unterschiedlicher Experimente durchgeführt, die sich in ihrer Gestaltung teilweise deutlich unterscheiden. Viele der Effekte können über mehrere Experimente hinweg beobachtet werden. Widersprüche zwischen den Ergebnissen sind nur in seltenen Fällen vorhanden und allesamt auf eine zu hohe Sensitivität des Rangkorrelationskoeffizienten zurückzuführen. Durch eine manuelle Begutachtung der Messwerte konnte ein solch widersprüchlicher Effekt in keinem der Fälle bestätigt werden.

3.6. Zusammenfassung der Evaluationsergebnisse

Die Evaluation des Simulationsmodells zeigt, dass die dort getroffenen Annahmen valide, und in der Implementierung korrekt umgesetzt sind.

Dazu wurden 25 Hypothesen überprüft, die auf Aussagen aus der Fachliteratur basieren. Jede Hypothese konnte mit unterschiedlichen Experimenten am Simulationsmodell gezeigt werden. Das Simulationsmodell reproduziert somit eine Vielzahl der in der Literatur beschriebenen Effekte.

Zusätzlich zu den untersuchten Aussagen wurden alle in das Szenario eingehenden Parameter auf ihre Abhängigkeit zu wichtigen Kenngrößen eines Softwareprojekts hin untersucht. Ziel dabei war es, Anomalien und Widersprüche aufzudecken, die im Rahmen der Hypothesenprüfung nicht gefunden werden können. So wurde gezeigt, dass die jeweiligen Parameter das Verhalten des Modells korrekt beeinflussen, und somit die erwartete Parametersensitivität bestätigt. Im Rahmen der dafür verwendeten Experimente wurden auch die Extremwerte der einzelnen Parameter untersucht. Für keinen der Parameter haben sich dabei unerwünschte Anomalien ergeben.

Neben der Validität des Simulationsmodells hinsichtlich der überprüften Effekte, hat sich auch die Implementierung als leistungsstark und stabil herausgestellt. Im Zuge der Evaluation wurden mehrere tausend Experimente mit über 1 Million unterschiedlicher Simulationsläufe fehlerfrei berechnet. Die Berechnungszeit liegt dabei durchschnittlich bei ca. 1 Sekunde pro Projektbeteiligten und Projektjahr auf einem handelsüblichen Arbeitsplatzcomputer, und damit deutlich unter den für Planspielen benötigten Rechenzeiten.

Sowohl die Hypothesen, als auch die einzelnen Experimente wurden vollständig nach der Konzeption und Entwicklung des Simulationsmodells definiert bzw. erstellt. Damit kann ausgeschlossen werden, dass das Modell speziell auf die Erfüllung der einzelnen Effekte hin entwickelt wurde. Handelt es sich bei den betrachteten Effekten nicht um elementare Zusammenhänge, werden diese vom Modell endogen erzeugt. Alle betrachteten Effekte aus der Literatur konnten am Simulationsmodell nachgewiesen werden. Die Einschränkungen dabei waren minimal. Das lässt die Annahme zu, dass noch weitere, bislang nicht betrachtete Effekte, die in der Literatur beschrieben sind, durch das Simulationsmodell erzeugt werden, was dessen Mächtigkeit noch besser unterstützen würde. Um diese These zu prüfen ist jedoch eine weiterführende Evaluation notwendig.

4 Anwendungen des Modells

Das Simulationsmodell besteht aus Datenmodell, Regelwerk und Ausführungsumgebung. Es stellt damit einen Simulationskern für Planspiele im Bereich von Softwaretechnik und Projektmanagement bereit. Um ein solches Planspiel zu erhalten, muss jedoch ein entsprechender Rahmen um diesen Kern herum entwickelt werden, der die Storyline des Spiels enthält, und dem Spieler eine Interaktion mit der Simulation ermöglicht. In diesem Rahmen wird das didaktische Konzept verankert, welches die mittels des Planspiels umzusetzenden Lernziele adressiert.

Dieses Kapitel zeigt Ansätze und Möglichkeiten auf, wie das Simulationsmodell eingesetzt werden kann, um konkrete Planspiele zu erstellen. Durch den generischen Ansatz des Modells, und der Möglichkeit, es direkt ohne spezielle Middleware in ein Rahmenprogramm einzubetten, stehen hier eine Vielzahl an Einsatzmöglichkeiten offen. Einige davon werden im Folgenden exemplarisch gezeigt, um die Mächtigkeit und Flexibilität des Simulationsmodells zu demonstrieren.

Dazu werden in Abschnitt 4.1 zunächst einige unterstützende Werkzeuge vorgestellt, die im Rahmen der Entwicklung des Simulationsmodells entstanden sind, und auch für die Erstellung von konkreten Planspiele mit dem Simulationsmodell hilfreich sein können.

In Abschnitt 4.2 wird eine Vorgehensweise zur Konzeption und Entwicklung von Planspielen vorgeschlagen, um einen allgemeinen Leitfaden bereitzustellen, an dem man sich bei der Erstellung neuer Planspiele orientieren kann, und um zu zeigen, wie das Simulationsmodell in den gesamten Entwicklungsprozess eines Planspiels einzuordnen ist.

Abschnitt 4.3 beschreibt ein konkretes Planspiel, das mit Hilfe des Simulationsmodells realisiert ist. Mit diesem Spiel wurde unter anderem getestet, wie praktikabel die Entwicklung eines auf dem Simulationsmodell basierenden Planspiels tatsächlich ist.

In Abschnitt 4.4 werden abschließend einige Ausschnitte aus verschiedenen Szenarien diskutiert, um zu zeigen, wie typische Situationen mit dem Simulationsmodell erstellt und simuliert werden können.

4.1. Werkzeuge

Im Rahmen dieser Arbeit ist ein Dateiformat für das Speichern von Szenarien, eine grafische Oberfläche für die Betrachtung von Szenarien und Simulationsverläufen, sowie ein Framework für die Durchführung der Experimente im Rahmen der Evaluation entstanden. Diese Werkzeuge können bei der Verwendung und Weiterentwicklung des Simulationsmodells hilfreich sein, und werden daher im Folgenden kurz vorgestellt.

4.1.1. Dateiformat für Szenarien

Um Szenarien einfach und unkompliziert abspeichern zu können, wird ein *XML-Format* bereitgestellt, das über ein entsprechendes *XML-Schema* definiert ist. Dieses entspricht dem Datenmodell des Simulationsmodells. Dadurch lässt sich nicht nur der Startzustand der Simulation, sondern auch alle weiteren Simulationszustände abspeichern.

Dazu enthält das Paket *Persistierung* (vgl. Abschnitt 2.2) sowohl ein Import-, als auch ein Exportmodul, welches die Daten aus dem Datenmodell in das XML-Format schreibt, beziehungsweise diese in das Datenmodell des Simulationsmodells überträgt.

Dies ermöglicht es, den Simulationszustand zu jedem Zeitpunkt zu sichern und anschließend wiederherzustellen. Dadurch wird es möglich, Abschnitte des Simulationsverlaufs unter den gleichen, oder veränderten Bedingungen erneut zu simulieren, und so unterschiedliche Verläufe miteinander zu vergleichen.

Das Dateiformat ist auf die im Simulationsmodell angebotenen allgemeinen Elemente eines Szenarios beschränkt. So können beispielsweise speziell für ein Szenario entwickelte besondere Einflüsse und Vorbedingungen nicht ohne eine entsprechende Erweiterung in diesem Format gespeichert werden. Gleiches gilt für besondere Ereignisse, für die eigene Routinen umgesetzt sind, sowie für spezielle Agenten, die zusätzlich in die Simulation eingebracht werden. Das XML-Schema ist jedoch so gestaltet, dass entsprechende Erweiterungen leicht umzusetzen sind, mit denen eigene Aspekte hinzugefügt werden können. Dazu muss lediglich das bestehende Schema in ein neues Schema eingebettet, und Import- und Exportmodul entsprechend erweitert werden.

4.1.2. Grafische Oberfläche für Szenarien und Simulation

Um Szenarien und deren Simulationsverläufe besser betrachten und analysieren zu können, existiert eine *grafische Oberfläche* mit der das Simulationsmodell beobachtet werden kann. Szenarien, die im XML-Format gespeichert wurden, können damit geladen und ausgeführt werden. Dabei können die Veränderungen an den Daten im Datenmodell direkt mitverfolgt werden. Diese werden dazu entsprechend aufbereitet und teilweise visualisiert.

Abbildung 4.1 zeigt die Hauptansicht des Werkzeugs. Sie ist in drei wesentliche Bereiche aufgeteilt:

- Auf der rechten Seite wird die Struktur des aktuell geladenen Szenarios dargestellt. Dessen einzelne Elemente sind dort entsprechend gruppiert und hierarchisch dargestellt.
- Nach Selektion eines Szenarioelements auf der rechten Seite werden im linken oberen Bereich dessen Details angezeigt und gegebenenfalls Interaktionsmöglichkeiten angeboten, um Eigenschaften des Elements direkt zu manipulieren.
- Im linken unteren Bereich wird der aktuelle Projektplan in Form eines Gantt-Diagramms dargestellt.

Über die Schaltflächen der Bedienleiste kann die Simulation in verschiedenen Zeitschritten, oder bis zu einem bestimmten Tag ausgeführt werden. Es besteht außerdem die Möglichkeit, das Szenario auf seinen Ausgangszustand zurückzusetzen.

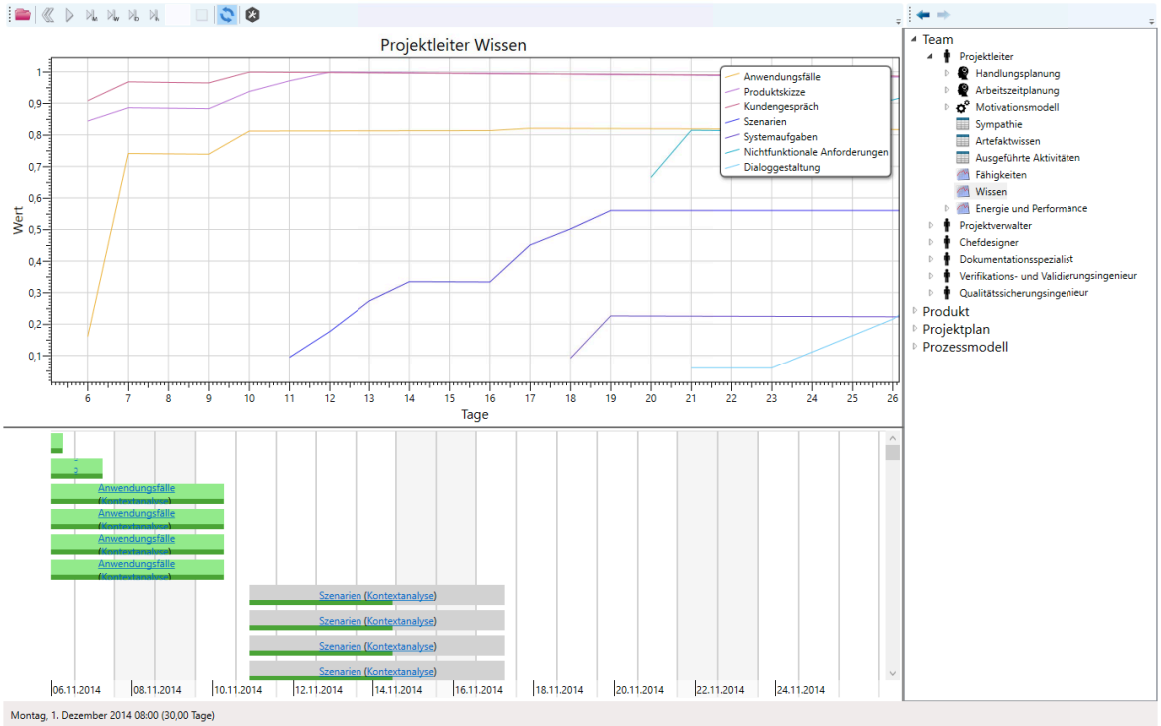
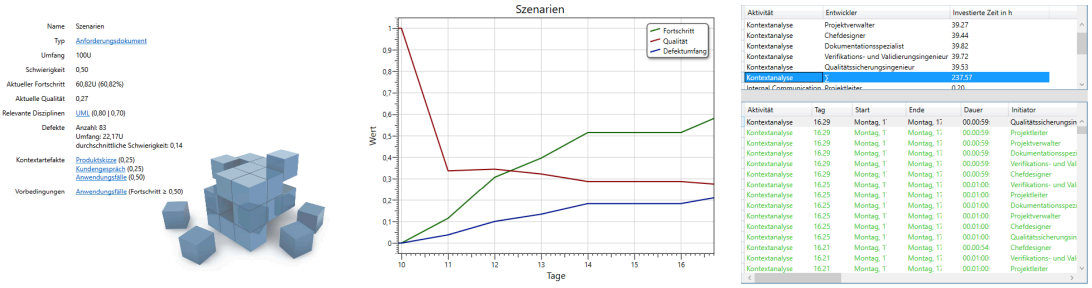


Abbildung 4.1.: Grafische Oberfläche zur Simulationsanalyse

Produkt

In der Gruppe *Produkt* sind alle Informationen über das zu erstellende Produkt abgelegt. Neben einer Übersicht, die die einzelnen Artefakte und die zwischen ihnen herrschenden Kontextbeziehungen und Vorbedingungen mittels eines Graphen darstellt, finden sich hier zu jedem Artefakt drei Ansichten, die in Abbildung 4.2 gezeigt werden.



(a) Details (b) Verlauf (c) Aktivitäten

Abbildung 4.2.: Detailinformationen zu Artefakten

Die Detailansicht zeigt die Attribute des Artefakts. Dazu gehören neben Umfang und Schwierigkeitsgrad, der aktuelle Fortschritt, Qualität, Defektanzahl und Umfang, sowie die für das Artefakt relevanten Disziplinen, dessen Kontextartefakte und Vorbedingungen.

In einer Verlaufsansicht werden die Entwicklung des Artefaktfortschritts, der Qualität und des Defektumfangs über die Zeit dargestellt. Anhand dieser Verlaufskurven lässt sich beispielsweise die Produktivität in einem bestimmten Zeitabschnitt abschätzen oder ermitteln, wie sich Ereignisse an bestimmten Zeitpunkten auf die dargestellten Werte ausgewirkt haben.

Die dritte Ansicht listet alle Tätigkeiten auf, die auf das betrachtete Artefakt angewendet wurden. Zu jeder Tätigkeit werden der konkrete Zeitraum in dem sie stattgefunden hat, die daran beteiligten Mitarbeiter, sowie die jeweils erzielte Arbeitsleistung aufgelistet. Zusätzlich wird der in die unterschiedlichen Aktivitäten investierte Zeitaufwand, gruppiert nach Mitarbeitern, aufgelistet. Anhand dieser Informationen lässt sich nachvollziehen, welche Entscheidungen die einzelnen Mitarbeiter im Projektverlauf getroffen haben, und in welchem Aufwandsverhältnis die durchgeführten Aktivitäten zueinanderstehen. Insbesondere der Anteil an Kommunikationsaktivitäten ist hierbei interessant. Die Zustandsänderung des Artefakts lässt sich anhand der in dieser Ansicht angebotenen Informationen im Detail nachvollziehen.

Prozessmodell

Die Gruppe *Prozessmodell* ist nochmals in die Teilgruppen *Artefakttypen*, *Rollen*, *Aktivitäten* und *Disziplinen* unterteilt.

Zu jedem Artefakttyp existiert eine Detailansicht die zeigt, welche Aktivitäten auf diesen Artefakttyp angewendet werden können, welche Aktivitäten Artefakte dieses Typs als Kontextartefakt akzeptieren und welche Artefakte diesen Typ einnehmen.

Zu jeder Rolle existiert eine Detailansicht die zeigt, welche Mitarbeiter diese Rolle einnehmen und für welche Aktivitäten diese Rolle benötigt wird.

Die Detailansicht der einzelnen Aktivitäten enthält deren Typ, Quantitäts- sowie Qualitätsmodifikator, die ihr jeweils zugeordneten Rollen, die Artefakttypen auf die sie angewendet werden kann und eine Liste der Artefakttypen, welche die akzeptierten Kontextartefakte beschreibt.

Zu jeder Disziplin wird angegeben, welche Mitarbeiter diese beherrschen und für welche Artefakte diese Relevant ist.

Projektplan

Der *Projektplan* wird in seinem zeitlichen Verlauf im unteren Bereich des Werkzeugs als *Gantt-Diagramm* dargestellt. Die einzelnen Aufgaben sind zusätzlich in Form einer Liste in der Baumansicht auf der rechten Seite zugreifbar. Aufgaben können im Gantt-Diagramm direkt manipuliert werden, um deren Start- und Endzeitpunkt zu verändern. Dies ist auch während einer Simulationsausführung möglich. In der Gantt-Ansicht können direkt neue Aufgaben erstellt, oder bestehende Aufgaben aus dem Projektplan entfernt werden.

Zu jeder Aufgabe existiert eine Detailansicht. In dieser wird neben Start- und Endzeitpunkt auch die Dauer in Stunden und die auf den Zeitraum entfallene tatsächliche Arbeitszeit aufgeführt. In der Detailansicht können die durchzuführende Aktivität, das zu bearbeitende Artefakt, sowie die dafür verantwortlichen Mitarbeiter für die Aufgabe ausgewählt werden.

Team

Um die Eigenschaften und Veränderungen der *Projektbeteiligten* zu beobachten, stehen für jeden Projektbeteiligten verschiedene Ansichten zur Verfügung.

In einer Übersicht werden die Basisattribute, wie die eingenommenen Rollen, das monatliche Gehalt und das Arbeitszeitkonto dargestellt. Der aktuelle Energiewert, die Motivation, alle Fähigkeiten, sowie alle Charaktereigenschaften werden als Balkendiagramme dargestellt, sodass das Profil des Mitarbeiters leicht zu erkennen ist. Zusätzlich werden einige simulationsmodellspezifische Informationen, wie die schon verbrauchte Arbeitszeit am aktuelle Tag oder die zuletzt durchgeführte Aktivität angegeben.

Zu Energie, Motivation, Fähigkeiten und Wissen existieren zusätzlich Liniendiagramme, welche die Entwicklung der einzelnen Attribute über die Zeit darstellen.

Sympathiewerte, sowohl gegenüber Aktivitäten, als auch gegenüber Kollegen, werden als tabellarische Übersicht angeboten. Ebenso tabellarisch steht eine Übersicht über das aktuelle Artefaktwissen des Mitarbeiters zu jedem einzelnen Artefakt zur Verfügung. Außerdem findet man eine Liste der vom jeweiligen Mitarbeiter durchgeführten Tätigkeiten, bezogen auf den gesamten schon berechneten Projektverlauf.

Für die Arbeitszeitplanung steht ein weiteres Liniendiagramm zur Verfügung, das die vertragliche Arbeitszeit und die geleistete Arbeitszeit pro Tag, sowie den Verlauf des Arbeitszeitkontos darstellt.

Handlungsplanung Um die VIE-Handlungsplanung nachvollziehen zu können, wird wie in Abbildung 4.3 gezeigt, eine grafisch aufbereitete Ansicht der einzelnen Berechnungsschritte angeboten.

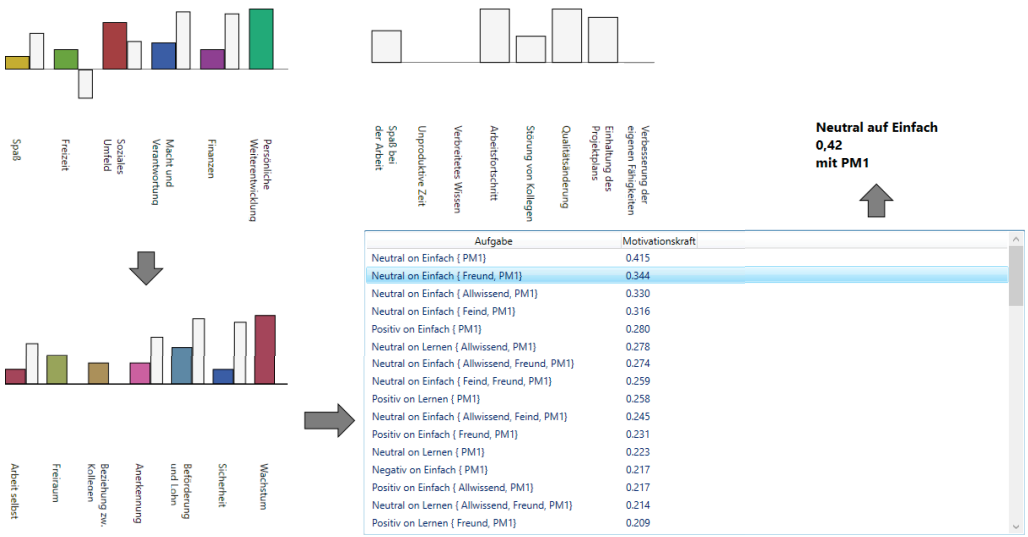


Abbildung 4.3.: Visualisierung des VIE-Entscheidungsprozesses

Links oben wird dazu die aktuelle Konfiguration des Bedürfnisvektors des betrachteten Mitarbeiters gezeigt. Darunter findet sich der daraus hervorgehende Valenzvektor der Folgen. Rechts unten findet sich eine Liste aller momentan zur Verfügung stehenden Handlungsalternativen. Deren erwartetes Ergebnis wird bei Auswahl einer Alternative direkt darüber angezeigt. Wird eine Handlungsalternative ausgewählt, werden sowohl bei den Folgen, als auch bei den Bedürfnissen, der jeweilige Erfüllungsgrad durch einen zusätzlichen grauen Balken neben dem Valenz- bzw. Bedürfniswert angezeigt.

Der Bedürfnisvektor kann direkt manipuliert werden. Die Auswirkungen auf die jeweilige Motivationskraft und die darauf basierende Rangfolge der Handlungsalternativen werden dabei sofort neu berechnet, sodass die Auswirkungen auf die Auswahl der als nächstes durchzuführenden Handlung direkt sichtbar werden.

Motivationsmodell Das *Motivationsmodell* verfügt ebenfalls über eine separate Darstellung seines Zustands. Die Ansicht, wie in Abbildung 4.4 gezeigt, beinhaltet das Zustandsmodell, in dem der aktuell eingenommene Zustand hervorgehoben ist, eine Auflistung der einzelnen Einflussfaktoren, sowie deren Anspruchsgewichte und eine grafische Darstellung des zweidimensionalen Zustandsraums, der durch Motivatoren und Hygienefaktoren aufgespannt wird. Die einzelnen Zustände aus dem Zustandsmodell sind darin mit unterschiedlichen Farben als Polygone eingezeichnet. Der aktuell vom jeweiligen Mitarbeiter eingenommene Wert im Zustandsraum wird durch einen roten Punkt markiert. Die Darstellungen wurden im Rahmen des Motivationsmodells in Abschnitt 2.4.5 genauer beschrieben.

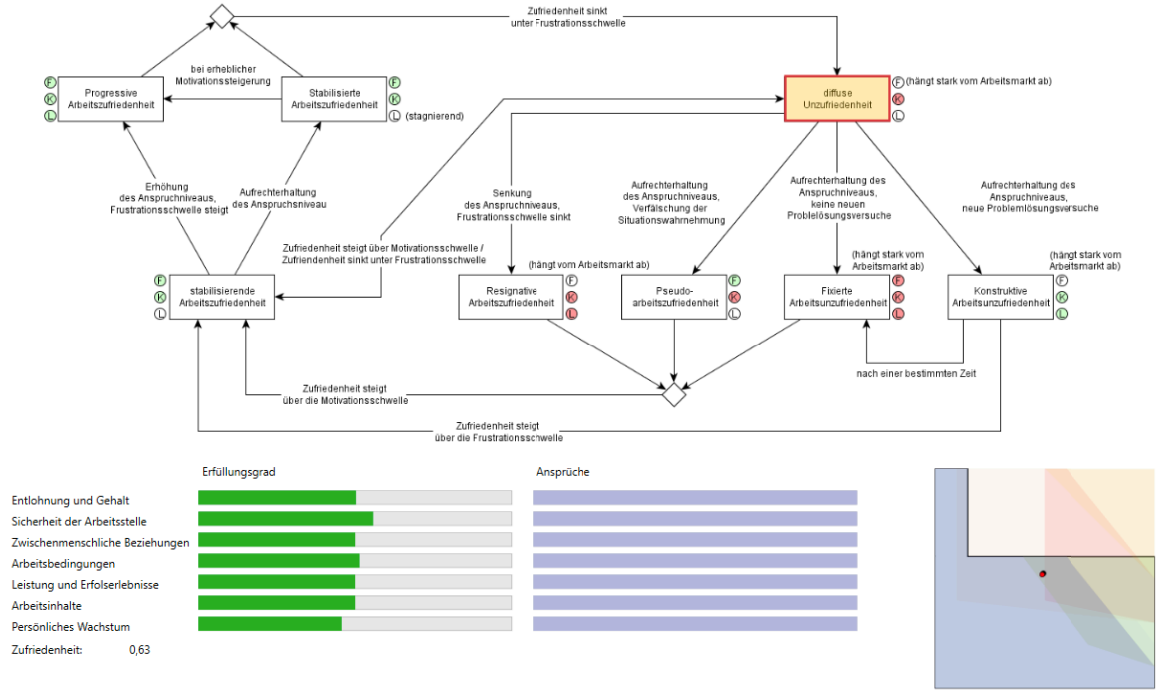


Abbildung 4.4.: Visualisierung des Motivationsmodells

4.1.3. Evaluationsframework

Um die Evaluation des Simulationsmodells effizient durchführen zu können, war es notwendig, ein kleines Framework zu schaffen, mit dem sich Experimente geschickt durchführen lassen. Es findet sich im Paket *Evaluation* (vgl. Abschnitt 2.2). Ein Experiment kann dabei als automatisierter Test gestaltet werden. Die einzelnen Hypothesen, die in Kapitel 3 beschrieben sind, wurden als solche Tests konzipiert. Werden Änderungen oder Erweiterungen am Simulationsmodell durchgeführt, können diese Tests in Form von Regressionstests verwendet werden, um sicherzustellen, dass durch die Änderungen

keine unerwünschten Nebeneffekte entstanden, und die zuvor sichergestellten Eigenschaften, wie Beispielsweise ermittelte Korrelationskoeffizienten, auch nach der Änderung noch gültig sind.

Gestaltung der Experimente

Im Rahmen der Planspielentwicklung werden typischerweise Szenarien definiert, die gewisse Eigenschaften aufweisen, oder bestimmte Effekte erzeugen sollen. Die geforderten Aspekte können mittels *Experimenten* geprüft und sichergestellt werden. Dazu muss zu jedem geforderten Aspekt ein entsprechendes Experiment erstellt werden.

Hierfür steht die abstrakte Klasse `Experiment` zur Verfügung. Das eigene Experiment wird von dieser Klasse abgeleitet. Dabei werden die Methoden `CreateGame`, `EvaluateExperiment`, und `CheckEvaluationResults` überschrieben, um das eigene Experiment zu realisieren.

In `CreateGame` wird das zu testende Szenario erstellt. Dieses kann dabei entweder algorithmisch, von Hand oder mittels diverser im Framework enthaltener Factories in Teilen oder vollständig zufällig erstellt, oder aus einer Datei geladen werden. Bei oder nach der Erstellung des Szenarios können gezielt Variationen umgesetzt werden, um unterschiedliche Ausprägungen eines Szenariokonzepts zu prüfen. Die variierten Werte werden dabei als Eingangswerte an die Basisklasse zurückgegeben, welche diese entsprechend protokolliert und später so wieder zur Verfügung stellen kann. Zusätzlich können Referenzwerte abgelegt werden, die bei der Auswertung des Simulationslaufs benötigt werden, um beispielsweise Differenzwerte zu berechnen. Sollen bestimmte Werte während des Simulationsverlaufs aufgezeichnet werden, müssen entsprechende *Monitorinstanzen* erstellt, und der Simulation hinzugefügt werden (vgl. Abschnitt 2.2). Auch zusätzliche Agenten, die den Projektverlauf beeinflussen, können an dieser Stelle der Simulation hinzugefügt werden.

In der `EvaluateExperiment`-Methode werden der Simulationsverlauf und das Simulationsergebnis nach Beendigung der Simulationsberechnung ausgewertet. Dazu bekommt die Methode den Zustand zu Berechnungsende und die vor oder während des Simulationslaufs erzeugten Referenzwerte übergeben. Daraus sind die zu betrachtenden Ausgabewerte zu berechnet und an die Basisklasse zurückzugegeben.

Um sowohl die Variation unterschiedlicher Szenarioparameter, als auch die Auswertung der diversen Aspekte einfacher zu gestalten, enthält das Framework für viele Eigenschaften entsprechende Routinen. Einige davon wurden in Kapitel 3 angesprochen und kamen bei der Evaluation des Simulationsmodells zum Einsatz.

In der Methode `CheckEvaluationResults` werden die Ergebnisse aller innerhalb eines Experiments durchgeführten Simulationsläufe gesammelt ausgewertet. Die konkrete Auswertung richtet sich dabei nach dem jeweils zu prüfenden Aspekt und erwarteten Ergebnis. Gefundene Auffälligkeiten werden an die Basisklasse zurückgemeldet, die diese sammelt und zu einem gemeinsamen Prüfbericht über mehrere Experimente hinweg zusammenfasst. Da für die Evaluation im Rahmen dieser Arbeit häufig die Korrelationen zwischen Eingabe- und Ausgabeparameter benötigt wurden, werden diese zuvor vom Framework berechnet und der Auswertungsmethode zur Verfügung gestellt.

Häufig ist es sinnvoll, die gesammelten Daten statistisch auszuwerten, um ein möglichst umfassendes und allgemeingültiges Bild über die ganze Klasse der betrachteten Szenarien zu bekommen. Es besteht jedoch auch die Möglichkeit die Datensätze einzeln auszuwerten, um harte Bedingungen zu prüfen, die nicht nur meistens, sondern in jedem einzelnen Simulationslauf gelten müssen. So lässt sich beispielsweise sicherstellen, dass die Gesamtarbeitszeit eines Mitarbeiters eine bestimmte Schwelle nicht überschreitet. Mittels der Monitore können auch während des Simulationslaufs nach jedem berechneten Simulationsschritt zuvor definierte Invarianten geprüft werden, um sicherzustellen, dass gewisse Aspekte auch während des Simulationsverlaufs immer gültig sind.

Ausführung der Experimente

Die so gestalteten Experimente werden instanziiert und dem Ausführungsteil des Evaluationsframeworks übergeben. Dieser führt anschließend für jedes Experiment folgende Schritte aus:

1. Über die `CreateGame`-Methode werden die geforderten Simulationsinstanzen erzeugt. Die genaue Anzahl, sowie eine etwaige Aufteilung in Sets wird dazu bei der Instanziierung der Experimentklasse angegeben.

2. Die Simulationsinstanzen werden einzeln ausgeführt. Dabei werden sowohl die Eingangs- als auch die Ausgangswerte in eine CSV-Datei geschrieben. Die einzeln erzeugten Szenarien werden in einem Archiv im oben beschriebenen XML-Format abgelegt. Für jeden Simulationslauf existiert somit eine eigene Szenariodatei, mit der jeder einzelne Simulationsverlauf im Bedarfsfall reproduzieren werden kann. Die Datei enthält dabei den Anfangszustand des Szenarios und den während der Simulation verwendeten bzw. erzeugten Projektplan.

   Das Framework ist in der Lage mehrere Simulationen parallel auszuführen, um alle verfügbaren Kerne der CPU auszunutzen, und damit den gesamten Berechnungsprozess zu beschleunigen. Wird diese Funktion verwendet, muss darauf geachtete werden, dass die einzelnen Experimente in allen Aspekten threadsicher sind.

3. Nach Abschluss jedes einzelnen Simulationslaufs wird die Methode `EvaluateExperiment` ausgeführt, um die Ausgabewerte zu berechnen.

4. Nach Abschluss aller Simulationsläufe werden die Korrelationskoeffizienten auf den in der CSV-Datei gesammelten Werten berechnet. Diese werden, zusammen mit den Daten der CSV-Datei der `CheckEvaluationResults`-Methode übergeben, die alle Simulationsläufe in ihrer Gesamtheit auswertet und prüft, ob der zu zeigende Aspekt korrekt umgesetzt ist.

Gefundenen Auffälligkeiten können häufig unter Verwendung der grafischen Oberfläche eingehender analysiert werden. Die im Archiv abgelegten Szenarios können dazu direkt in das Werkzeug geladen werden, um dort den entsprechenden Simulationslauf erneut auszuführen.

4.2. Konzeption und Entwicklung von Planspielen

In diesem Abschnitt wird ein Vorgehen beschrieben, wie mit Hilfe des Simulationsmodells ein vollständiges Planspiel entwickelt werden kann, und wie das Simulationsmodell in diesen Entwicklungsprozess einzuordnen ist. Im Rahmen dieser Arbeit hat sich ein Vorgehen als praktikabel herausgestellt, dass in die folgenden Schritte unterteilt werden kann [Nassal, 2017]:

1. Auswahl und Formulierung der Ziele.
2. Konzeption des Szenarios.
3. Entwicklung und Ausarbeitung des Szenarios.
4. Einschränkung des Simulationsmodells auf die benötigten Aspekte.
5. Erweiterung des Simulationsmodells um zusätzliche Aspekte.
6. Definition der Storyline mit zusätzlichen Ereignissen.
7. Entwicklung von Testfällen zur Validierung des Modellverhaltens bezüglich der Ziele.
8. Entwurf und Implementierung der Rahmenanwendung und Integration des Simulationsmodells.
9. Anreicherung des Spiels um zusätzliche Spielmaterialien.
10. Testlauf mit kleinen Testgruppen und gegebenenfalls Anpassung der vorherigen Schritte.

Auswahl und Formulierung der Ziele

Bevor mit der Entwicklung eines Planspiels begonnen werden kann, sollten die Ziele die mit dem Planspiel verfolgt werden, geklärt werden. Alle weiteren Schritte der Entwicklung müssen an diesen Zielen ausgerichtet werden. Typischerweise sollen mit einem Planspiel Lerninhalte transportiert werden. Bei den meisten Zielen handelt es sich dabei entsprechend um *Lernziele*. Neben den Lernzielen können jedoch auch andere Aspekte eine Rolle spielen. Bei einem Planspiel kann beispielsweise auch der Unterhaltungs- und Spaßaspekt im Vordergrund stehen. Planspiele können auch dazu eingesetzt werden, Spieler bei ihrer Entscheidungsfindung in konkreten Problemsituationen zu beobachten, um mehr über deren Verhalten zu erfahren. Je nach Ziel ist das Planspiel anders zu gestalten.

Vor allem bei der Definition von Lernzielen ist es wichtig, zunächst die Zielgruppe festzulegen. Insbesondere deren Ausgangswissen und Erfahrung mit Softwareprojekten ist wichtig, um später ein geeignetes Niveau für das Planspiel zu finden. Die Lernziele müssen an der Zielgruppe ausgerichtet werden. Anfänger, die noch nicht einmal die grundlegenden Prozesse kennen, werden mit der Lösung komplexer Situationen in einem Projekt sehr wahrscheinlich überfordert sein, während erfahrene

Projektmanager bei der Anwendung einfacher Prozessmodelle vielleicht schnell ihre Motivation verlieren werden.

Soll das Planspiel begleitend zu anderen Lehrformen, wie beispielsweise entsprechenden Vorlesungen oder Schulungen eingesetzt werden, sollten alle Teile entsprechend aufeinander abgestimmt werden. So können beispielsweise theoretisch besprochene Problemsituationen an konkreten Beispielen mit dem Planspiel praktisch erfahren werden.

Am Ende dieses Schritts sollte geklärt sein, wer die Spieler des Planspiels sind, in welchem Kontext das Spiel eingesetzt, und was mit diesem Einsatz bezweckt werden soll.

Konzeption, Entwicklung und Ausarbeitung des Szenarios

Im nächsten Schritt muss zunächst entschieden werden, mit Hilfe welches Szenarios die zuvor festgelegten Ziele erreicht werden können. Dazu müssen unterschiedliche Fragen geklärt, und grundsätzliche Entscheidungen getroffen werden:

- **Spielmodus.** Wird das Spiel als klassische Einzelspieleranwendung konzipiert, bei der jeweils ein einzelner Spieler unabhängig von anderen agiert? Oder nehmen mehrere Spieler an einem gemeinsamen großen Szenario teil, in dem beispielsweise mehrere Projekte situiert sind, die um einen gemeinsamen Arbeitsmarkt oder um ein gemeinsames Budget konkurrieren? Weiterhin muss entschieden werden, ob das Spiel durch einen Spielleiter moderiert, und gegebenenfalls gesteuert bzw. beeinflusst wird, oder ob die Spieler alle Aufgaben ausschließlich über das Spiel selbst durchführen und somit alle notwendigen Informationen durch das Spiel erhalten müssen.

- **Zeitablauf.** Es muss geklärt werden, über welchen zeitlichen Rahmen sich das Spiel erstrecken soll. Zum einen ist dabei relevant, wie viel Zeit die Spieler in das Spiel investieren müssen, und ob die investierte Zeit am Stück oder mit Unterbrechungen aufgewendet wird. Ein Planspiel kann beispielsweise begleitend zu einer Vorlesung in mehreren Abschnitten stattfinden, wobei die einzelnen Abschnitte auf den Ergebnissen des vorherigen Abschnitts aufbauen, oder voneinander unabhängig sein können. Weiterhin muss geklärt werden, über welchen Zeitraum sich das simulierte Projekt erstrecken soll und wie die Spielzeit im Verhältnis zur Realzeit skaliert ist.

- **Produkt.** In der Konzeptionsphase sollte festgelegt werden, welches Produkt der Spieler während des Planspiels entwickeln muss. Dabei geht es zunächst nicht um die konkrete Ausarbeitung der Artefakte, sondern um eine grundsätzliche Idee für ein Produkt, einen ungefähren Umfang und eine grobe Struktur und Granularität der Artefakte. Die einzelnen Entscheidungen sollten dabei vor allem die zuvor aufgestellten Ziele berücksichtigen. Ein Beispiel für ein solches Produkt findet sich in Abschnitt 4.3.

- **Prozessmodell.** Passend zum Produkt müssen grundsätzliche Überlegungen angestellt werden, welche Aktivitäten der Spieler zur Verfügung hat, um das Produkt zu entwickeln. Basis dafür kann ein bestehendes Prozessmodell sein. Es kann jedoch auch dem Spieler überlassen sein, welchen Prozess er anwendet. In diesem Fall sollten die Aktivitäten später möglichst unabhängig von einem konkreten Prozessmodell definiert werden.

- **Team.** Es muss festgelegt werden, welches Team der Spieler zur Verfügung hat. In der Konzeptionsphase ist vor allem wichtig, eine ungefähre Anzahl der Mitarbeiter zu definieren und darüber zu entscheiden, ob und wie der Spieler Einfluss auf die Auswahl seiner Mitarbeiter hat. Eine Möglichkeit dem Spieler Entscheidungsmöglichkeiten über Einstellung und Entlassung von Mitarbeitern zu ermöglichen, ist die Umsetzung eines Arbeitsmarkts. Abhängig von den mit dem Planspiel verfolgten Zielen, können auch spezielle Mitarbeiterkonstellationen entwickelt werden, die der Spieler handhaben muss.

- **Sonstiges.** Sind noch weitere Aspekte relevant, die nicht im Simulationsmodell verankert sind, sollten diese erfasst werden, um zu klären, wie sie sich in das Planspiel integrieren lassen.

Sobald ein grundsätzliches Konzept des Szenarios vorliegt, kann damit begonnen werden dieses im Detail auszuarbeiten. Dazu werden die einzelnen Elemente wie Artefakte, Aktivitäten, Mitarbeiter,

etc. im Detail definiert und deren Attribute mit geeigneten Werten belegt. Weiterhin müssen die Abhängigkeiten zwischen den Artefakten mittels Vorbedingungen, Einflüssen und Kontextbeziehungen definiert werden. Das Produkt muss über die Artefakttypen mit den Aktivitäten verknüpft werden, Mitarbeiter werden über Rollen mit den Aktivitäten verknüpft.

Mit Hilfe der grafischen Oberfläche können verschiedene Aspekte des Szenarios mit dem Simulationsmodell simuliert werden, um zu sehen, ob die einzelnen Parameter gut aufeinander abgestimmt sind, die angestrebten Aspekte grundsätzlich funktionieren, und um ein Gefühl für das Verhalten des Szenarios zu bekommen.

Einschränkung des Simulationsmodells

Je nach Definition der Ziele kann es sein, dass das Simulationsmodell Aspekte abdeckt, die für das zu entwickelnde Planspiel nicht relevant sind. Solche Teile des Simulationsmodells können gegebenenfalls deaktiviert werden. Alle Elementarmodelle können dazu ein neutrales Verhalten annehmen, sodass das entsprechende Modell keinen Einfluss auf die Simulation hat. Modelle, die ein Teilergebnis liefern, das als Faktor in die Berechnung des weiteren Simulationsverlaufs eingeht, liefern den neutralen Wert 1.0, geht der Wert als Summand ein, entspricht der neutrale Wert des Teilmodells 0.0.

Werden Teile des Simulationsmodells auf diese Weise entfernt, wird die Simulation einfacher nachzuvollziehen, was zum einen Vorteile für den Entwicklungsprozess des Planspiels hat, zum anderen werden die übrig gebliebenen Effekte in den meisten Fällen deutlicher sichtbar, da jeder weitere Einflussfaktor eine zusätzliche Unschärfe verursacht. So kann beispielsweise der Einfluss von Fähigkeiten auf die Produktivität besser beobachtet werden, wenn er nicht von anderen Effekten, wie beispielsweise der Gesundheit des Mitarbeiters verzerrt wird. Gleichzeitig wird die Simulation durch das Entfernen von Modellteilen aus dem Simulationsmodell jedoch auch unrealistischer, da Aspekte, die in der Realität vorhanden sind, nun im Simulationsmodell fehlen. Daher gilt es die Einfachheit des Modells, die Klarheit der Effekte und den Realismus der Simulation sorgfältig gegeneinander abzuwägen.

Die grafische Oberfläche unterstützt den Entwickler dabei, indem sie ihm die Möglichkeit bietet, einzelnen Teilmodelle direkt aus- und wieder einzuschalten. So können die Auswirkungen entsprechender Entscheidungen direkt ausprobiert werden. Dieser Schritt kann auch parallel und ergänzend zur Ausarbeitung des Szenarios durchgeführt werden.

Erweiterung des Simulationsmodells

Genauso wie Aspekte aus dem Simulationsmodell entfernt werden können, so können ihm auch zusätzliche Aspekte hinzugefügt werden. Für einige Szenarien kann es notwendig sein, das Simulationsmodell zu erweitern oder zu verändern. Dafür existieren mehrere Ansatzpunkte, die in Abschnitt 2.7 besprochen wurden.

Durch das Austauschen einzelner Elementarmodelle, dem Hinzufügen spezieller Agenten mit spezifischem Verhalten oder der direkten Manipulation des Simulationszustands vor oder nach Ausführung eines Simulationsschritts, können Effekte erzeugt und Aspekte umgesetzt werden, die nicht durch das Simulationsmodell abgedeckt sind. Wird das Simulationsmodell auf diese Weise verändert, muss gegebenenfalls gesondert geprüft werden, dass die gewünschten Effekte, die zuvor sichergestellt worden sind, immer noch vom Simulationsmodell korrekt erzeugt werden. Außerdem muss sichergestellt werden, dass die hinzugefügten Effekte mit der Realität übereinstimmen. Veränderungen am Simulationsmodell sollten dementsprechend möglichst sparsam eingesetzt werden. Maßgeblich dafür, was zusätzlich benötigt wird, sind die im ersten Schritt aufgestellten Ziele.

Definition der Storyline

Die Storyline definiert neben der Ausgangssituation des Spiels, alle Ereignisse während des Spielverlaufs. Eine gut ausgearbeitet Storyline erhöht nicht nur den Spielspaß, sondern ermöglicht es auch, den Spieler durch die Einführung von zusätzlichem Hintergrundwissen durch das Spiel zu begleiten. So können beispielsweise Aspekte, die im Szenario definiert, für den Spieler jedoch nicht direkt sichtbar sind, in Form von Zusatzinformationen mitgeteilt werden, die in eine Rahmengeschichte

eingebettet sind. Dadurch weiß der Spieler beispielsweise, dass und warum zwei Mitarbeiter nicht gut miteinander auskommen.

Besondere Ereignisse, die während des Spielverlaufs auftreten, machen das Spiel erst richtig interessant und können gleichzeitig wertvolle Lerneffekte auslösen. Dawson beschreibt in [Dawson, 2000] zwanzig „Dirty Tricks“ um angehende Softwareingenieure im Rahmen von studentischen Projekten mit einschneidenden Situationen und Ereignissen aus der Realität zu konfrontieren. Darunter fällt beispielsweise das nachträgliche Verändern von Anforderungen, die Simulation eines Hardwareausfalls, erzwungene Veränderungen im Team und vorgezogene Deadlines. Jedes der dort beschriebenen Ereignisse provoziert ein entsprechendes (Fehl-)Verhalten der Studierenden oder nutzt typisches Fehlverhalten aus, um den maximalen Schaden zu verursachen. Die Lernenden werden direkt mit den Auswirkungen ihrer Handlung konfrontiert, was zu einem nachhaltigen Lerneffekt führt.

Solche Ereignisse können auch während des Planspielverlaufs auftreten. Dazu müssen sie in die Storyline eingeplant werden. Ereignisse können dabei entweder zu festgelegten Zeitpunkten oder abhängig vom Verhalten des Spielers und dem Zustand seines Projekts eintreten. So können beispielsweise unmotivierte Mitarbeiter kündigen, überlastete Mitarbeiter krank werden oder schon fertiggestellte Produktteile aufgrund eines Serverausfalls verloren gehen. Ereignisse manipulieren dabei direkt den Simulationszustand zwischen zwei Zeitschritten. Sie müssen dazu explizit und zusätzlich zum Simulationsmodell implementiert werden.

Entwicklung von Testfällen zur Validierung

Das Framework enthält die in Kapitel 3 beschriebenen Experimente. Diese Experimente können verwendet werden um sicherzustellen, dass bestimmte Effekte auch nach einer Änderung oder Ergänzung des Simulationsmodells noch vorhanden sind. Werden von dem für das neue Planspiel erstellte Szenario zusätzliche Effekte gefordert oder erwartet, empfiehlt es sich entsprechende Tests in Form von Experimenten zu entwickeln, um diese Aspekte nach und während der Entwicklung des Planspiels sicherzustellen.

Es ist insbesondere sinnvoll, die unterschiedlichen Strategien, welche die Spieler am Planspiel ausprobieren können und sollen, ausreichend mit Tests abzudecken. Dabei sind vor allem auch falsche Strategien und schlechte Entscheidungen zu prüfen. Neben der Entwicklung automatisierbarer Tests ist es unter Umständen auch notwendig bestimmte Aspekte von Hand zu überprüfen.

Entwurf und Implementierung der Rahmenanwendung

Um dem Spieler die Planspielsimulation zugänglich zu machen, muss diese in eine entsprechende Rahmenanwendung eingebettet werden. Diese enthält die Benutzeroberfläche, Anzeigekonzepte, Hilfestellungen für den Spieler und alle weiteren Spielinhalte die nicht Teil des Simulationsmodells sind.

Spätestens für diesen Schritt muss entschieden werden, welche Plattform für das Planspiel verwendet wird. Beispielsweise kann das Planspiel als Einzelplatzanwendung gestaltet werden. Alternative Möglichkeiten sind eine Webanwendung, die vor allem bei Mehrspielerszenarien einige Vorteile verspricht. Es ist auch möglich dem Spieler keinen direkten Zugang zur Simulation zu geben und seine Entscheidungen mit Hilfe von Papierformularen über einen Spielleiter in das Simulationsmodell zu bringen.

In den meisten Spielen bekommt der Spieler einen direkten Zugang zur Simulation, indem er seine Entscheidungen direkt in das System eingibt und die Informationen über Projektstand und -verlauf direkt vom System erhält. Dazu muss zunächst entschieden werden, welche Informationen der Spieler benötigt, und welche er davon bekommen soll. Eine Offenlegung des vollständigen und exakten Simulationszustands wird den Spieler in den meisten Fällen überfordern und ist zudem unrealistisch. In der Realität hat ein Projektleiter über viele Dinge nur vage Informationen. Er weiß beispielsweise nicht genau, wie viele Fehler noch in der entwickelten Software sind oder wie hoch das Motivationsniveau jedes einzelnen Mitarbeiters gerade ist. Daher ist es sinnvoll entsprechende Informationen vor dem Spieler zu verbergen, zu filtern, zu verzerren oder zumindest auf eine gröbere Skala zu quantisieren. So kann dem Spieler beispielsweise mitgeteilt werden, ob es einem Mitarbeiter gut oder schlecht geht, anstelle ihm genaue Zahlen über dessen Motivation und Gesundheit zu geben.

Neben der Entscheidung welche Informationen dem Spieler zur Verfügung gestellt werden, muss auch darüber entschieden werden, ob und wie diese aufbereitet werden. Beispielsweise können Informationen in regelmäßigen Berichten an den Spieler weitergegeben werden, die bestimmte Zeiträume zusammenfassen. Es besteht auch die Möglichkeit, unterschiedliche Diagrammen zu verwenden, um beispielsweise zeitliche Verläufe darzustellen.

Neben den Informationen, die das Planspiel dem Spieler liefert, müssen auch Eingabemöglichkeiten angeboten werden, über die der Spieler die Simulation beeinflussen kann. Dazu muss entschieden werden, welche Änderungen der Spieler am Szenario vornehmen darf. Häufig gehört dazu die Gestaltung des Projektplans. Gegebenenfalls kann der Spieler auch Mitarbeiter einstellen und entlassen. Auch weitere Interaktionsmöglichkeiten können notwendig sein. So kann der Spieler beispielsweise zu Beginn des Projekts gefragt werden, nach welchem Vorgehensmodell er entwickeln möchte, worauf hin die Aktivitäten des Szenarios entsprechend angepasst werden.

Für alle benötigten Eingabeschnittstellen muss ein geeignetes Eingabekonzept erarbeitet werden. Dieses sollte sich nach Möglichkeit an der Darstellung des Projektzustands orientieren, sodass eine in sich konsistente und gut bedienbare Benutzeroberfläche entsteht.

Wir das Spiel ohne besondere Begleitung durch einen Tutor oder Spielleiter konzipiert, ist es gegebenenfalls sinnvoll, zusätzliche Erklärungen, Hinweise, theoretische Grundlagen oder andere Hilfestellungen in der Rahmenanwendung zu platzieren. Auch Erklärungen zum aktuellen Projektverlauf, vom Spielern gemachten Fehlern oder übersehenen Aspekten können den Lerneffekt steigern.

Außerdem muss der Spieler eine Möglichkeit erhalten, zeitlich durch das Projekt zu navigieren. In welcher Form dies abläuft, und welche Freiheiten der Spieler dabei hat, hängt von der Spielform und den Zielen des Spiels ab. Planspiele laufen in den wenigsten Fällen in Echtzeit ab. Typischerweise wird die simulierte Zeit gerafft, um deutlich weniger Zeit in das Spiel investieren zu müssen, als in ein reales Projekt. Dabei kann die Zeit kontinuierlich ablaufen oder der Spieler selbst entscheiden, bis zu welchem Zeitpunkt er im Projekt fortschreiten möchte. Eine interessante Möglichkeit kann es sein, dem Spieler zu erlauben auch rückwärts in der Zeit zu gehen, um in der Vergangenheit gemachte Fehler korrigieren zu können. Das widerspricht zwar der Realität auf das Äußerste, kann jedoch den Lerneffekt unter Umständen erheblich steigern. Das Simulationsmodell unterstützt dazu das Speichern und Wiederherstellen von Simulationszuständen zu beliebigen Zeitpunkten.

Anreicherung des Spiels um zusätzliche Spielmaterialien

Um den Spielspaß, und damit in der Regel auch den Lernerfolg zu steigern, kann das Planspiel multimedial ausgestaltet werden. Grafiken, Videos, Ton und Texte sind wichtig, um zum einen Informationen zu transportieren, zum anderen aber auch, um ein ästhetisch anspruchsvolles Spielerlebnis bieten zu können.

Eine erzählte Rahmenhandlung kann helfen, in und durch das Spiel zu führen, ohne das der Spieler nur mit trockener Theorie konfrontiert wird. Insbesondere Bilder helfen, sich an gelerntes zu erinnern, und ein gewisses Maß an Humor kann den Spielspaß erheblich steigern.

Testlauf und Anpassungen

Bevor das Planspiel in der Breite eingesetzt wird, sollte es mittels kleiner Gruppen getestet und evaluiert werden. Nur so kann sichergestellt werden, dass die erwarteten Ziele auch geeignet umgesetzt wurden, das Planspiel in seiner Konzeption und Realisierung den Ansprüchen der Zielgruppe genügt und von den Benutzern angenommen wird.

Insbesondere Planspiele, die in der Lehre eingesetzt werden, sollten ausgiebig evaluiert werden, um zu zeigen, dass der durch das Spiel entstandene Lerneffekt die dafür vom Spieler aufgebrachte Zeit rechtfertigt. Dabei können die gleichen Evaluationstechniken zum Einsatz kommen, wie sie auch bei anderen Erhebungen zum Lernerfolg und der Qualität der Lehre angewandt werden. Allem voran stehen hier Befragungen mittels Fragebogen und Tests, bei denen Wissen über den Lernstoff abgefragt werden.

Die Ergebnisse des Testlaufs können verwendet werden, um das Planspiel zu verfeinern, Fehler, die in den vorherigen Schritten gemacht wurden zu korrigieren und das Spiel insgesamt zu verbessern. Dazu kann es notwendig sein, einige Schritte zurückzugehen und einen Teil des Entwicklungsprozesses noch

einmal zu durchlaufen. Wie auch bei vielen Softwareprojekten ist die Entwicklung eines Planspiels häufig ein iterativer Prozess.

4.3. Fallstudie: Ein Planspiel zum Softwaregrundprojekt

Ein wesentlicher Bestandteil der Informatikausbildung an der Universität Ulm ist das Softwaregrundprojekt. Dieses zweisemestrige Praktikum steht verpflichtend im Lehrplan aller informatiknahen Bachelorstudiengänge. In den meisten davon wird das Projekt im dritten und vierten Semester begleitend zu einer Einführungsvorlesung in die Softwaretechnik absolviert. Die Studierenden entwickeln im Rahmen des Projekts ihr erstes vollständiges Softwaresystem. Angefangen bei der Anforderungsanalyse, bis hin zur Systemabnahme, sind dabei alle wesentlichen Phasen des typischen Entwicklungsprozesses zu durchlaufen.

Die meisten Studierenden haben zum Zeitpunkt des Projektbeginns noch keinerlei Erfahrung mit den notwendigen Prozessen, Aktivitäten und Aufgaben, die auf sie zukommen werden, und gehen daher meist mit falsche Vorstellungen und Erwartungen an das Projekt heran. Um diesem Problem entgegenzuwirken und den Projektteilnehmern möglichst früh einen Überblick über den kommenden Projektverlauf, sowie die wichtigsten Elemente und Zusammenhänge zu geben, wurde mit Hilfe des in dieser Arbeit geschaffenen Frameworks ein Planspiel entwickelt, mit dem das Projekt in kurzer Zeit von Anfang bis Ende durchgespielt werden kann [Nassal, 2015]. Damit konnten gleichzeitig Erfahrungen mit dem Framework gemacht, und gezeigt werden, dass sich damit neue Planspiele einfach und ohne großen Aufwand entwickeln lassen. Im Folgenden wird das Spiel und die Schritte seiner Entwicklung näher beschrieben. Eine Übersicht darüber findet sich auch in [Nassal, 2017].

Ziele und grundlegendes Konzept

Für das Planspiel wurden vor Beginn der Entwicklung folgende Ziele festgelegt:

- Der Spieler soll mit Hilfe des Spiels die wesentlichen Konzepte, Artefakte, Aktivitäten, Zusammenhänge und Abläufe des Softwaregrundprojekts kennen- und verstehen lernen. Das Softwaregrundprojekt basiert auf der *Fusion-Methode* [Coleman, 1994], die viele der zu erstellenden Dokumente und die dafür anzuwendenden Aktivitäten vorgibt.
- Der Spieler soll lernen, den Umfang der einzelnen Artefakte und die für deren Erstellung benötigte Zeit einzuschätzen.
- Der Spieler soll lernen, die Artefakte korrekt in ihren Kontext einzuordnen. Er soll in der Lage sein, eine sinnvolle Bearbeitungsreihenfolge der Artefakte anzugeben, und über die Abhängigkeiten zwischen den Artefakten Bescheid wissen. Er soll außerdem in der Lage sein, diese Erkenntnisse in einem Projektplan umzusetzen. Dabei soll auch das parallele Bearbeiten von Aufgaben berücksichtigt werden.
- Der Spieler soll lernen, die Mitarbeiter gemäß ihren Stärken und Schwächen für die passenden Aufgaben einzusetzen.
- Der Spieler soll lernen, dass eine kontinuierliche Qualitätssicherung über den gesamten Projektverlauf notwendig ist.
- Der Spieler soll lernen, die grundlegenden Funktionalitäten des Projektmanagementwerkzeugs *Microsoft Project 2013* zu verwenden, um einen Projektplan zu erstellen und zu pflegen.

Basierend auf diesen Zielen wurde ein dazu passendes Konzept erarbeitet. Das Spiel wird als unbegleitetes Einzelspielerkonzept realisiert, das begleitend zu einer Einführung in das Softwaregrundprojekt auf freiwilliger Basis angeboten wird. Der Spieler muss dementsprechend in der Lage sein, es ohne weitere Hilfestellung verwenden zu können.

Da das Thema des Projekts jedes Jahr wechselt, wird vom Spieler ein abstraktes Produkt entwickelt. Das hat den Vorteil, dass das Spiel nicht für jeden Projektdurchgang angepasst werden muss. Außerdem lernt der Spieler so die Konzepte, die er später auch auf andere Projekte übertragen kann.

Das Spiel besteht aus zwei wesentlichen Teilen. In einer Art Nachschlagewerk werden alle Elemente des Projekts detailliert beschrieben. Dazu gehören vor allem die Aktivitäten und Artefakte, die für das Projekt relevant sind. Der zweite Teil besteht aus der Vorgangsplanung, bei der der Spieler einen zum Projekt passenden Projektplan aufstellen muss.

Das Spiel wird direkt in *Microsoft Project* integriert. Es wird dazu als AddIn unter Verwendung der Office-API entwickelt. Über das AddIn werden alle notwendigen Funktionen direkt in Microsoft Project bereitgestellt. Die Installation einer weiteren Anwendung ist somit nicht notwendig. Über die API kann das Spiel auf nahezu alle Daten und Funktionen von Project zugreifen. Dieses Konzept bietet zwei große Vorteile. Zum einen verwendet der Spieler zur Planung seines Projekts ein ausgereiftes Werkzeug, das einen großen Funktionsumfang mitbringt. Einen solchen Funktionsumfang in einer eigens für das Planspiel entwickelten Rahmenanwendung bereitzustellen, erfordert einen erheblichen Aufwand. Zum anderen lernt der Spieler ein Werkzeug kennen, das er später in realen Projekten produktiv einsetzen kann.

Das Softwaregrundprojekt wird in Teams mit jeweils 6 Personen durchgeführt. Der Spieler bekommt dementsprechend ebenfalls ein Team aus 6 Projektbeteiligten zur Verfügung gestellt. Dieses kann von ihm nicht verändert werden. Wie auch im realen Softwaregrundprojekt spielen finanzielle Aspekte im Planspiel keine Rolle.

Am Ende des Spiels wird für den Spieler ein Score berechnet, der ihm aufzeigt, wie erfolgreich sein Projekt war. Dazu wird die Vollständigkeit und Qualität der erstellten Artefakte, sowie die dafür benötigte Zeit ausgewertet.

Der Spieler hat die Möglichkeit, sich in beliebigen Zeitschritten vorwärts durch die Simulation zu bewegen. Er kann die Simulation außerdem auf den Anfang zurücksetzen, und erneut bis zu einem bestimmten Zeitpunkt ausführen. Dadurch erhält er die Möglichkeit, Schlüsse aus dem Simulationsverlauf zu ziehen und seinen Projektplan im Gesamten zu optimieren.

Szenario

Das Szenario orientiert sich stark an den Vorgaben des realen Projekts. Wie auch im realen Projekt beträgt die Laufzeit des Projekts neun Monate, von Anfang November bis Ende Juli. An Weihnachten sind zwei Wochen Urlaubszeit eingeplant. Folgende Aktivitäten stehen dem Spieler zur Verfügung:

- Kontextanalyse
- Entwurf
- Implementierung
- Dokumentation
- Vorbereitung Qualitätssicherung
- Inspektion
- Testen
- Projektplanung

Das Produkt besteht aus 64 einzelnen Artefakten, die jeweils einem der folgenden 8 Artefakttypen zugeordnet sind:

- **Vorgabedokument** für Dokumente, welche dem Team vor Beginn des Projekts ausgehändigt werden. Dazu gehört vor allem die Produktskizze, welche als Grundlage für die Anforderungsanalyse dient.
- **Anforderungsdokument** für alle Artefakte, die im Rahmen der Anforderungsanalyse zu erstellen sind.
- **Benutzerschnittstellenspezifikation** für alle Artefakte, die Teile der Benutzerschnittstelle und Benutzerinteraktion mit dem System beschreiben. Dies sind spezielle Entwurfsartefakte, die schon während der Anforderungsanalysephase erstellt werden.
- **Entwurfsdokument** für alle Artefakte, die einen Teil des Systementwurfs enthalten.
- **Qualitätssicherung** für alle Artefakte, die Spezifikationen für die Durchführung von Test- und Inspektionsaktivitäten enthalten.

- **Dokumentation** für alle zusätzlich zu erstellenden Dokumente. Dazu gehören unter anderem das Benutzerhandbuch und die Installationsanleitung.

- **Controlling** für Dokumente, die im Rahmen des Projekts zusätzlich erstellt werden müssen. Dazu gehört vor allem der Arbeitszeitnachweis, den die Studierenden während des Projekts führen müssen.

Für die einzelnen Artefakte wurden sowohl Kontextbeziehungen als auch Vorbedingungen angelegt. Grundlage dafür sind die in der Fusion-Methode angegebenen Zusammenhänge zwischen den einzelnen Artefakten. Basiert ein Artefakt auf einem anderen, so wird dies über eine Vorbedingung mit diesem verknüpft. Kontextbeziehungen werden für alle Artefaktpaare angegeben, die Informationen aus einem anderen Artefakt verwenden oder verfeinern. Abbildung 4.5 zeigt den Aufbau des Pflichtenhefts, das während der Phase der Anforderungsanalyse entsteht. Die Artefakte *Produktskizze* und *Kundengespräch* sind Vorgabedokumente auf deren Basis die Anforderungsanalyse aufgebaut wird. Am Ende der Analysephase steht mit dem Artefakt *Pflichtenheft* ein Gesamtdokument, das alle in der Phase entwickelten Teile vereint.

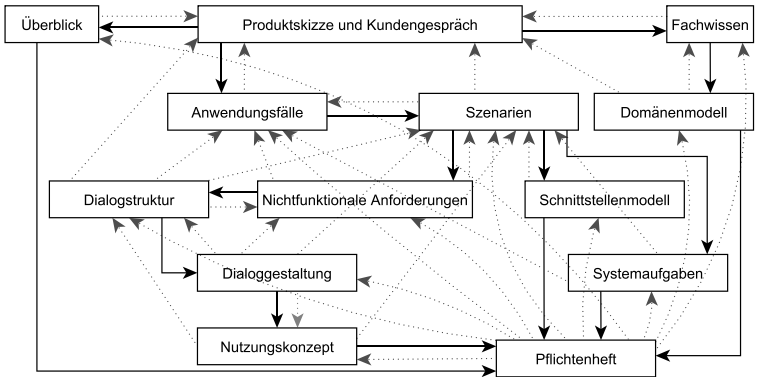


Abbildung 4.5.: Aufbau des Pflichtenhefts (Erstveröffentlichung in [Nassal, 2017])

Der Umfang der Artefakte wurde passend zum im realen Projekt für die einzelnen Teile veranschlagten Aufwand gewählt. Der Einfachheit halber geht das Planspiel von Vollzeitmitarbeitern aus, während die Studierenden im realen Projekt nur einen Teil ihrer Zeit auf das Projekt verwenden. Die Umfänge der einzelnen Artefakte wurden entsprechend auf ein Vollzeitprojekt umgerechnet, um diese Diskrepanz auszugleichen.

Einigen der Artefakte wurden zusätzlich eine oder mehrere Disziplinen hinzugefügt, um zu verdeutlichen, wo deren Schwerpunkte liegen. Dazu wurden die Disziplinen

- UML,
- Gestaltung,
- relationale Datenbanken,
- und objektorientierte Programmierung

angelegt. Beispielsweise bestehen viele der Artefakte in der Analysephase aus UML-Diagrammen. Entsprechenden Artefakten wurde deshalb diese Disziplin zugewiesen.

Während des Projekts sollen sich die sechs Teammitglieder jeweils auf unterschiedliche Schwerpunkte der Softwareentwicklung spezialisieren und sich entsprechendes Expertenwissen aneignen, um für die jeweiligen Aufgaben eine Führungsfunktion einzunehmen. Entsprechend werden auch die Fähigkeiten der Mitarbeiter im Planspiel angelegt. Abbildung 4.6 zeigt exemplarisch die Verteilung der Fähigkeiten des Architekten. Die ersten vier Balken beschreiben die Fähigkeiten bezüglich der Disziplinen, die weiteren acht Balken die Fähigkeiten bezüglich der Aktivitäten. Der Architekt bringt beispielsweise die für seinen Schwerpunkt vorteilhaften Fähigkeiten *Entwurf*, *objektorientierte Programmierung* und *UML* mit, hat dafür aber Schwächen im Bereich von *Dokumentation*, *Projektmanagement* und *Qualitätssicherung*.

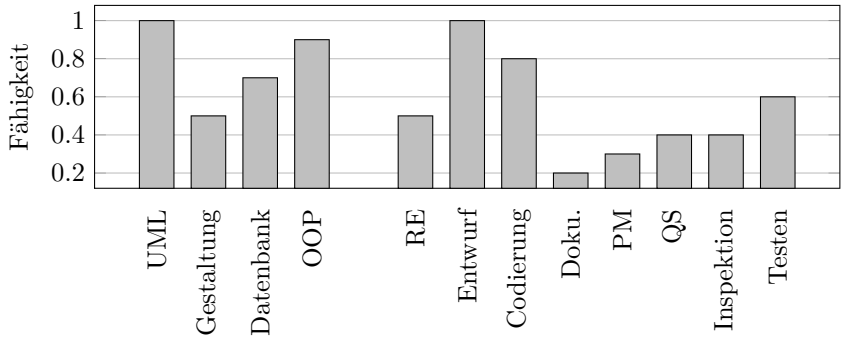


Abbildung 4.6.: Fähigkeitenprofil des Architekten (Erstveröffentlichung in [Nassal, 2017])

Um den Spieler nicht mit weiteren Besonderheiten seiner Mitarbeiter abzulenken, wurden für alle weiteren Parameter der Mitarbeiter deren Standardwerte belassen. Sie unterscheiden sich daher nur hinsichtlich ihrer Fähigkeiten.

Da das Simulationsmodell alle Aspekte, die für das Planspiel benötigt werden abdeckt, ist keine Umsetzung zusätzlicher Aspekte notwendig. Es wurden außerdem keine Teile des Simulationsmodells deaktiviert.

Das Spiel verfügt über keine weitergehende Storyline. Ausgangssituation und Ablauf sind durch das reale Projekt vorgegeben und werden in der dazugehörigen Begleitveranstaltung außerhalb des Spiel ausreichend eingeführt. Da die Zielgruppe aus sehr unerfahrenen Spielern besteht, die teilweise noch nie Kontakt mit Projektarbeit in diesem Umfang im Allgemeinen, und der Entwicklung von Softwaresystemen im Speziellen hatten, wurde auf besondere Ereignisse und schwierige Situationen im Projektverlauf vollständig verzichtet. Der Spieler kann sich somit in Ruhe mit den Grundlagen vertraut machen, ohne mit besonders schwierigen Problemen konfrontiert zu werden.

Validierung

Da das Simulationsmodell weder eingeschränkt noch erweitert wird, gelten die in Kapitel 3 gemachten Aussagen. Das Szenario wurde mit weiteren Experimenten geprüft, um vor allem unterschiedliches Spielerverhalten zu betrachten. Da die Parameter des Szenarios vom Spieler nicht verändert werden können, bleiben diese auch im Rahmen der Experimente unverändert. Der einzige Einfluss, den der Spieler auf die Simulation ausüben kann, besteht in der Gestaltung des Projektplans. Die Experimente beschränken sich daher auf diesen Teil.

Um den Spieler zu ersetzen, wird ein zusätzlicher Agent verwendet, der die Gestaltung des Projektplans übernimmt. Er setzt dabei die unterschiedlichen Strategien um, die geprüft werden sollen. Bezogen auf die Lernziele ergeben sich folgende Strategien in unterschiedlichen Kombinationen [Nassal, 2017]:

- Die Reihenfolge der Artefakte entspricht exakt der durch die Fusion-Methode vorgegebenen Reihenfolge.
- Die Reihenfolge der Artefakte wird beliebig gewählt, es wird jedoch auf die Einhaltung der entsprechenden Vorbedingungen geachtet.
- Die Reihenfolge der Artefakte wird vollständig zufällig gewählt.
- Die Mitarbeiter werden den Aktivitäten passend zu ihren Fähigkeiten zugeordnet.
- Die Mitarbeiter werden den Aktivitäten zufällig zugeordnet.
- Die Aktivitäten werden einzeln nacheinander ausgeführt.
- Die Aktivitäten werden nach Möglichkeit parallel ausgeführt.
- Qualitätssicherungsaktivitäten werden regelmäßig nach der Erstellung der Artefakte eingesetzt.
- Qualitätssicherungsaktivitäten werden erst am Ende des Projekts eingesetzt.
- Es werden keinerlei Qualitätssicherungsaktivitäten eingesetzt.

Die einzelnen Simulationsläufe wurden aufgezeichnet und anschließend ausgewertet. Dabei wurde insbesondere betrachtet, wie sich die unterschiedlichen Strategien auf die Qualität des Produkts, und die benötigte Entwicklungszeit auswirken. Im realen Projekt müssen zu gewissen Zeitpunkten

bestimmte Ergebnisse vorliegen. Diese Meilensteine können auch in der Simulation ausgewertet werden, um festzustellen, wie lange vor oder nach einem Meilenstein die dazugehörigen Artefakte fertiggestellt wurden.

Anhand dieser Experimente wurden die einzelnen Parameter des Szenarios nachjustiert, um für die erwünschten Strategien bessere, und für die falschen Strategien schlechtere Projektergebnisse zu erzielen. Dadurch werden die Lerneffekte verstärkt, da leichter erkannt werden kann, was gut und was schlecht funktioniert. Die besten Ergebnisse werden bei einer möglichst parallelen Bearbeitung der Artefakte durch die passenden Mitarbeiter mit kontinuierlicher Qualitätssicherung erreicht.

Rahmenanwendung

Um dem Spieler die Simulation zugänglich zu machen, wird diese in das Projektmanagementwerkzeug Microsoft Project eingebettet. Project wird dabei um die für das Spiel und die Steuerung der Simulation notwendigen Aspekte und Funktionen erweitert. Alle zusätzlichen Funktionen, die der Spieler zur Bedienung des Spiels benötigt, sind in einer gemeinsamen Menüleiste, wie in Abbildung 4.7 zu sehen, zusammengefasst. So kann der Spieler das Szenario in Project laden, und die Simulation in verschiedenen Zeitschritten ausführen. Bei der Simulationsausführung zeigt der Cursor, der in Project normalerweise das aktuelle Datum in der Vorgangsansicht angibt, den Zeitpunkt an, an dem die Simulation steht.

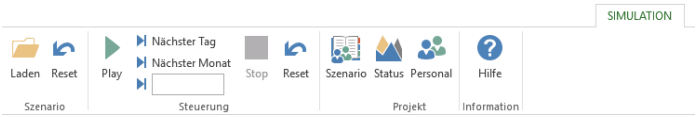


Abbildung 4.7.: Steuerung der Simulation (Erstveröffentlichung in [Nassal, 2015])

Neben der Hilfe, die alle Informationen zur Bedienung des Spiels enthält, findet der Spieler drei weitere Dialoge, die ihn mit Informationen zu seinem Projekt versorgen. Unter Szenario findet er die Informationen zu allen Szenarioelementen seines aktuell geladenen Projekts. Wie in Abbildung 4.8 ausschnittsweise gezeigt, werden die einzelnen Elemente ausführlich beschrieben und der Zusammenhang mit anderen Szenarioelementen aufgezeigt. Der Spieler kann sich so mit den Elementen des Softwaregrundprojekts vertraut machen. Unter Status wird ihm zu jedem Artefakt dessen aktuelle Vollständigkeit und Qualität angezeigt. Mittels des Dialogs zum Personal kann der Spieler sich über die Fähigkeiten seiner Mitarbeiter informieren.

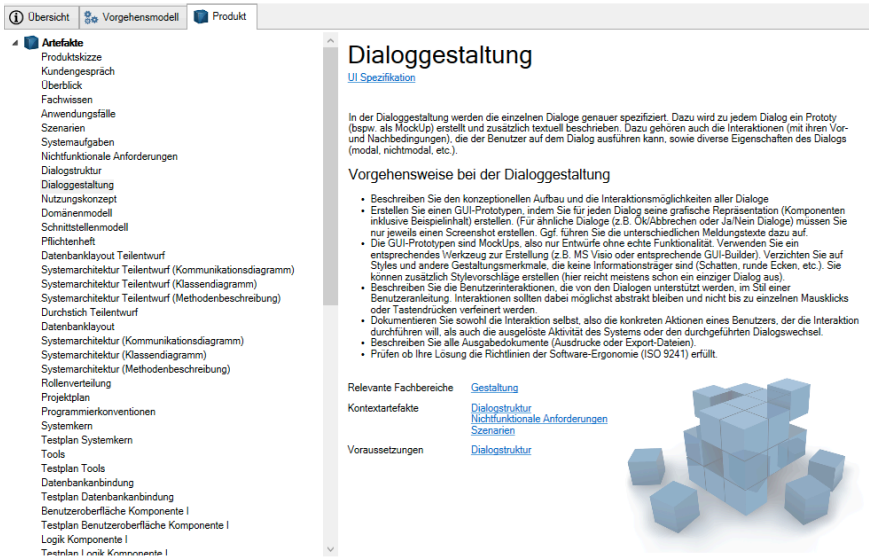


Abbildung 4.8.: Informationen zu einem Artefakt (Erstveröffentlichung in [Nassal, 2015])

Den Projektplan kann der Spieler direkt in Project anlegen und pflegen. Project stellt dazu eine Vorgangsplanung bereit, bei der zu jedem Vorgang ein Start- und Endzeitpunkt bzw. die Dauer des Vorgangs angegeben werden kann. Die ausführenden Mitarbeiter werden als Ressourcen zum jeweiligen Vorgang angegeben. Dazu werden beim Laden des Szenarios entsprechende Ressourcen in Project angelegt. Um zu jedem Vorgang das dazugehörige Artefakt und die durchzuführende Tätigkeit anzugeben, wird die Vorgangsplanung um zwei zusätzliche Spalten erweitert, in denen der Spieler Aktivität und Artefakt aus einer Liste auswählen kann. Abbildung 4.9 zeigt, wie die Vorgangsplanung in Project aussieht.

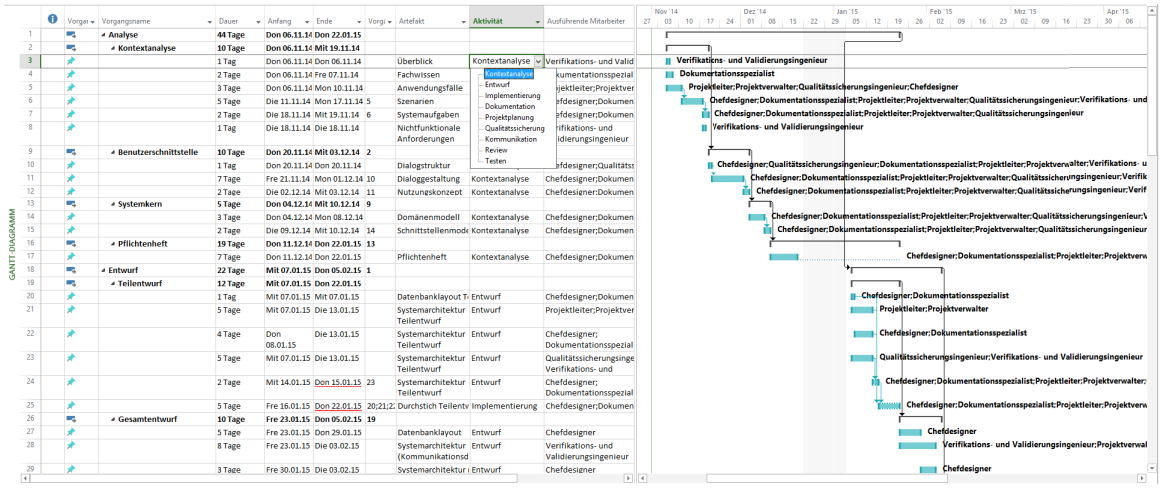


Abbildung 4.9.: Vorgangsplanung in Microsoft Project (Erstveröffentlichung in [Nassal, 2015])

Der Spieler kann bei der Vorgangsplanung auf alle Funktionen von Project zugreifen. Insbesondere kann er Abhängigkeiten zwischen den Vorgängen festlegen, diese gruppieren und einfärben, und sich die Ressourcenbelegung anzeigen lassen.

Um einen Simulationsschritt auszuführen, wird aus den vom Spieler geplanten Vorgängen ein Projektplan im Simulationsmodell erzeugt. Dieser kann anschließend ausgeführt werden, um die Änderungen am Simulationszustand für den gewünschten Zeitschritt zu berechnen. Die Anzeigen in Project werden nach Abschluss der Berechnung mit den Daten des neuen Zustands belegt, um dem Spieler die Änderungen an seinem Projekt mitzuteilen.

Der Spieler kann sich jederzeit seinen aktuellen Punktestand anzeigen lassen. Es ist ihm selbst überlassen, ob er die vollständige ihm zur Verfügung stehende Projektzeit ausnutzt, oder sein Projekt früher beendet. Er hat auch die Möglichkeit, das vorgegebene Projektende zu überschreiten. Dies wirkt sich allerdings negativ auf die Punktewertung aus. Die Punkte werden wie in Formel 4.1 [Nassal, 2015] beschrieben berechnet. Eine Wertung von 0.0 entspricht dabei dem schlechtesten Ergebnis das erzielt werden kann, 1.0 entspricht einem perfekten Projektverlauf.

$$
\begin{aligned}
\text{Produkt} &= \frac{\sum_{\alpha \in A} \alpha_c \cdot \alpha_q}{\sum_{\alpha \in A} \alpha_s} \\
\text{Zeitaufwand} &= \frac{t_{\text{jetzt}} - t_{\text{Projektbeginn}}}{t_{\text{Projektende}} - t_{\text{Projektbeginn}}} \\
\text{Score} &= \frac{\text{Produkt}}{\max(1, \text{Zeitaufwand})}
\end{aligned}
\tag{4.1}
$$

A : Menge der zu entwickelnden Artefakte

α_c : fertiggestellter Umfang des Artefakts α

α_q : Qualität des Artefakts α

α_s : gesamter Umfang des Artefakts α

Testlauf und Evaluation

Um die angestrebten Lerneffekte, die Akzeptanz des Spiels, sowie dessen Bedienbarkeit zu evaluieren, wurde eine Vorversion des Spiels einer kleinen Gruppe von 14 Studierenden zugänglich gemacht. Bei dieser Untersuchung ging es nicht darum, einen wissenschaftlichen Beleg für die Nützlichkeit des Spiels und das Erreichen der Lernziele zu erhalten. Vielmehr sollte das Feedback der Probanden Hinweise zu liefern, an welchen Stellen noch Probleme bestehen, und wo das Spiel noch verbessert werden kann. Die Evaluation wurde deswegen nicht als wissenschaftliche Studie konzipiert, sondern beschränkt sich auf eine einfache Umfrage.

Mit Hilfe eines Onlinefragebogens wurden die Spielteilnehmer zu verschiedenen Aspekten des Spielerlebnisses befragt. Da die quantitativen Ergebnisse bei einer so kleinen Stichprobe nicht ausreichend belastbar sind, wurde die Befragung durch einen qualitativen Teil in Form von persönlichem Feedback ergänzt. Die Ergebnisse der Evaluation sind auch in [Nassal, 2015] nachzulesen.

Der verwendete Fragebogen enthält 30 Fragen, die sich den folgenden Kategorien zuordnen lassen:

- **Erfassung der Rahmensituation.** Es werden Fragen zum allgemeinen Spielverhalten gestellt, die vor allem klären sollen, welche Spiele der Proband spielt, wie häufig er dies tut, und wie viel Freizeit er in Spiele allgemein investiert. Außerdem werden hier Fragen gestellt, die aufzeigen, wie der Proband mit komplexen Problemen umgeht, und wie ausdauernd er bei der Lösung schwieriger Aufgaben ist.

- **Spielerlebnis.** Es werden allgemeine Fragen zum Planspiel gestellt die aufzeigen, wie viel Zeit der Proband in das Spiel investiert hat, wie viel Spaß er dabei hatte, und ob er das Spiel öfters, gegebenenfalls mit einem anderen Szenario, spielen, bzw. es seinen Freunden weiterempfehlen würde.

- **Lernerfolg bezüglich Microsoft Project.** Um den Lernfortschritt im Umgang mit Microsoft Project zu evaluieren, wurden vor allem Fragen zum Kenntnisstand bezüglich Project vor und nach Benutzung des Spiels erfragt und um eine Selbsteinschätzung des Spielers bezüglich des Lernerfolgs gebeten.

- **Lernerfolg bezüglich des Prozessmodells.** Analog zum letzten Punkt wurde der Kenntnisstand bezüglich des im Spiel zu verwendenden Prozessmodells vor und nach Verwendung des Planspiels erfragt und um eine Selbsteinschätzung des Lernerfolgs gebeten.

- **Lernerfolg bezüglich der Artefakte.** Um zu klären, ob die Probanden die Artefakte durch das Spielen besser kennengelernt haben, wurden Fragen zum Kenntnisstand vor und nach dem Spiel gestellt und um eine entsprechende Selbsteinschätzung gebeten.

- **Lernerfolg bezüglich des Projektmanagements.** Es wurde einige Frage gestellt, um zu erfahren, welche Erfahrungen die Probanden beim Spielen des Spiels gemacht haben, und ob sie sich nach dem Spiel in der Planung und Durchführung vergleichbarer Projekte sicherer fühlen.

- **Gestaltung des Spiels.** Es wurden verschiedene Fragen zur Darstellung der Informationen, zur Bedienbarkeit und zur allgemeinen Gestaltung des Spiels gestellt.

Die Befragung hat ergeben, dass der Spielspaß eher gering eingeschätzt wird. Da bei der Entwicklung des Spiels wenig Aufwand in diesen Aspekt investiert wurde – es wurden beispielsweise keine zusätzlichen Spielmaterialen, eine interessante Storyline oder ähnliches entwickelt – ist dieses Ergebnis wenig überraschend. Trotzdem würden die meisten Probanden das Spiel mit einem anderen Szenario erneut spielen, um neue Aspekte des Projektmanagements zu lernen.

Einen Lerneffekt haben die Probanden durchgehen und für alle drei Bereiche (Prozessmodell, Artefakte und Microsoft Project) festgestellt. Alle Spieler haben angegeben, durch das Spiel etwas gelernt zu haben, der Umfang war jedoch recht unterschiedlich. Vor allem diejenigen, welche zuvor einen eher geringen Kenntnisstand hatten, haben nach eigener Aussage vom Spiel profitiert. Alle Spieler haben angegeben, die Aktivitäten und Artefakte nach Abschluss des Spiels benennen, erklären

und in den Entwicklungsprozess einordnen zu können. Außerdem trauen sich alle Spieler nach eigener Aussage zu, nach Abschluss des Planspiels auch das reale Softwaregrundprojekt als Projektleiter zu begleiten.

Mit der Bedienung des Spiels sind alle Teilnehmer zurechtgekommen. Die im Spiel integrierte Hilfe war ausreichend, um alle Funktionen zu finden und korrekt zu bedienen. Eine weitergehende Hilfestellung hat keiner der Spieler benötigt.

Für einige Darstellungen gab es Anregungen, wie diese weiter verbessert werden können. Insbesondere bei den Zusammenhängen zwischen Aktivitäten und Artefakten wurde die Übersichtlichkeit bemängelt. Bei der Bedienbarkeit von Microsoft Project hingegen gingen die Meinung von „problemlos“ bis hin zu „nur schlecht bedienbar“ weit auseinander.

Das Ergebnis des ersten Testlaufs hat gezeigt, dass das Konzept an sich funktioniert. An einigen Stellen sind kleiner Anpassungen und Optimierungen sinnvoll. Insbesondere die mediale Gestaltung des Spiels sollte nochmals überarbeitet werden, um den Spielspaß zu erhöhen. Dazu müssen geeignete Konzepte ausgearbeitet werden, die zum einen ein weiterführendes didaktisches Konzept enthalten, und zum anderen den Spielspaß fördern.

Fazit

Insgesamt hat sich bei der Entwicklung des Planspiels gezeigt, dass sich das mit dieser Arbeit entwickelte Framework gut einsetzen lässt.

Die Entwicklung des Planspiels hat in etwa einen Personenmonat in Anspruch genommen. Dabei entfällt eine Woche auf die Gestaltung des Szenarios. Eine weitere Woche wurde benötigt, um die Beschreibungstexte und weiteren Materialien für das Spiel zu erstellen. Die zwei verbleibenden Wochen entfallen auf die Integration des Spiels in Microsoft Project, wobei die Einbindung des Simulationsmodells nur wenige Stunden in Anspruch genommen hat. Der Großteil der Zeit wurde für auf die Gestaltung der Dialoge und der Benutzerschnittstellen, sowie ein funktionierendes Konzept des Speicherns und Ladens von Spielständen benötigt. Dazu wurde das vom Framework vorgegebene XML-Format um die spielspezifischen Aspekte erweitert und in das von Project vorgegebene Dateiformat für Projektpläne integriert. Die Spieler haben so die Möglichkeit die regulären Speicher- und Ladefunktionen von Project zu verwenden, um einen aktuellen Spielstand zu sichern und wiederherzustellen. Die Evaluation hat eine zusätzliche Woche in Anspruch genommen.

Anpassungen am Simulationsmodell waren nicht notwendig. Alle gewünschten Aspekte haben sich mit dem Framework direkt umsetzen lassen. Die grafische Oberfläche war bei der Entwicklung des Szenarios und der Abstimmung der einzelnen Parameter aufeinander sehr hilfreich, da das Verhalten des Modells nach Parameteranpassungen direkt beobachtet werden konnte.

4.4. Weitere Anwendungsbeispiele

In diesem Abschnitt wird anhand konkreter Beispiele gezeigt, wie bestimmte Situationen in einem Szenario des Simulationsmodells umgesetzt werden können. Dazu werden die betrachteten Aspekte als Ausschnitt eines größeren Szenarios in Form eines Minimalbeispiels dargestellt. Weitere Ansätze, bestimmte Sachverhalte zu modellieren finden sich in Anhang A.5.

4.4.1. Schnittstellenentwurf und Integration

Der Entwurf einer Systemarchitektur mit wohldefinierten Schnittstellen ist die Grundvoraussetzung einer erfolgreichen Integration [Sommerville, 2007] (vlg. auch Anhang A.5, Punkt 1). Um den Studierenden die Relevanz einer guten Schnittstellenspezifikation zu verdeutlichen, und die negativen Konsequenzen aufzuzeigen, wenn eine Schnittstellenspezifikation schlecht oder gar nicht vorgenommen werden, lässt sich ein Szenario, wie in [Nassal, 2017] beschriebene, umsetzen.

Modellierung Um den betrachteten Aspekt zu realisieren, wird der dafür relevante Teil eines größeren Szenarios betrachtet. Dieses Beispiel beschränkt sich dabei auf eine einzelne Komponente des zu entwickelnden Produkts, das Muster kann jedoch beliebig repliziert werden, um komplexere Produkte zu erstellen. Es werden die vier in Abbildung 4.10 gezeigten Artefakte betrachtet.

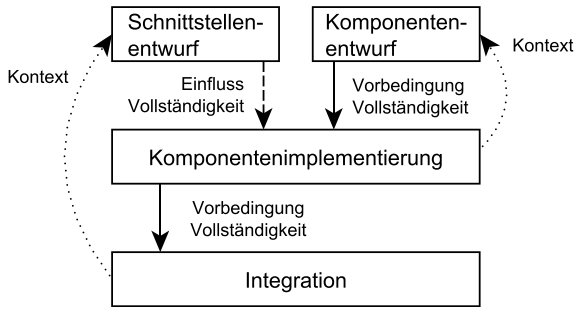


Abbildung 4.10.: Artefakte des Integrationsszenarios und deren Beziehungen (Erstveröffentlichung in [Nassal, 2017])

Für den Schnittstellenaspekt sind die Schritte Entwurf, Implementierung und Integration relevant, zu denen jeweils ein entsprechendes Artefakt existiert, das mit einer geeigneten, hier nicht näher definierten Aktivität erstellt werden kann. Der Entwurf wird in Schnittstellenentwurf und Komponentenentwurf unterteilt, um die Schnittstellenspezifikation getrennt betrachten zu können. Das Artefakt *Komponentenentwurf* repräsentiert alle Entwurfsaspekte, ausgenommen derer, welche die Schnittstelle betreffen. Die *Komponentenimplementierung* stellt den vollständigen Code der betrachteten Komponente dar. Im Rahmen des hier betrachteten Aspekts ist eine detailliertere Aufteilung nicht notwendig. Das Artefakt *Integration* enthält den Code der diese Komponente in das Gesamtsystem integriert.

Um die Artefakte miteinander zu verbinden, werden unterschiedliche Beziehungen verwendet. Um die Reihenfolge von Entwurf, Implementierung und Integration festzulegen, werden entsprechende Vorbedingungen eingeführt, die eine Bearbeitung eines Artefakts erst zulassen, wenn das vorangegangene Artefakt abgeschlossen wurde. Damit wird beispielsweise verhindert, dass eine Komponente integriert wird, die noch gar nicht existiert.

Da untersucht werden soll, was passiert, wenn kein, oder nur ein unvollständiger Schnittstellenentwurf gemacht wird, darf dieser nicht als Vorbedingung der Komponentenimplementierung modelliert werden. Es muss jedoch realisiert werden, dass eine Änderung des Schnittstellenentwurfs einen Einfluss auf die Komponentenimplementierung hat, sodass das Verändern der Schnittstellenspezifikation eine Überarbeitung der Komponentenimplementierung notwendig macht. Dazu wird ein besonderer Einfluss des Schnittstellenentwurfs auf die Komponentenimplementierung definiert. Ausgehend von der Annahme, dass 20% der Komponentenimplementierung vom Schnittstellenentwurf abhängig sind, wird bei einer Änderung am Schnittstellenentwurf der Fortschritt der Komponentenimplementierung anteilig reduziert. Werden beispielsweise 50% des Schnittstellenentwurfs erstellt, verliert die Komponentenimplementierung 10% ihres Fortschritts. Da die Integration über eine Vorbedingung von der Komponentenimplementierung abhängt, hat dies gegebenenfalls auch auf diese entsprechende Auswirkungen. Da besondere Einflüsse frei definiert werden können, ist dies nur eine von vielen Möglichkeiten den Einfluss des Schnittstellenentwurfs auf die Komponentenimplementierung zu realisieren.

Um den Einfluss des Schnittstellenentwurfs auf die Integration zu realisieren, wird eine entsprechende Kontextbeziehung zwischen den beiden Artefakten definiert. Eine weitere Kontextbeziehung wird zwischen Komponentenentwurf und Komponentenimplementierung angelegt.

Erzeugtes Verhalten Das von diesem Szenario erzeugte Verhalten lässt sich an drei unterschiedlichen Strategien und deren Ergebnissen zeigen. Der Spieler kann beispielsweise folgende Ansätze ausprobieren:

- **Lehrbuchvorgehen.** Der Spieler verhält sich, wie es im Lehrbuch als optimale Vorgehensweise beschrieben wird. Er erstellt zuerst Schnittstellenentwurf und Komponentenentwurf, und führt eine geeignete Qualitätssicherung der Entwürfe durch, bevor er mit der Implementierung beginnt. Nach Abschluss der Komponentenimplementierung und deren Qualitätssicherung wird die Integration durchgeführt.

- **Keine Qualitätssicherung.** Der Spieler verhält sich wie im Lehrbuch vorgegeben, verzichtet jedoch auf eine Qualitätssicherung des Entwurfs.

- **Kein Schnittstellenentwurf.** Der Spieler verzichtete vollständig auf die Erstellung des Schnittstellenentwurfs. Alle anderen Komponenten werden wie beim Lehrbuchvorgehen entwickelt.

Um die Auswirkungen der unterschiedlichen Strategien zu beobachten, wird zu jeder Strategie ein entsprechender Projektplan erstellt, der von einem Mitarbeiter ausgeführt wird. Der so entstandene Simulationsverlauf wird ausgewertet. Dabei werden die Zeiten betrachtet, die für die einzelnen Teile und den gesamten Projektabschnitt benötigt wurde. Außerdem wird die Qualität der einzelnen Ergebnisse betrachtet. Tabelle 4.1 zeigt die so erhaltenen Simulationsergebnisse. Die Gesamtdauer umfasst zusätzlich zu den in der Tabelle angegebenen Zeiten für Schnittstellenentwurf und Integration einen Aufwand von 136h für Komponentenentwurf und -implementierung, der für allen drei Strategien gleich ist.

	Lehrbuchvorgehen	k. Qualitätssicherung	k. Schnittst.entwurf
Dauer Schnittst.entwurf	72h	32h	—
Qualität Schnittst.entwurf	0.89	0.57	—
Dauer Integration	40h	144h	256h
Qualität Integration	0.83	0.45	0.32
Gesamtdauer	248h	312h	392h

Tabelle 4.1.: Vergleich der unterschiedlichen Vorgehensweisen im Integrationsszenario [Nassal, 2017]

Anhand der durch die Simulation erhaltenen Daten lässt sich zeigen, dass das Vorgehen nach Lehrbuch die besten Ergebnisse liefert. Der gute Schnittstellenentwurf wirkt sich positiv auf die Integrationszeit aus. Die zusätzlich in den Schnittstellenentwurf investierte Zeit wird dabei mehrfach bei der Integration eingespart, sodass sich insgesamt eine positive Gesamtbilanz ergibt. Ein fehlerbehafteter und nicht vorhandener Schnittstellenentwurf hingegen spart zunächst Zeit. Bei der Integration entsteht dadurch jedoch ein Mehraufwand, der so groß ist, dass sich eine Einsparung des Schnittstellenentwurfs nicht lohnt. Ein zusätzlicher Simulationslauf hat sogar gezeigt, dass es sich selbst nach Fertigstellung der Implementierung noch lohnt, den Schnittstellenentwurf nachträglich zu erstellen, und die entsprechenden Teile der davon beeinflussten Implementierung neu zu erstellen, um eine gute Integration durchführen zu können.

Testet der Spieler die beschriebenen Strategien am hier vorgestellten Szenario, so erfährt er dadurch, dass das Vorgehen nach Lehrbuch sinnvoll ist und was passiert, wenn man es nicht einhält. Das Verhalten der Simulation ist dabei wesentlich von den genauen Parametern des Szenarios abhängig. Der hier beschriebene Effekt ist drastischer dargestellt, als er in den meisten Fällen in der Realität auftreten wird. Er lässt sich durch das Einflussgewicht der Kontextbeziehung zwischen Schnittstellenentwurf und Integration regeln. Der Mehraufwand, der durch eine nachträgliche Erstellung des Schnittstellenentwurfs entsteht, wird durch den Faktor im besonderen Einfluss geregelt. Diese beiden Parameter können je nach Lernziel entsprechend gewählt werden, um einen Effekt realistischer oder überzeichnet, und damit leichter erkennbar zu machen.

4.4.2. Referenzdokumente für Inspektionen

Die Effektivität, mit der Fehler durch eine Inspektion gefunden werden, hängt maßgeblich von der Qualifikation der Inspektoren, und der Vollständigkeit und Korrektheit der Referenzdokumente ab [Weller, 1993] (vgl. auch Anhang A.5, Punkt 10). Ein Szenario, mit dem dieser Aspekt in einem Planspiel verdeutlicht werden kann, lässt sich wie in [Nassal, 2017] beschrieben modellieren.

Modellierung Der Aspekt kann über eine Kontextbeziehung zwischen den beiden Artefakten *Prüfling* und *Referenzdokument* realisiert werden. Dadurch wird bei der Bearbeitung des Prüflings der Zustand des Referenzdokuments entsprechend berücksichtigt. Abbildung 4.11 zeigt den Aufbau des relevanten Szenarioausschnitts. Als Referenzdokument wird hier beispielsweise eine Checkliste angeboten. Der Prüfling selbst ist vom Typ *Code*. Die Inspektionsaktivität kann auf Artefakte des Typs Code angewendet werden und berücksichtigt Kontextartefakte des Typs Referenzdokument.

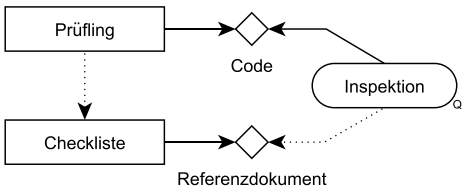


Abbildung 4.11.: Artefakte des Inspektionsszenarios

Um die Auswirkungen dieser Konstellation zu untersuchen, wird dem Szenario ein Inspektionsteam mit vier Mitgliedern hinzugefügt. Dem simulierten Prüfling werden 66 Defekte hinzugefügt, die es zu identifizieren gibt. Die Schwierigkeit der Defekte wird dabei gleichmäßig zwischen 0 und 1 verteilt.

Erzeugtes Verhalten Die Effektivität, mit der die Inspektion durchgeführt wird, lässt sich anhand der gefundenen Defekte und der dafür benötigten Zeit abschätzen. Es werden verschiedene Zustände des Referenzdokuments und unterschiedliche Fähigkeiten der Inspektoren hergestellt, und das Szenario unter diesen Bedingungen simuliert. Tabelle 4.2 enthält die so erhaltenen Messwerte. Eine Qualität des Referenzdokuments von 1.0 entspricht einem fehlerfreien Artefakt, eine Qualität von 0.7 dem durchschnittlichen Qualitätsniveau eines solchen Dokuments und eine Qualität von 0.2 einem stark fehlerbehafteten Dokument. Ein Fähigkeitsniveau von 0.5 entspricht der durchschnittlichen Fähigkeit eines Inspektors.

Checkliste		Inspektoren	gefundene Defekte	
Vollständigkeit	Qualität	Fähigkeitsniveau	pro Tag	insgesamt
100%	0.7	0.5	8.0	48
50%	0.7	0.5	3.5	42
0%	0.7	0.5	2.7	36
100%	0.2	0.5	4.2	42
100%	1.0	0.5	13.2	66
100%	0.7	1.0	16.5	66
100%	0.7	0.1	1.4	18

Tabelle 4.2.: Ergebnisse der Inspektion unter verschiedenen Bedingungen [Nassal, 2017]

Die ersten beiden Abschnitte der Tabelle zeigen, wie sich der Zustand des Referenzdokuments auf den Fehlerfindungsprozess auswirkt. Je höher die Vollständigkeit und Qualität des Dokuments ist, umso schneller werden die Defekte im Prüfling aufgedeckt. Der Zustand des Referenzdokuments hat jedoch nicht nur Einfluss auf die Geschwindigkeit, sondern auch darauf, welche Defekte überhaupt gefunden werden können. So konnten bei einer Vollständigkeit von 50% und einer Qualität von 0.7 nur 42 von 66 Defekten aufgedeckt werden. Um auch die schwierigsten Defekte zu finden, ist es notwendig, entweder mit sehr guten Referenzdokumenten zu arbeiten oder Inspektoren mit sehr

hohen Fähigkeiten einzusetzen. Insbesondere ein niedriges Fähigkeitsniveau der Inspektoren führt zu einem schlechten Inspektionsergebnis, selbst wenn die Referenzdokumente in Ordnung sind.

Ein weiterer interessanter Aspekt, der sich an diesem Szenario beobachten lässt, ist die Einarbeitungszeit, die benötigt wird, um sich mit den Referenzdokumenten und dem Prüfling vertraut zu machen. Abbildung 4.12 zeigt den zeitlichen Verlauf der simulierten Inspektionsaktivität. Die Qualität des Prüflings ist als durchgezogene Linie, und der Anteil Umfang der im Artefakt vorhandenen Defekte als gestrichelte Linie dargestellt.

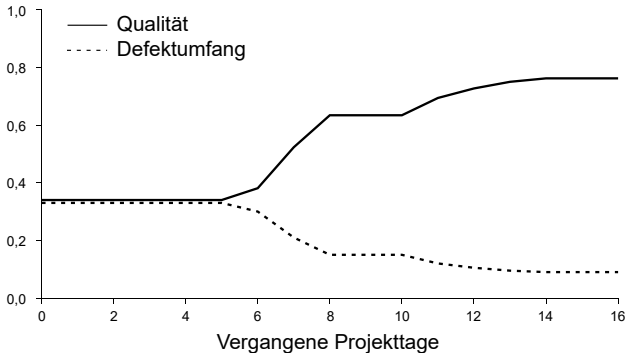


Abbildung 4.12.: Veränderung des Prüflings während eines Inspektionsvorgangs (Erstveröffentlichung in [Nassal, 2017])

Es fällt auf, dass am Anfang des Vorgangs keine Veränderung am Artefakt zu sehen ist. Auf dieses Zeitintervall entfallen 2 Wochenendtage an denen nicht gearbeitet wird, die restlichen drei Tage werden für die Einarbeitung in den umfangreichen Prüfling und die Checkliste benötigt. Im weiteren Verlauf sinkt der Umfang der unentdeckten Defekte, während die Qualität gleichermaßen steigt. Zu Beginn des Prozesses werden die Defekte schneller identifiziert da die Defektdichte höher ist und mehr einfach zu findende Defekte im Artefakt existieren. Tag 9 und 10 sind wieder Wochenendtage, entsprechend wird an diesen Tagen kein Fortschritt erzeugt.

4.4.3. Einbeziehung des Kunden bei der Inspektion nicht-technischer Dokumente

In [Pressman, 1997] findet sich die Aussage, dass die Effektivität von Inspektionen nicht-technischer Dokumente gesteigert werden kann, indem der Kunde miteinbezogen wird (vgl. auch Anhang A.5, Punkt 12).

Dieser Aspekt kann mittels zweier unterschiedlicher Ansätze umgesetzt werden. Grundlage beider Ansätze ist die Annahme, dass der Vorteil der Zusammenarbeit mit dem Kunden durch dessen Wissensvorsprung bezüglich des zu begutachtenden Dokuments gegenüber den Entwicklern zustande kommt. Da der Kunde typischerweise über umfangreicheres und detaillierteres Wissen über die Anwendungsdomäne verfügt, ist er prädestiniert entsprechende Ergebnisse zu begutachten und zu beurteilen.

Eine der Möglichkeiten dies zu realisieren ist es, das Domänenwissen über eine eigene Disziplin abzubilden. Der Wissensvorsprung des Kunden kann so direkt mittels des Fähigkeitsniveaus bezogen auf diese Disziplin umgesetzt werden, indem es für die Entwickler auf einen niedrigen, und für den Kunden auf einen hohen Wert gesetzt wird. Für die betroffenen Dokumente wird diese Disziplin verlangt. Der Kunde selbst wird als Teil des Teams modelliert, die Aktivitäten, die er durchführen kann, werden über das Rollenkonzept entsprechend eingeschränkt.

Die höhere Produktivität bei der Durchführung der Inspektion ergibt sich, wie in Hypothese 1 gezeigt, durch die höhere fachliche Fähigkeit des Kunden. Werden für das zu begutachtende Artefakt neben dem Anwendungsdomänenwissen noch weitere Fähigkeiten benötigt, beispielsweise technische Fähigkeiten, welche die Entwickler besitzen, der Kunde jedoch nicht, kann die Produktivität noch weiter gesteigert werden, indem Kunde und Entwickler gemeinsam am Artefakt arbeiten und sich in ihren Fähigkeiten so entsprechend ergänzen können.

Ein anderer Ansatz ist die Realisierung mittels eines zusätzlichen Artefakts, das nicht bearbeitet oder verändert werden kann, und welches das gesamte relevante Domänenwissen enthält. Während der Kunde über das vollständige in diesem Artefakt vorhandene Wissen verfügt, besitzen die Entwickler davon nur wenig oder gar nichts. Das Domänenwissenartefakt kann nun als Kontext für die zu begutachtenden Dokumente gesetzt werden. Über den Artefakttyp des Domänenwissenartefakts und die von den Aktivitäten jeweils akzeptierten Kontexttypen kann die Kontextbeziehung dabei bei Bedarf auf Inspektionsaktivitäten eingeschränkt werden, sodass andere Aktivitäten nicht betroffen sind. Über den Einflussfaktor der Beziehung lässt sich steuern, wie stark die Ergebnisse der Inspektion vom Wissen der Inspektoren abhängt. Wird der Kunde in die Aktivität eingebunden, so steht dem Inspektionsteam das notwendige Wissen zur Verfügung. Alternativ werden die Inspektoren sich selbstständig in das erforderliche Domänenwissen einarbeiten, was je nach dessen Umfang jedoch unter Umstände sehr viel Zeit in Anspruch nehmen kann. Der Einfluss von Wissen wird im Rahmen von Hypothese 2 untersucht. Dort wird gezeigt, dass ein höheres Wissen zu einer höheren Produktivität führt.

Der Unterschied zum vorherigen Ansatz ist, dass Domänenwissen nicht als Fähigkeit, sondern als Wissen abgebildet wird. Dieses können sich die Entwickler deutlich schneller selbstständig aneignen, als es mit Fähigkeiten der Fall ist, die deutlich mehr auf Übung und Erfahrung basieren als auf reinem Wissen. Welcher Ansatz gewählt werden sollte hängt daher maßgeblich davon ab, ob es sich bei dem benötigten Domänenwissen um reines Wissen oder eher um Erfahrung handelt. Beide Ansätze lassen sich problemlos kombinieren, falls beide Ausprägungen berücksichtigt werden sollen.

4.4.4. Testgetriebene Entwicklung

Bei der *testgetriebenen Entwicklung* werden zuerst die Testfälle entwickelt bevor der zu testende Code geschrieben wird (vgl. auch Anhang A.5, Punkt 13).

Um ein solches Vorgehen in einem Szenario umzusetzen, kann dieses wie in Abbildung 4.13 modelliert werden.

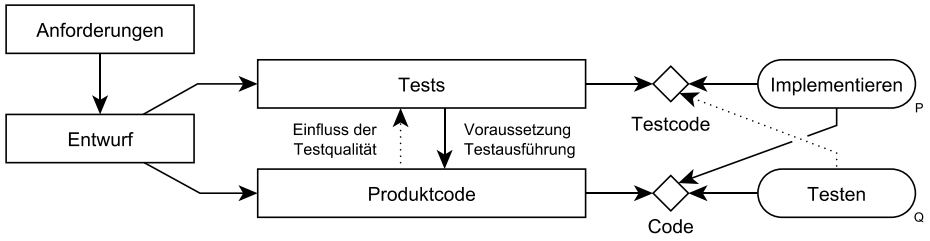


Abbildung 4.13.: Ausschnitt aus einem Szenario zur testgetriebenen Entwicklung

Zusätzlich zum Artefakt *Produktcode*, das den Quellcode des Produkts enthält, existiert ein Artefakt *Tests*, welches den Testcode enthält, um die Implementierung zu prüfen. Durch eine Vorbedingung zwischen diesen beiden Artefakten können Tests auf dem Produktcode nur durchgeführt werden, wenn diese zuvor entwickelt wurden. Die Qualität der Tests hat außerdem einen direkten Einfluss auf deren Ergebnisse, was durch eine entsprechende Kontextbeziehung realisiert wird.

Der Spieler hat in diesem Szenario die Wahl, ob er zuerst die Tests oder den Produktcode entwickelt. Wird der Produktcode vor den Tests entwickelt, so hat der Spieler keine Möglichkeit eine entwicklungsbegleitende Qualitätssicherung durchzuführen, was, wie im Rahmen von Hypothese 17 gezeigt, Nachteile mit sich bringt. Um den Spieler zu zwingen, zuerst die Tests zu entwickeln, kann eine zusätzliche Vorbedingung eingefügt werden, die eine Implementierung des Produktcodes erst erlaubt, wenn die Entwicklung der Tests abgeschlossen ist.

Eine weitere Variation des Szenarios kann erzeugt werden, indem die Vorbedingung zwischen Produktcode und Tests entfernt wird. Der in der übrig gebliebenen Kontextbeziehung hinterlegte Einfluss wird auf 1 gesetzt. Damit müssen die Tests nicht mehr vollständig entwickelt werden, um Testaktivitäten auf dem Produktcode durchzuführen. Der Zustand der Tests, sowohl Vollständigkeit als auch Qualität, haben jedoch einen direkten Einfluss auf die Ergebnisse der Qualitätssicherungsaktivität.

4.4.5. Anforderungsänderungen während der Entwicklung

In [Curtis, 1988] findet sich die Aussage, dass während des Entwicklungsprozesses häufig neue Anforderungen aufkommen, da diese erst identifiziert werden können, wenn ein gewisser Anteil des Systems schon umgesetzt wurde (vgl. auch Anhang A.5, Punkt 59). Um diesen Sachverhalt nachzubilden, gibt es verschiedene Ansätze, die hier kurz skizziert werden.

Das Produkt kann beispielsweise, wie in Abbildung 4.14 gezeigt, in einzelne Teile zerlegt werden, die über entsprechende Vorbedingungen verknüpft sind, welche die Reihenfolge der Entwicklung festlegen. Das Gesamtprodukt, das der Spieler zu erstellen hat, besteht beispielsweise aus den Artefakten *Produkt*, *Produktänderungen* und *Erweitertes Produkt*, die gegebenenfalls wiederum in einzelne kleinere Artefakte zerlegt werden können, um beispielsweise den Entwurfsaspekt zu realisieren. Um den Spieler mit unvorhergesehenen Anforderungsänderungen zu konfrontieren, werden die neuen Anforderungen und die damit verbundenen neuen Produktteile erst während des Spielverlaufs zu einem vorher definierten Zeitpunkt sichtbar gemacht.

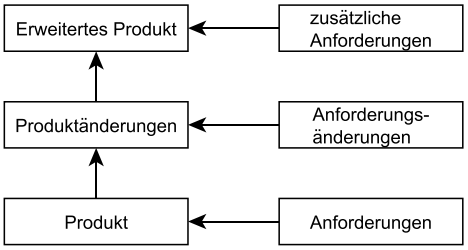


Abbildung 4.14.: Aufteilung des Produkts zur Realisierung sich ändernder Anforderungen

Sollen die Anforderungen und das Produkt jeweils als ein einziges Artefakt modelliert werden, muss ein anderer Ansatz gewählt werden. Dabei werden die beiden Artefakte über Vorbedingungen wechselseitig in Beziehung gesetzt. Hierfür wird der Fortschritt der beiden Artefakte relativ zueinander betrachtet. D. h. das Produkt kann nur soweit entwickelt werden, wie auch die Anforderungen dazu erfasst wurden, die Vollständigkeit der Anforderungserfassung kann wiederum maximal zu einem festgelegten Prozentsatz über der Vollständigkeit des Produkts liegen. Abbildung 4.15 zeigt einen solchen Aufbau.

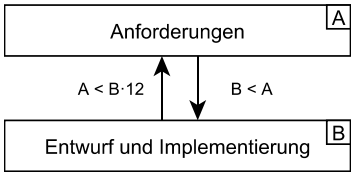


Abbildung 4.15.: Verknüpfung zweier gegenseitig voneinander abhängiger Artefakte

Artefakt A kann so weit entwickelt werden, bis seine Vollständigkeit ein zwölffaches der Vollständigkeit von Artefakt B erreicht. Danach muss zuerst Artefakt B weiterentwickelt werden. Artefakt B kann jeweils nur soweit entwickelt werden, wie Artefakt A schon fortgeschritten ist. Dadurch müssen Artefakt A und Artefakt B in einem entsprechenden Wechsel bearbeitet werden. Für einen sehr kleinen Fortschritt von Artefakt A und Artefakt B müssen entsprechende Ausnahmen vorgesehen werden, um sehr kleine Bearbeitungsintervalle bzw. Intervalle der Länge 0 zu vermeiden.

Im Gegensatz zur oben gezeigten Umsetzung, beschreibt dieser Ansatz ein kontinuierlicheres Hinzufügen neuer Anforderungen. Dem Spieler kann dabei der Gesamtumfang von Anforderungs- und Produktartefakt verborgen bleiben und nur angezeigt werden, welchen Anteil der Anforderungen er schon umgesetzt hat, und ob es momentan noch neue Anforderungen gibt, die er noch nicht entdeckt hat.

Eine weitere Möglichkeit ist es, die Artefakte und deren Beziehungen während des Projektverlaufs zu bestimmten Zeitpunkten explizit zu manipulieren, um so die sich ändernden Anforderungen zu realisieren.

4.4.6. Agiles Vorgehen mit Scrum

Viele der bislang besprochenen Szenarien sind auf lineare Prozessmodelle fokussiert. Dieser Abschnitt zeigt, dass mit dem Simulationsmodell auch agile Prozesse nachgebildet werden können. Im Folgenden wird dazu ein Szenario skizziert, das auf ein agiles Vorgehen mit *Scrum* ausgerichtet ist, und damit eine von vielen Möglichkeiten aufzeigt, wie ein entsprechendes Planspiel umgesetzt werden kann. Abbildung 4.16 zeigt die wesentlichen Aspekte der Struktur dieses Szenarios.

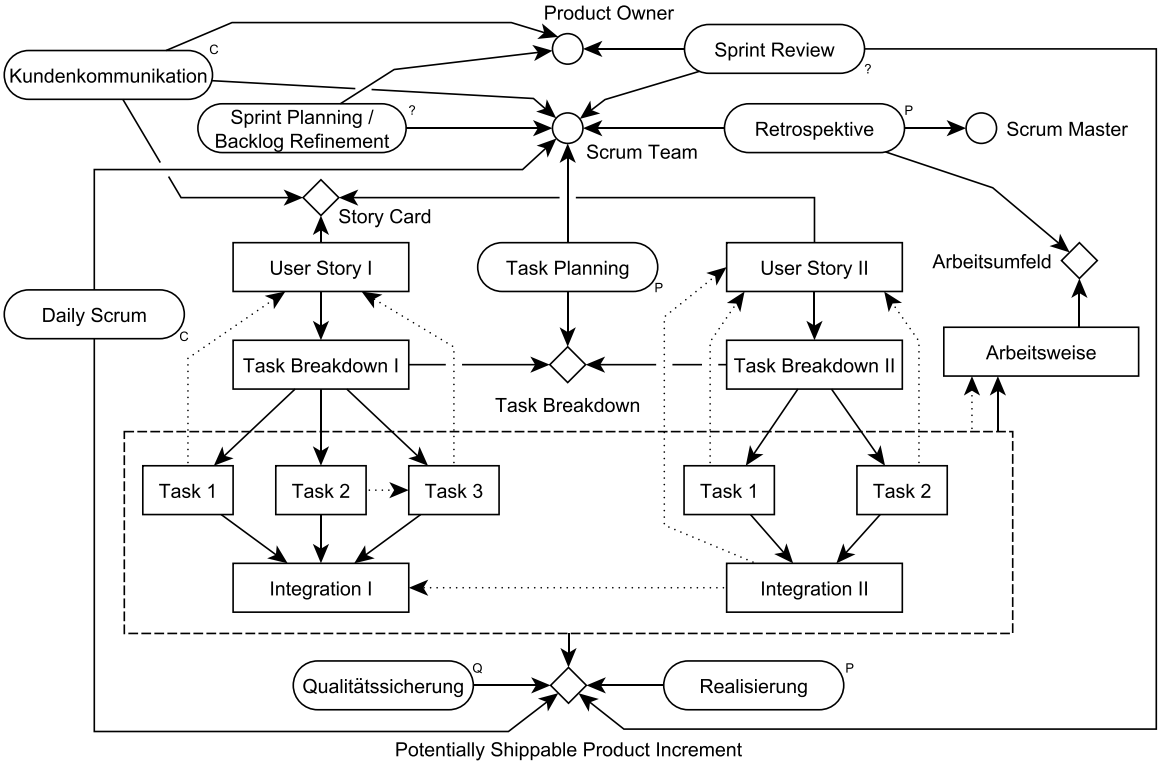


Abbildung 4.16.: Mögliches Szenario für ein Planspiel zu Scrum

Aufbau des Produkts

Basis für die Struktur des Produkts sind die *User Stories* auf denen die vom Kunden gewünschten Features definiert sind. Eine zu erstellende Produkteinheit besteht aus folgenden Artefakten:

- Eine **User Story** beschreibt das Feature, welches die Produkteinheit umsetzen soll. Sie wird als Artefakt des Typs *Story Card* realisiert, das neben den Informationen, was umgesetzt werden soll, auch den geschätzten Umfang in Story Points und die Priorität des Features enthält.
- Zu jeder User Story existiert eine bestimmte Menge unterschiedlicher **Tasks**, welche die eigentliche Umsetzung der Produkteinheit darstellen.
- Alle Tasks einer User Story werden über ein Artefakt **Task Breakdown** bedingt. Es repräsentiert das Zerlegen der Gesamtaufgabe in einzelne Teilaufgaben. Das Artefakt hängt von der jeweiligen User Story ab und hat einen geringen Umfang. Erst wenn es vollständig erstellt wurde, können die darunterliegenden Tasks bearbeitet werden.
- Jede Produkteinheit enthält ein **Integrationsartefakt**. Dieses Artefakt kann erstellt werden, sobald alle Tasks vollständig sind. Es stellt die Integration der Produkteinheit in das Gesamtsystem dar und kann deshalb auch von anderen Produktteilen über eine Kontextbeziehung abhängig sein.

Tasks und Integration stellen die *Potentially Shippable Product Increments* dar, und sind damit das, was der Product Owner am Ende eines Sprints begutachtet und abnimmt. Das Gesamtprodukt besteht aus mehreren der oben beschriebenen Produkteinheiten, die gegebenenfalls auch untereinander durch Kontextbeziehungen gekoppelt sein können. Das kann sowohl die Integrationsartefakte betreffen, als auch einzelne Tasks, die auf Ergebnissen anderer Produkteinheiten basieren. Außerdem können auch Tasks einer Produkteinheit untereinander in Beziehung stehen. Dabei sind alle der im Simulationsmodell vorhandenen Beziehungstypen möglich.

Zusätzlich sind einige der Potentially Shippable Product Increments über Kontextbeziehungen von der entsprechenden User Story abhängig. Um diese entwickeln zu können, ist es wichtig, genau zu wissen, was der Product Owner möchte. Dies geht typischerweise über den Text auf der Story Card hinaus und erfordert die Kommunikation mit dem Product Owner.

Das Product Backlog ist die Menge aller aufgedeckter User Stories. Bei der Entwicklung des Szenarios werden alle möglichen Stories definiert. Da sich im Verlauf des agilen Prozesses Anforderungen typischerweise ändern, ist es sinnvoll dem Spieler zu Beginn des Spiels nicht alle Stories zur Verfügung zu stellen. Im Verlauf des Spiels können sich ändernde Anforderungen durch das Hinzufügen und Entfernen der vordefinierten Stories realisiert werden.

Das Sprint Backlog wird ebenfalls in der Rahmenanwendung verwaltet. Es enthält die im Rahmen eines Sprint Plannings ausgewählten Stories für den kommenden Sprint. Die Rahmenanwendung muss dem Spieler dazu die Möglichkeit geben, Stories für den kommenden Sprint zu markieren.

Aktivitäten und Rollen

Um die Artefakte zu erstellen, stehen mehrere Aktivitäten zur Verfügung, die mit den für Scrum üblichen Rollen verknüpft sind. Es sind folgende Aktivitäten definiert:

- Um die Tasks einer User Story aufzudecken, wird die Aktivität **Task Planning** vom Scrum Team durchgeführt. Mit einem geringen Zeitaufwand wird das dazugehörige Task-Breakdown-Artefakt vervollständigt und damit die davon abhängigen Tasks sichtbar.

- Tasks und Integrationsartefakte können mit den Aktivitäten **Realisierung** und **Qualitätssicherung** bearbeitet werden. Aktivitäten und Artefakte unterscheiden sich in diesem Punkt nicht von den entsprechenden Aktivitäten in Szenarien für lineare Prozessmodelle.

- Im Rahmen des **Daily Scrum** wird Wissen über die aktuell zu entwickelnden bzw. schon fertiggestellten Produktteile ausgetauscht. Entsprechend wird Daily Scrum als Kommunikationsaktivität konzipiert, die auf die entsprechenden Artefakttypen angewendet werden kann.

- Für eine erfolgreiche Durchführung eines Projekts mit Scrum ist eine kontinuierliche **Kommunikation mit dem Kunden** notwendig, um die benötigten Informationen über die zu entwickelnden Produktteile zu erhalten. Der Product Owner wird dazu so konfiguriert, dass er über das vollständige Wissen der Story-Card-Artefakte verfügt. Die zu entwickelnden Produktteile sind über Kontextbeziehungen mit diesen Artefakten verknüpft, sodass das Wissen darüber Fortschritt und Qualität bei der Erstellung maßgeblich beeinflusst.

- Die Aktivität **Sprint Planning / Backlog Refinement** unterscheidet sich von den anderen Aktivitäten des Szenarios, da sie nicht über das Simulationsmodell umgesetzt werden kann, sondern eine gesonderte Implementierung erfordert. Wird diese Aktivität von Product Owner und Scrum Team gemeinsam durchgeführt, werden die User Stories im Product Backlog verändert, und die Prioritäten und Story Points angepasst. Dieser Vorgang muss außerhalb des Simulationsmodells in der Rahmenimplementierung umgesetzt werden.

  Um das Aufdecken und die Priorisierung der User Stories umsetzen zu können, ist es sinnvoll eine Abhängigkeit zwischen den Stories zu definieren und eine Storyline zu entwerfen, die angibt, welche Story Card wann und wie sichtbar wird.

  Im Rahmen des Sprint Plannings muss es dem Spieler ermöglicht werden, User Stories im Product Backlog für den kommenden Sprint zu markieren. Diese Auswahl bildet die Bewertungsgrundlage für das Review am Ende des Sprints.

- Das **Sprint Review** ist die zweite besondere Aktivität in diesem Szenario, die nicht vom Modell abgedeckt werden kann. Sie muss passend zum Sprint Planning implementiert werden, um auf Basis des dort festgelegten Sprint Backlogs die im Sprint geschaffenen Produktteile zu begutachten und gegebenenfalls freizugeben. Welche User Stories freigegeben wurden, beeinflusst gegebenenfalls die Änderungen an Product Backlog und Priorisierung der noch offenen User Stories im Rahmen des Backlog Refinements.

Retrospektive

Ein wichtiger Teil von Scrum ist die kontinuierliche Verbesserung der Arbeitsweise und Steigerung von Effektivität und Effizienz des Teams. Dazu wird am Ende jedes Sprints eine Retrospektive durchgeführt, bei der die aktuelle Arbeitsweise kritisch betrachtet, und bei Bedarf geändert und optimiert wird. Da das Simulationsmodell die Aspekte des Arbeitsumfelds nicht abdeckt, kann dieser Aspekt der agilen Entwicklung mit Scrum nur indirekt umgesetzt werden.

Dazu wird ein zusätzliches Artefakt eingeführt, das die Arbeitsweise abstrahiert. Dieses Artefakt kann als Dokument aufgefasst werden, indem davon ausgegangen wird, dass die einzuhaltende Arbeitsweise dort beschrieben wird. Im Rahmen der Retrospektive wird dieses Dokument auf Basis der gemachten Erfahrungen erstellt und weiterentwickelt. Die Aktivität *Retrospektive* ist somit eine Fortschrittsaktivität, die auf dieses Dokument angewendet werden kann. Um den Einfluss der Arbeitsweise auf die Entwicklung zu realisieren, werden entsprechende Kontexteinflüsse zwischen den zu entwickelnden Produktartefakten und dem Artefakt für die Arbeitsweise festgelegt.

Die Weiterentwicklung der Arbeitsweise basiert auf den Erfahrungen, die in den einzelnen Sprints gemacht wurden. Um zu verhindern, dass ein Spieler zu Beginn des Spiels mittels einer langen Retrospektive das gesamte Artefakt Arbeitsweise erstellt, sollten Vorbedingungen eingefügt werden, die verhindern, dass die Arbeitsweise weiter entwickelt wird, als es die durch die Entwicklung der Produktteile gemachten Erfahrungen zulassen.

Bei Bedarf können anstelle eines einzelnen Artefakts, das alle von der Retrospektive behandelten Aspekte enthält, auch mehrere Artefakte eingesetzt werden. Dadurch erhält der Spieler die Möglichkeit, bei der Retrospektive unterschiedliche Schwerpunkte zu setzen. Der Entwickler des Spiels erhält gleichzeitig die Möglichkeit den Einfluss unterschiedlicher Aspekte auf die einzelnen Aufgaben detaillierter zu bestimmen. So kann beispielsweise die Gestaltung des Arbeitsplatzes einen großen Einfluss auf die Kommunikation haben, weil der Product Owner jetzt im Büro nebenan sitzt, während die Verbesserung der verwendeten Werkzeuge die Produktivität bei der Implementierung steigert.

Rahmenanwendung eines entsprechenden Spiels

In den vorherigen Abschnitten wurden an einigen Stellen Anpassungen und Implementierung erwähnt, die in der Rahmenanwendung vorzunehmen sind. Dazu gehören vor allem die Gestaltung eines Scrum Boards mit Product Backlog und Sprint Backlog und ein geeignetes Hinzufügen von neuen bzw. das Entfernen bestehender User Stories im Rahmen des Sprint Plannings.

Je nach Lernziel müssen die Möglichkeiten, die der Spieler hat, stärker auf den Scrum Prozess eingeschränkt werden, um zu verhindern, dass er davon abweicht. Das Szenario macht keine Einschränkungen, wann welche Aktivität angewendet werden kann. So ist es zunächst möglich Aktivitäten wie Sprint Planning auch während des Sprints durchzuführen, was der Idee hinter Scrum jedoch widersprechen würde. Die Rahmenanwendung muss dazu ein Konzept des Sprints einführen und das Szenario entsprechend einschränken und weiter regulieren.

Alle anderen Aspekte werden vom Simulationsmodell abgedeckt. Es berechnet die Veränderungen der einzelnen Produktteile, berücksichtigt dabei die angegebenen Zusammenhänge, das erforderliche Wissen und die dafür benötigte Kommunikation. Lediglich die Beurteilung und Freigabe der Ergebnisse im Rahmen des Sprint Reviews anhand der Qualität und Vollständigkeit der Produktartefakte, muss zusätzlich implementiert werden. Entsprechende Kriterien können beispielsweise in einer Storyline definiert werden, die Grundlage für die vom Product Owner eingebrachten User Stories, deren Priorisierung und Abnahmekriterien ist, und in den Aktivitäten Sprint Planning und Sprint Review umgesetzt wird.

5 Bewertung und Diskussion

Mit dieser Arbeit wurde ein Simulationsmodell für Planspiele im Bereich der Softwaretechnik und des Projektmanagements vorgestellt. Ziel der Arbeit war es, den Entwicklungsaufwand solcher Planspiele zu reduzieren, die Schwierigkeit herabzusetzen und die Entwicklungszeit zu verkürzen.

Dazu wurde ein Modell, basierend auf mehreren Elementarmodellen entwickelt, mit dem die Vorgänge bei der Durchführung eines Softwareprojekts simuliert werden können. Um ein möglichst allgemeingültiges und flexibel einsetzbares Modell zu erhalten, wurde dabei von einem konkreten Projekt abstrahiert. Aufbau und Funktionsweise des Modells wurden in Kapitel 2 ausführlich beschrieben.

Das so entstandene Modell wurde in einer Evaluation, wie in Kapitel 3 beschrieben, umfassend untersucht. Durch unterschiedliche Experimente wurden dabei 25 Hypothesen am Modell überprüft. Die Evaluation wurde durch einige ergänzende Experimente abgerundet. Dabei hat sich gezeigt, dass das Modell das in der entsprechenden Literatur beschriebene Verhalten erzeugt und die dort angegebenen Effekte aufweist, und somit geeignet ist, die in den Planspielen auftretenden Situationen adäquat zu simulieren.

Um zu zeigen, dass Simulationsmodell und Framework dazu geeignet sind, um damit Planspiele zu realisieren, wurden dessen Einsatzmöglichkeiten anhand einiger Beispiele in Kapitel 4 aufgezeigt. Weiterhin wurde dort ein Vorgehen zur Konzeption und Entwicklung von Planspielen vorgeschlagen und Simulationsmodell und Framework in dieses eingeordnet.

Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen und vergleicht sie mit den am Anfang besprochenen Zielen. Anschließend werden die Ergebnisse im Vergleich mit anderen Ansätzen betrachtet. Das Kapitel schließt mit einer Betrachtung der Erweiterungsmöglichkeiten des Simulationsmodells und einer kurzen Betrachtung seines Potentials außerhalb der Planspielentwicklung.

5.1. Zusammenfassung und Bewertung

Diese Arbeit beschreibt ein neuartiges Simulationsmodell. Die wesentlichen Ergebnisse der Arbeit sind:

- Ein Datenmodell (Abschnitt 2.3), mit dessen Hilfe Softwareprojekte abstrakt dargestellt werden können, und das eine Basis für die Simulation unterschiedlicher Projekte mit einem gemeinsamen Simulationsmodell bildet. Das Datenmodell ist unabhängig von einem konkreten Einsatzszenario und kann für eine Vielzahl unterschiedlicher Planspiele verwendet werden.
- Ein auf diesem Datenmodell aufbauendes Simulationsmodell (Abschnitt 2.4 und Abschnitt 2.5), mit dem die zeitlichen Verläufe der einzelnen Projekte berechnet werden können.
- Ein eigenständiges Teilmodell zur Planung und Ausführung individuellen Verhaltens der einzelnen Projektbeteiligten (Abschnitt 2.6), basierend auf psychologischen Modellen der Handlungsplanung.
- Ein Framework (Abschnitt 2.2), welches das Simulationsmodell, eine geeignete Ausführungsumgebung, sowie einigen Werkzeugen enthält, und die Ergebnisse dieser Arbeit dadurch leichter verwendbar macht.

Das Simulationsmodell basiert dabei auf einem Ansatz, bei dem Effekte durch die Emergenz weniger elementarer Teilmodelle entstehen, welche die grundlegenden Aspekte bei der Erstellung von Softwareprojekten nachbilden. Dieser Ansatz und die grundsätzliche Konzeption des Simulationsmodells haben folgende Vorteile:

- Das Simulationsmodell basiert auf wenigen elementaren Teilmodellen. Durch deren Zusammenspiel werden die gewünschten Effekte erzeugt. So muss nicht jeder Effekt einzeln umgesetzt werden. Unerwünschte Wechselwirkungen zwischen den verschiedenen Effekten können durch diesen Ansatz geringgehalten werden. Die Auswahl und Umsetzung, sowie die Kombination der einzelnen Teilmodelle, werden in dieser Arbeit in Abschnitt 2.4 ausführlich beschrieben.

- Durch seinen zustandsbasierten Ansatz (Abschnitt 2.7) erlaubt das Simulationsmodell beliebige zusätzliche Manipulationen des Projektzustands während der Simulationsausführung. Durch die variable Schrittlänge lässt sich jeder beliebige Zeitpunkt im Simulationsverlauf erreichen, um dort eine solche Manipulation durchzuführen.

- Die verwendeten Teilmodelle entspringen wissenschaftlich fundierten Arbeiten und empirischen Studien. Jedes Teilmodell behandelt dabei einen begrenzten Aspekt, der sich leichter untersuchen und belegen lässt als komplexe Zusammenhänge. Erst durch die Kombination der Teilmodelle wird ein komplexes Verhalten generiert.

- Durch den modularen Aufbau des Simulationsmodells (Abschnitt 2.7) lassen sich dessen Teile einzeln betrachten, verwenden und austauschen. Dies unterstützt besonders die Entwicklung und Erweiterbarkeit des Modells.

Ein wesentlicher Aspekt des in dieser Arbeit verfolgten Ansatzes ist die Berücksichtigung der Individualität der einzelnen am Projekt beteiligten Mitarbeiter. Dieser Aspekt wird in dieser Arbeit durch folgende Punkte abgedeckt:

- Die simulierten Projektbeteiligten werden als einzelne Individuen aufgefasst. Sie besitzen einen eigenen Charakter und können selbstständig im Projekt agieren, wie es auch im Original der Fall ist.

- Durch die auf einem Multi-Agenten-System basierende Architektur der Simulation (Abschnitt 2.1.3), agieren die einzelnen Mitarbeiter zunächst unabhängig. Es existiert keine zentrale Koordination. Das Zusammenspiel der einzelnen Mitarbeiter kommt durch entsprechende Absprachen direkt zwischen den Mitarbeitern, oder über den Projektleiter, in Form des Projektplans zustande.

- Mitarbeiter haben individuelle Charakterzüge und eigenständige Eigenschaften (Abschnitt 2.3). Auf Basis derer wird ein individuelles Verhalten für jeden Mitarbeiter generiert (Abschnitt 2.6.2). Grundlage dafür ist die aus der Psychologie stammende Valenz-Instrumentalitäts-Erwartungs-Theorie nach Vroom und die Motivationstheorie nach Pritchard und Ashwood.

- Da die einzelnen Mitarbeiter als unabhängige Agenten umgesetzt sind, lassen sich diese problemlos zusätzlich individualisieren und dadurch an entsprechende Planspiele anpassen.

Um die Validität des Simulationsmodells sicherzustellen wurden folgen Schritte unternommen:

- Auf Basis der Fachliteratur wurden 25 Hypothese aufgestellt, die Effekte beschreiben, welche das Simulationsmodell aufweisen soll (Abschnitt 3.1). Es wurde gezeigt, dass alle 25 Hypothesen für das Simulationsmodell akzeptiert werden können (Abschnitt 3.4.1).

- Ergänzend zur Untersuchung einzelner Hypothesen, wurde der Zusammenhang zwischen einer Vielzahl unterschiedlicher Aspekte möglicher Szenarien und der dazugehörigen Projektverläufe untersucht, um unerwünschtes Verhalten aufzudecken (Abschnitt 3.4.2).

- Die grundsätzliche Einsetzbarkeit des Simulationsmodells wurde anhand eines Beispielprojekts gezeigt, bei dem ein vollständiges Planspiel mit dem Simulationsmodell entwickelt wurde (Abschnitt 4.3).

- Die flexible Einsetzbarkeit des Simulationsmodells wurde anhand von mehreren Ausschnitten unterschiedlicher Anwendungsszenarien diskutiert (Abschnitt 4.4).

Die Anwendbarkeit des Simulationsmodells wird durch folgende Punkte unterstützt:

- Alle Ergebnisse dieser Arbeit, insbesondere das Simulationsmodell, sind in ein gemeinsames Framework eingebettet (Abschnitt 2.2). Dieses Framework kann zur Erstellung von Planspielen verwendet werden. Um den Planspielentwickler nicht einzuschränken, wurde auf Generatoren und Middleware verzichtet. Stattdessen können alle Teile direkt als Codebibliothek in entsprechende Rahmenprogramme eingebunden werden.

- Durch den konsequent verfolgten objektorientierten Ansatz und den modularen Aufbau des Simulationsmodells, kann dieses ohne großen Aufwand angepasst und erweitert werden (Abschnitt 2.7). Das ermöglicht die Weiterentwicklung und den langfristigen Einsatz des Modells.

- Das Framework enthält Werkzeuge, die sowohl den Modell- als auch den Planspielentwickler bei der Arbeit mit dem Simulationsmodell unterstützen (Abschnitt 4.1).

Im Rahmen dieser Arbeit wurden vier Forschungsfragen gestellt (vgl. Abschnitt 1.4), die im Folgenden diskutiert werden.

RQ1: Effektgenerierung durch Emergenz elementarer Teilmodelle

Bisherige Modelle setzen die im Planspiel benötigten Effekte meist direkt um. Dabei muss jeder Effekt einzeln modelliert werden. Gleichzeitig muss sichergestellt werden, dass sich die Effekte gegenseitig nicht negativ beeinflussen und so das Simulationsergebnis verfälschen. Der Ansatz dieser Arbeit war, ein Simulationsmodell auf Basis weniger elementarer Teilmodelle zu entwickeln, dass die geforderten Effekte durch Emergenz seiner Teile erzeugt. Dazu wurden Modelle aus der Psychologie und den Arbeitswissenschaften verwendet, die wissenschaftlich belegt sind.

Die erste Forschungsfrage sollte klären, ob ein Modell auf diese Weise gebaut werden kann, und wie gut sich damit die entsprechenden Effekte erzeugen lassen.

Die Ergebnisse dieser Arbeit zeigen, dass es grundsätzlich möglich ist, ein Simulationsmodell für Planspiele auf diese Weise zu konstruieren. Das so entstandene Modell umfasst eine überschaubare Anzahl an Teilmodellen, die durch ihre Kombination eine größere Anzahl an Effekten erzeugen.

Dass viele der benötigten Teilmodelle gut erforscht und empirisch belegt sind, ist ein großer Vorteil dieses Ansatzes. Außerdem können die einzelnen kleinen Teilmodelle leichter nachvollzogen werden, als ein großes unstrukturiertes Modell, was das Gesamtmodell insgesamt besser beherrschbar macht.

Die Schwierigkeit dieses Ansatzes liegt in der richtigen Kombination der Teilmodelle. Vor allem für die aus der Psychologie stammenden Modelle sind häufig keine genauen Schnittstellen definiert, und die Skalen und Wertebereiche der in das Modell eingehenden und ausgehenden Werte sind nicht klar. Die Modelle beschreiben außerdem zwar Zusammenhänge zwischen unterschiedlichen Eigenschaften, verzichten dabei jedoch meist auf entsprechende Quantisierungen.

Vor Verwendung eines solchen Teilmodells müssen daher zunächst dessen Schnittstellen festgelegt werden. Dabei muss darauf geachtet werden, dass diese zu den Schnittstellen der anderen Teilmodelle bzw. dem unter der Simulation liegenden Datenmodell passen. Dazu wurden im Rahmen dieser Arbeit entsprechende Datentypen und Wertebereiche festgelegt, und die Teilmodelle an einem gemeinsamen Datenmodell ausgerichtet.

Aufbau und Ablauf der Evaluation, und insbesondere die dort untersuchten Effekte, wurden erst nach der Erstellung des Simulationsmodells festgelegt. Während der Konzeptions- und Entwicklungsphase des Modells waren die Effekte, die vom Modell später erwartet wurden, nicht bekannt. Das Modell wurde dementsprechend nicht speziell auf diese Effekte bezogen entwickelt. Trotzdem konnte alle gewünschten Effekte im Rahmen der Evaluation später nachgewiesen werden. Das lässt erwarten, dass das Simulationsmodell noch weitere, bislang nicht untersuchte Effekte aufweist, die im Bereich von Softwareprojekten relevant sind. Werden Effekte in einem Modell explizit umgesetzt, lassen sich die einzelnen Effekte zwar besser kontrollieren, es besteht jedoch die Gefahr, dass relevante Effekte vergessen, und somit nicht durch das Modell abgedeckt werden, und dass es zu einer ungewollten Wechselwirkung zwischen den einzelnen Effekten kommt. Da das in dieser Arbeit entwickelte Simulationsmodell versucht Effekte auf die gleiche Weise zu erzeugen, wie sie auch im Original zustande kommen, ist diese Gefahr hier geringer.

RQ2: Flexibilität und Konfigurierbarkeit des Modells

Ein Ziel dieser Arbeit war es, ein allgemeines Simulationsmodell zu entwickeln, dass nicht nur für ein bestimmtes Planspiel eingesetzt werden kann, sondern eine Vielzahl unterschiedlicher Spiele und Spielkonzepte unterstützt. Dazu muss das Modell zum einen so allgemein sein, dass es sich in dieser Breite einsetzen lässt, zum anderen muss es für die einzelnen konkreten Planspiele angepasst und konfiguriert werden können.

Ein wesentlicher Schritt, um diese Anforderungen umzusetzen, war die Entwicklung des Datenmodells, auf dem das Simulationsmodell arbeitet. Durch einen entsprechend hohen Abstraktionsgrad von Artefakten und Aktivitäten, können diese sehr flexibel eingesetzt werden. Die eigentliche Anpassung an die konkrete Situation findet dabei über die Zusammenhänge zwischen den einzelnen Entitäten des Szenarios und deren Parametrisierung statt. Dabei beschränken sich die Eigenschaften auf die Teile, welche vom Simulationsmodell benötigt werden. Andere Aspekte, die zwar für das Planspiel relevant sind, die Berechnungen im Simulationsmodell jedoch nicht beeinflussen, sind nicht Teil des Datenmodells. Dazu gehören unter anderem die Anzeigekonzepte, über die der Simulationszustand dem Spieler zugänglich gemacht wird, und die Auswertung und Bewertung des Spielverlaufs. Diese planspielspezifischen Aspekte werden ausschließlich in der Rahmenanwendung umgesetzt.

Neben der Konfiguration der Szenarien ist es auch wichtig, bestimmte Situationen im Projektverlauf zuverlässig herstellen zu können. Durch die Möglichkeit, den Simulationszustand jederzeit beliebig verändern zu können, lassen sich diese Situationen jederzeit herstellen, auch wenn sie nicht im Simulationsmodell vorgesehen sind. Insbesondere bei der Evaluation des Modells hat sich gezeigt, dass dies in den allermeisten Fällen problemlos mit sehr wenig zusätzlichem Code möglich ist. Hier wurden nie mehr als 15 Zeilen Code benötigt, um die gewünschte Änderung zu realisieren. Das Simulationsmodell selbst wird dabei nicht verändert.

Dass das Simulationsmodell flexibel für die unterschiedlichsten Szenarien eingesetzt werden kann, zeigen unter anderem die Beispiele in Kapitel 4 und die Modellierungsvorschläge zu weiteren Aspekten, die in Anhang A.5 zu finden sind.

An einigen Stellen reicht das Datenmodell nicht aus, um alle für ein Planspiel relevanten Eigenschaften abzulegen. Durch den objektorientierten Ansatz, der eine Erweiterung aller im Datenmodell definierten Klassen ermöglicht, ist es nicht nötig, eine zweite parallele Datenstruktur zu erstellen. Stattdessen wird das Datenmodell um die zusätzlich benötigten Aspekte erweitert. Das funktioniert gleichermaßen auch für das Simulationsmodell und dessen Teilemodelle.

RQ3: Anforderungen an das Simulationsmodell

Zu Beginn der Entwicklung des Simulationsmodells wurden sechs Anforderungen an das Modell festgelegt (vgl. Abschnitt 1.4). Diese konnten im Wesentlichen erfüllt werden:

G1 Unabhängigkeit des Modells von konkreten Planspielen. Bei der Entwicklung des Simulationsmodells wurden nur allgemeine Anforderungen von Planspiele berücksichtigt. Das entstandene Modell ist unabhängig von konkreten Einsatzszenarios. Um das Modell für ein konkretes Planspiel einzusetzen, muss eine passende Rahmenanwendung realisiert werden, welche die planspielspezifischen Aspekte enthält. Das Modell wird, wie im Rahmen von RQ2 diskutiert, durch die Definition eines entsprechenden Szenarios auf das Planspiel angepasst.

G2 Darstellung menschlicher Aspekte. Das Simulationsmodell enthält ein eigenes Teilmodell zur Simulation menschlichen Verhaltens. Der Simulationsansatz geht dabei über ein einfaches Regelwerk hinaus, und berücksichtigt sowohl rationale als auch emotionale Aspekte der Handlungsplanung. Dazu wurden Modelle aus der Psychologie verwendet und auf das Simulationsmodell hin angepasst.

Die Handlungsplanung wird durch die individuellen Charaktereigenschaften der einzelnen Projektbeteiligten ergänzt, die deren Verhalten und Leistung beeinflussen. Diese Umsetzung erlaubt die Realisierung individueller Projektbeteiligter, die menschliches Verhalten nachbilden, und damit in dem beschränkten Rahmen des Simulationsmodells eine realistische Darstellung menschlichen Handelns ermöglichen.

G3 Mächtigkeit des Simulationsmodells. Ein Ziel dieser Arbeit war es, ein Simulationsmodell zu entwickeln, das möglichst viele Aspekte der Softwareentwicklung abdeckt. Durch die Evaluation konnte gezeigt werden, dass das erstellte Simulationsmodell eine große Zahl unterschiedlicher Aspekte behandelt. Maßstab dafür sind die in der Literatur für Softwareprojekte beschriebenen Effekte.

Der Aufwand, der betrieben werden muss, um die gewünschten Effekte zu erzeugen ist relativ gering. Es hat sich gezeigt, dass in den allermeisten Fällen eine passende Konfiguration des Szenarios ausreichend ist, um eine entsprechende Ausgangssituation herzustellen, in welcher der jeweilige Effekt auftritt.

G4 Anpassbarkeit und Erweiterbarkeit. Das in dieser Arbeit vorgestellte Simulationsmodell deckt schon sehr viele Aspekte der Softwareentwicklung ab. Es erhebt jedoch in keinerlei Hinsicht den Anspruch auf Vollständigkeit. Egal wie umfangreich ein Modell ist, es gibt immer Aspekte, die nicht enthalten sind. Stellt sich heraus, dass ein nicht berücksichtigter Aspekt in einem Planspiel benötigt wird, so ist es notwendig, diesen in das Modell einzufügen. Um das zu ermöglichen, ist das Simulationsmodell nach den Prinzipien der Objektorientierung und stark modular aufgebaut. Seine Architektur ermöglicht den Austausch und das Hinzufügen einzelner Aspekte. Die Architektur ist dabei so gewählt, dass die Änderungen möglichst lokal durchgeführt werden können, ohne dabei viel des bestehenden Codes verändern zu müssen.

Stellt sich in Zukunft heraus, dass einzelne Aspekte des Simulationsmodells noch zu unpräzise, ungenügend oder falsch umgesetzt wurden, können diese problemlos ausgetauscht werden, indem die jeweiligen Teilmodelle ersetzt werden. Das im Framework vorhandene Evaluationspaket, sowie die grafische Oberfläche, unterstützen den Entwickler dabei, sicherzustellen, dass Änderungen am Simulationsmodell keine unerwünschten Nebeneffekte verursachen, und die bestehenden Effekte nicht zerstört werden.

G5 Reproduktion des Simulationsverlaufs. Eine vollständige Reproduktion eines Simulationsverlaufs wurde nur mit kleinen Einschränkungen erreicht. Prinzipiell lässt sich jeder Simulationsverlauf vollständig und exakt reproduzieren, wenn von einem identischen Startzustand und unveränderten Einflüssen während des Projektverlaufs ausgegangen wird. Das beinhaltet auch die Zustände der jeweils verwendeten Zufallsgeneratoren.

Wird der Zustand der Simulation verändert, hat dies einen direkten Einfluss auf die davon abhängigen Aspekte. Es können jedoch auch indirekte Einflüsse auftreten, die durch die verwendeten Zufallsgeneratoren zustande kommen. Wird aufgrund der Änderung beispielsweise ein zusätzlicher Zufallswert benötigt, verändern sich dadurch auch die anderen Zufallswerte. In den allermeisten Fällen sind die Auswirkungen sehr gering und im Simulationsergebnis nicht sichtbar. In einzelnen Situationen kann es jedoch vorkommen, dass durch eine Verkettung von einzelnen Effekten ein unerwartetes Ergebnis entsteht. Beispielsweise kann sich durch eine entsprechende Konstellation die Ausführungsreihenfolge der Agenten ändern. Dies kann wiederum zu einem veränderten Verhalten, und somit zu anderen Simulationsergebnissen führen. Das Ergebnis wird dadurch nicht falsch – menschliches Verhalten ist nicht deterministisch, und alle zufällige Werte in der Simulationsberechnung werden bewusst eingesetzt – es wird jedoch gegebenenfalls suggeriert, dass die Änderung im Simulationsergebnis auf die Änderung der Ausgangssituation zurückzuführen wäre. Dies ist zwar technisch gesehen richtig, dieser erlebte Zusammenhang ist jedoch nicht repräsentativ und wird in vergleichbaren Situationen mit hoher Wahrscheinlichkeit nicht auftreten.

Um diese Schwachstelle zu beheben, müsste für jeden Zufallswert ein unabhängiger Zufallsgenerator eingesetzt werden. Da eine solche Menge an Zufallsgeneratoren das Simulationsmodell unnötig unübersichtlich macht, und die oben beschriebenen Probleme nur sehr selten auftreten, wurde darauf bewusst verzichtet.

G6 Geringer Zeitaufwand für die Konfiguration der Simulationsparameter. Das Simulationsmodell soll mit einem geringen Zeitaufwand verwendet werden können. Dazu gehört insbesondere der Zeitaufwand, der in die Erstellung der Szenarien und die Konfiguration deren

Parameter investiert werden muss. Gegenüber der Erstellung eines neuen Simulationsmodells ist der dafür benötigte Zeitaufwand sehr gering.

Die Anzahl der festzulegenden Parameter ist vom Umfang und Detailgrad des gewünschten Szenarios abhängig. Ab einer bestimmten Größe kann es sinnvoll sein, Teile davon zu generieren, da ansonsten eine große Zahl unterschiedlicher Parameter von Hand belegt werden muss.

Im Rahmen der Entwicklung des Simulationsmodells sind eine Vielzahl unterschiedlicher Szenarien entstanden. Eine Auswahl davon wurde in dieser Arbeit vorgestellt. Der dafür benötigte Zeitaufwand belief sich, abhängig vom Umfang des Szenarios, auf wenige Minuten bis hin zu mehreren Tagen. Insgesamt scheint der zu investierende Aufwand der jeweiligen Komplexität und des jeweiligen Umfangs des Szenarios angemessen.

Dass für alle Parameter Wertebereiche vorgegeben sind, erleichtert die Konfiguration. Es hat sich herausgestellt, dass es häufig hilfreich ist, zunächst die Standardwerte der einzelnen Parameter zu verwenden, und diese nach und nach zu verändern. Mittels der im Framework enthaltenen Werkzeuge lassen sich die Auswirkungen solcher Änderungen gut nachvollziehen.

RQ4: Grenzen des Ansatzes

Im Rahmen dieser Arbeit wurden auch die Grenzen des gewählten Ansatzes untersucht. Grundlage dafür sind vor allem die persönlichen Erfahrungen, die bei der Entwicklung und Anwendung des Modells gemacht wurden.

Ein Simulationsmodell auf Basis elementarer Teilmodelle aufzubauen, hat sich als sinnvoll herausgestellt. Durch die wenigen Teilmodelle lassen sich eine Vielzahl an Effekten erzeugen. Es hat sich jedoch gezeigt, dass es oft mit erheblichem Aufwand verbunden ist, Effekte, die nicht durch das Zusammenspiel der Teilmodelle zustande kommen, dem Simulationsmodell hinzuzufügen. Anstelle den Effekt über einen geeigneten Mechanismus direkt zu erzeugen, wie es bei den meisten herkömmlichen Ansätzen geschieht, muss analysiert werden, durch welche Zusammenhänge der gewünschte Effekt im Original zustande kommt, und warum das Simulationsmodell davon abweicht. Grund dafür ist erfahrungsgemäß entweder ein fehlendes Teilmodell, oder dass die vorhandenen Teilmodelle nicht ausreichend präzise sind.

Einige Modelle, die für Planspiele entwickelt wurden, dienen nicht nur der Simulation des Spielverlaufs, sondern gleichzeitig auch dem Verständnis der dort abgebildeten Zusammenhänge. Da diese Zusammenhänge direkt durch die dort verwendeten Modelle beschrieben werden, lassen sie sich häufig direkt am Modell erkennen. Insbesondere wenn Effekte in einzelnen voneinander isolierten Modellen vorliegen, lassen sie sich einfacher erkennen und verstehen. Im Simulationsmodell sind die Effekte nicht einzeln modelliert und sind daher am Modell nicht ohne weiteres zu erkennen. Das Modell selbst kann daher nicht als einfache Darstellung und Erklärung für bestimmte Effekte verwendet werden. Ob und wie dieses Defizit behoben werden kann, ist eine interessante Fragestellung für weiterführende Arbeiten auf diesem Gebiet.

5.2. Vergleich mit anderen Modellierungsansätzen

Modelle für Planspiele nehmen im Bereich der Modellierung von Softwareentwicklungsprozessen nur einen geringen Anteil ein. Eine wichtige Klasse sind hier die Prozessmodelle, mit Hilfe derer Abläufe für Softwareprojekten spezifiziert werden können. Im folgenden Abschnitt wird das Simulationsmodell von diesen Modellen abgegrenzt.

In einem weiteren Abschnitt werden die System-Dynamics-Modelle von Abdel-Hamid, Madnick und Madachy betrachtet, die, wie auch das in dieser Arbeit vorgestellte Simulationsmodell, der Simulation von Softwareentwicklungsprozessen dienen.

5.2.1. Abgrenzung zur Prozessmodellierung

Ziel der *Prozessmodellierung* ist es, Vorgehensweisen, Abläufe und Regeln aufzustellen, welche die Mitarbeiter anleiten und sie dabei unterstützen, komplexe und umfangreiche Arbeiten durchzuführen. Vertreter dieser Modellklasse sind beispielsweise die Vorgehensmodelle, die im Bereich der

Softwareentwicklung eingesetzt werden. Dazu gehören unter anderem das Wasserfallmodell und das V-Modell.

Das in dieser Arbeit vorgestellte Simulationsmodell ist kein Prozessmodell. Die dort definierten Regeln geben nicht vor, wie bestimmte Aufgaben und Entwicklungsschritte zu erledigen sind, sondern was passiert, wenn gewisse Handlungen durchgeführt werden. Im Gegensatz zu einem Prozessmodell ist es damit kein Vorbild für durchzuführende Handlungen, sondern ein Abbild des Originals. Dies gilt für alle Teile des Simulationsmodells, einschließlich der Modellteile zur Handlungsplanung, deren Regelwerk ebenfalls nicht versucht ein möglichst gutes Vorgehen zu definieren, sondern die Handlungen der Mitarbeiter im Original nachzubilden.

Prozessmodelle können jedoch auf das Simulationsmodell angewendet werden. Ziel vieler Planspiele im Bereich der Softwaretechnik ist es, ein bestimmtes Prozessmodell kennen- und anwenden zu lernen. Der Spieler verwendet dabei die im Prozessmodell vorgegebenen Regeln, um seine Mitarbeiter im simulierten Projekt zu steuern. Um die Möglichkeiten des Spielers auf einen vorgegebenen Prozess einzuschränken zu können, erlaubt das Datenmodell des Simulationsmodells die eingeschränkte Modellierung eines Prozesses. Durch *Rollen* und *Artefakttypen* lassen sich Aktivitäten auf Artefakte und Mitarbeiter einschränken. Mittels Vorbedingungen zwischen den einzelnen Produktartefakten lässt sich eine Bearbeitungsreihenfolge definieren. Das so definierte „Prozessmodell“ ist jedoch kein Teil des eigentlichen Simulationsmodells. Lediglich die Sicherstellung dessen Einhaltung wird von ihm unterstützt.

Prozessmodelle können auf dem Simulationsmodell ausgeführt werden. Bei Planspielen übernimmt dabei typischerweise der Spieler die Ausführung. Es besteht jedoch auch die Möglichkeit, die im Prozessmodell festgelegten Regeln als ausführbaren Code zu implementieren und durch einen Agenten des Simulationsmodells ausführen zu lassen. Im Rahmen der in Kapitel 3 beschriebenen Evaluation ist dies für einige der dort geprüften Hypothesen gemacht worden. Dabei wurde in der Regel geprüft, ob sich das Simulationsmodell bei Anwendung eines vorgegebenen Prozesses korrekt verhält.

5.2.2. Modelle zur Dynamik von Softwareentwicklungsprozessen

Es existieren verschiedene Arbeiten, die ebenfalls zum Ziel haben, die Vorgänge und Zusammenhänge bei der Entwicklung von Software in Modelle zu fassen. Zwei wichtige Vertreter sind die Werke *System Project Dynamics* von Tarek K. Abdel-Hamid und Stuart E. Madnick [Abdel-Hamid, 1991], und *System Process Dynamics* von Raymond J. Madachy [Madachy, 2007]. Die Autoren beschreiben dort mit Hilfe von *System-Dynamics-Modellen* (vgl. Abschnitt 2.1.3) die unterschiedlichen Aspekte von Softwareprojekten. Ihr erklärtes Ziel ist es dabei, eine integrative Sicht auf das Projektmanagement von Softwareprojekten zu entwickeln, und dadurch das Verständnis dafür zu verbessern. Dabei streben sie eine Mächtigkeit des Modells an, die es erlaubt dieses auch für Vorhersagen einzusetzen.

Sowohl Ziele als auch Ansatz unterscheiden sich damit von dem in dieser Arbeit vorgestellten Modell. Während das Simulationsmodell darauf ausgelegt ist, einen möglichst realistischen Verlauf eines Softwareprojekts unter den vorgegebenen Rahmenbedingungen und den entsprechenden Einflüssen während des Projektverlaufs zu erstellen, versuchen Abdel-Hamid/Madnick und Madachy die beobachtbaren Zusammenhänge in Einklang zu bringen und daraus ein allgemeingültiges Modell zu entwickeln.

Da das Simulationsmodell das Original nur verkürzt darstellen kann, sind die in die Simulationsberechnung eingehenden Eigenschaften nicht ausreichend, um das Projektergebnis bezogen auf das Original eindeutig zu beschreiben. Das bedeutet, würden die in einem Szenario definierten Ausgangsbedingungen im Original hergestellt, so wäre dort der Projektausgang durch diese Bedingungen trotzdem nicht eindeutig vorbestimmt. Der Verlauf des Projekts ist unterspezifiziert und es sind mehrere, im Allgemeinen unendlich viele Projektverläufe und -ergebnisse möglich. Das Simulationsmodell beschränkt sich darauf, einen dieser plausiblen Verläufe zu erzeugen.

Da die Modelle von Abdel-Hamid/Madnick und Madachy den Anspruch haben, auch zur Vorhersage von Projektverläufen verwendet werden zu können, ist dieser Ansatz dort nicht möglich. Da die exakten Zusammenhänge im Detail nicht bekannt sind, und sich Eigenschaften wie Motivation, Gesundheit und Fähigkeiten nicht genau messen lassen, können solche Vorhersagemodelle nur auf einem entsprechend hohem Abstraktionsgrad erstellt werden. Im so einem Modell können nur Zusammenhänge abgebildet werden, die im Original auch genau gemessen werden können. Abdel-Hamid hat dazu verschiedene

NASA-Projekte ausgewertet, um die Zusammenhänge in seinem Modell korrekt einzustellen. Das Modell ist dadurch stark an diesen Projekten orientiert, und es ist fraglich, wie zuverlässig es sich auf andere Projekte übertragen lässt.

Ein grundsätzlicher Unterschied dieser Arbeit zu den Arbeiten von Abdel-Hamid/Madnick und Madachy besteht im verwendeten Ansatz. Bei den Modellen zur Dynamik des Softwareentwicklungsprozesses wurde versucht, die beobachteten Zusammenhänge nachzubauen und ein konsistentes Bild der Beobachtungen in Form eines Modells zu erzeugen. Im Gegensatz dazu, war der Ansatz dieser Arbeit die Erklärungen für die beobachteten Effekte zu verwenden, um diese auf die gleiche Weise zu erzeugen, wie es auch im Original vermutet wird. Die so konstruierten Zusammenhänge können im Original nicht direkt gemessen werden, sondern müssen durch Rückschlüsse aus den gemachten Beobachtungen rekonstruiert werden. Dadurch ist das Simulationsmodell an manchen Stellen unpräziser als es die Modelle von Abdel-Hamid/Madnick und Madachy sind, kann dafür aber breiter eingesetzt werden.

Einer der Hauptgründe, warum die Modelle von Abdel-Hamid/Madnick und Madachy für die Verwendung in Planspielen ungeeignet sind, ist die Tatsache, dass die Projektmanagementstrategien Teil des Modells sind. Der Projektleiter, die Projektphilosophie, die einzelnen Entscheidungen, Richtlinien, Prozessmodelle und weiteren Aspekte, die in einem Planspiel der Spieler übernehmen muss, sind durch das Modell schon vorgegeben, und lassen sich auch nur schwer daraus entfernen. Ein weiterer Grund ist das hohe Abstraktionsniveau. Um in Planspielen ein anschauliches und einprägsames Spielerlebnis zu erzeugen, ist es notwendig, Details darzustellen. Dazu gehört beispielsweise die Motivation der einzelnen Mitarbeiter und deren Beziehungen untereinander. Das Abstraktionsniveau der Modelle von Abdel-Hamid/Madnick und Madachy lassen dies nicht zu. Mitarbeiter werden dort beispielsweise in Form einer Größe für die Arbeitskraft modelliert.

Abbildung 5.1 zeigt einen Ausschnitt des in [Madachy, 2007] beschriebenen Modells, das Brooks' Law realisiert. Der Entwicklungsprozess wird hier durch die Überführung von Anforderungen in entwickelte Software modelliert. Die Entwicklungsgeschwindigkeit wird durch die Softwareentwicklungsrate beschrieben. Diese wird von drei Faktoren beeinflusst: der nominalen Produktivität, dem neuen Personal und dem erfahrenen Personal. Anforderungen, entwickelte Software, neues Personal und erfahrenes Personal sind Variablen des Modells, die seinen Zustand beschreiben. Sie werden als einfache nummerische Werte repräsentiert. Die Einstellung neuer Mitarbeiter wird mittels der Personalzuweisungsrate beschrieben. Die Einarbeitung des neuen Personals ist über die Assimilierungsrate modelliert worden.

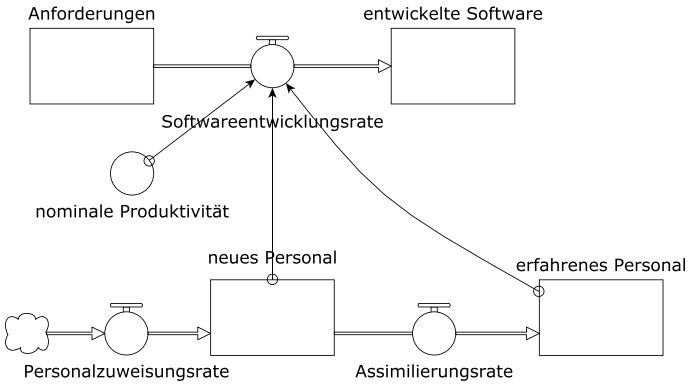


Abbildung 5.1.: Ausschnitt aus einem System Dynamics Modell zum Einfluss neuer Mitarbeiter auf die Produktivität [Madachy, 2007, S. 106]

An diesem Beispiel werden die Unterschiede im Abstraktionsgrad zwischen den Modellen von Abdel-Hamid/Madnick und Madachy und dem in dieser Arbeit beschriebenen Simulationsmodell deutlich. Das bezieht sich zum einen auf die Modellierung des Zustands. Das Simulationsmodell stellt beispielsweise das Produkt wesentlich detaillierter dar, indem es dessen einzelne Artefakte mit ihren verschiedenen Attributen im Detail beschreibt. Madachy hingegen definiert lediglich die Anzahl an noch nicht bearbeiteten und schon fertiggestellten Anforderungen.

Zum anderen ist die Simulationsberechnung bei Madachy auf einem deutlich gröberen Detailniveau angesetzt. So werden Anforderungen mittels einer einfachen Rate von unbearbeitet zu entwickelt übertragen. Diese Rate wird wiederum von der Größe der beiden im Modell vorhandenen Mitarbeitergruppen bestimmt. Dass sich die in diesen Gruppen befindlichen Mitarbeiter typischerweise unterscheiden, wird dabei nicht berücksichtigt. Gerade jedoch solche Details sind für Planspiele äußerst wichtig, da sie zum einen ein realistischeres Bild zeichnen, und zum anderen den Spieler lehren, auf solche Aspekte bei der Führung seines Projekts zu achten.

Die Modelle von Abdel-Hamid/Madnick und Madachy sind hervorragend geeignet, Zusammenhänge auf einem übersichtlichen Abstraktionsniveau zu begreifen. Durch die Feedbackschleifen, die hinter den Modellen stehen, lassen sich Einflüsse, die zwischen den einzelnen Aspekten herrschen, gut darstellen. Es könnte sich lohnen zu untersuchen, welche Vorteile es bringt solche Modelle in der Softwaretechniklehre einzusetzen. Für die Simulation von Projektverläufen in Planspielen sind sie jedoch in ihrer derzeitigen Form nicht geeignet, da sie zu wenig Details enthalten. Insbesondere auf individuelle Aspekte, beispielsweise einzelner Mitarbeiter oder bestimmter Produktartefakte, gehen sie nicht ein.

5.3. Vergleich mit Baukästen zur Planspielentwicklung

Neben den Projekten, die sich auf das Erstellen eines konkreten Planspiels konzentrieren, gibt es Arbeiten, die sich darüber hinaus auch mit der Planspielentwicklung an sich befasst haben. Mit *SESAM* und *SimSE* werden hier zwei wichtige Vertreter vorgestellt und mit den Ergebnissen dieser Arbeit verglichen.

5.3.1. SESAM

SESAM – ein Akronym für Software Engineering Simulation durch animierte Modelle – war vermutlich das erste Projekt, dass sich intensiv mit der Erstellung von Planspielen im Softwaretechnikbereich befasst hat (vgl. Abschnitt 1.2.1). Es ist Anfang der 90er Jahre in der Forschungsgruppe um Jochen Ludewig an der Universität Stuttgart entstanden. Die Informationen in diesem Abschnitt stammen im Wesentliche aus den Quellen [Ludewig, 1992], [Ludewig, 1994], und [Schneider, 1994].

Ziel von SESAM war es den Studierenden die Möglichkeit zu geben, Softwareentwicklung spielerisch erfahren zu können. In SESAM muss der Spieler ein Softwareprojekt zum Erfolg führen und wird dabei mit den typischen Herausforderungen und Schwierigkeiten solcher Projekte konfrontiert.

Darüber hinaus wurde im Rahmen von SESAM eine konzeptionelle Basis für derartige Planspiele geschaffen. Es richtet sich an zwei Zielgruppen: die Modellierer, die durch das Entwickeln der Simulationsmodelle zu einem besseren Verständnis der Zusammenhänge bei der Softwareentwicklung kommen sollen, und die Spieler, die durch die Modelle und deren Einbettung in Planspiele Erfahrung im Bereich Softwaretechnik erwerben sollen.

Da SESAM neben den Konzepten und der Methodik zur Erstellung von Modellen und den darauf basierenden Planspielen auch die für deren Erstellung benötigten Werkzeuge enthält, lässt es sich grundsätzlich zur Entwicklung eigener neuer Planspiele einsetzen.

Der Kern eines SESAM-Spiels ist das Modell, das die Spielmechanik beschreibt. Dazu wurde ein eigener Modellierungsansatz entwickelt, den Schneider in [Ludewig, 1994] als *Effektorientierte Modellierung* bezeichnet.

SESAM-Modelle bestehen aus drei Teilen:

- Durch die Definition eines **Schemas** legt der Modellierer zuerst die Domäne des Modells fest. Ein Schema beinhaltet dazu die Entitätsklassen sowie deren Attribute und potentielle Relationen, die später im Modell vorkommen können. Ein Schema grenzt so den modellierten Ausschnitt ab und dient als Basis für die dynamischen Modelle.

- Auf Basis des Schemas werden sogenannte **Situationsmodelle** als konkrete Ausprägung des Schemas definiert. Dazu werden die Entitätsklassen als Objekte instanziiert und deren Attributbelegung festgelegt. Ein Situationsmodell bildet so eine konkrete Projektsituation ab.

- Durch das Hinzufügen von **Regeln** werden die dynamischen Aspekte modelliert. Regeln können entweder über den ganzen Simulationszeitraum oder nur zu einem festgelegten Zeitpunkt wirken. Die Anwendung der Regeln wird gegebenenfalls durch entsprechende Vorbedingungen eingeschränkt.

Ein Situationsmodell wird in SESAM als Graph interpretiert, wobei die Objekte die Knoten, und die Relationen zwischen den Objekten die Kanten des Graphen bilden. Während der Simulation wird dieser Graph durch die Ausführung der Regeln verändert. Dabei können Regeln automatisch angewendet werden, wenn deren Vorbedingungen erfüllt sind, oder durch Aktionen des Spielers explizit aktiviert werden.

Die Ausführung der Regeln basiert auf dem Ansatz der Graphersetzungssyteme. Eine Regel wird dabei als Ersetzungsregel beschrieben. Sie besteht aus einem Mustergraph, der vor Anwendung der Regel gelten muss, und einem Ersetzungsgraph gegen den der Mustergraph bei Anwendung der Regel ausgetauscht wird.

SESAM-Modelle, bestehend aus Schema, Situationsmodell und Regelsatz, können von einem eigenen Simulator ausgeführt werden, um den Spielverlauf eines Planspiels zu erzeugen.

SESAM liefert ein Konzept und Werkzeuge um Planspiele zu entwickeln. Damit stimmen die Ziele des SESAM-Projekts mit denen dieser Arbeit in vielen Punkten überein. Bei der Entwicklung von SESAM sind die Autoren zu der Erkenntnis gekommen, dass sich nicht ohne weiteres festlegen lässt, welche Aspekte eines Softwareprojekts in ein entsprechendes Planspiel aufgenommen werden sollen oder müssen [Schneider, 1994]. Sie haben sich deswegen vor allem auf die konzeptionelle Basis, den Simulator, und die Werkzeuge konzentriert, die für die Erstellung eines Planspiels nützlich sind. Insbesondere sind ein Modellierungsansatz und eine dazu passende Modellierungssprache entstanden, mit der Modelle für Planspiele entwickelt werden können.

Die konkreten Modelle, scheinen bei der Entwicklung von SESAM eher im Hintergrund gestanden zu haben. So sind in der Literatur zwar einige kleine Beispielmodelle beschrieben, an denen die Konzepte von SESAM getestet und evaluiert wurden, mit dem QA-Modell [Drappa, 2000a] lässt sich jedoch nur ein einziges Modell finden, das in einem veröffentlichten Planspiel verwendet wurde. SESAM-Modelle verfolgen im Wesentlichen den Ansatz, der auch für die in Abschnitt 5.2.2 beschriebenen System-Dynamics-Modelle verwendet wird, das zu modellieren, was beobachtet werden kann. Dadurch beschränken sie sich auf eben diese Beobachtungen, die in Modellannahmen vor Beginn der Modellierung festgehalten, und anschließend im Modell umgesetzt werden. Dieser Weg unterscheidet sich von dem für das Simulationsmodell dieser Arbeit gewählten, bei dem nicht die beobachteten Auswirkungen, sondern deren Ursachen modelliert werden.

Individuelle Mitarbeiter sind in SESAM nur schwer umsetzbar. Mitarbeiter verhalten sich dort streng nach den für sie vorgegebenen Regeln, ein psychologisch fundierter Handlungsansatz, wie er im Simulationsmodell dieser Arbeit realisiert wurde, ist in SESAM kaum umsetzbar.

Mit SESAM kann zwar eine Art Bausteinbibliothek aufgebaut werden, es existiert jedoch kein integriertes Gesamtmodell, das ohne großen Aufwand für konkrete Planspiele angepasst werden kann. Damit ist es unabdingbar, sich intensiv mit der Modellierung der entsprechenden Zusammenhänge, und deren Validierung auseinanderzusetzen, um ein neues Planspiel zu entwickeln.

Obwohl sich das SESAM-Projekt in einigen Punkten von dem hier verfolgten Ansatz unterscheidet, waren seine Ergebnisse eine ergiebige Quelle an Ansätzen, Konzepten, Erfahrungen und Ideen, die zur Entstehung dieser Arbeit beigetragen hat.

5.3.2. SimSE

Ein weiteres Projekt, das über die Erstellung eines einzelnen Spiels hinausgeht, ist *SimSE* [Navarro, 2006], und wurde bis 2006 an der University of California, Irvine entwickelt (vgl. Abschnitt 1.2.1). Obwohl die Arbeit, in der SimSE entstanden ist, den Fokus auf die Untersuchung der didaktischen Wirksamkeit von Planspielen im Softwaretechnikbereich legt, wurde auch eine umfangreiche Simulations- und Spielumgebung geschaffen. Diese wurde dazu verwendet, die angestrebten Untersuchungen zu realisieren.

SimSE enthält umfangreiche Editoren zur Erstellung eigener Inhalte und Modelle. Mittels eines Generators lassen sich diese anschließend in ein Spiel mit interaktiver grafischer Benutzerschnittstelle transformieren. Während der Entwicklung von SimSE sind sechs Beispielszenarien zu unterschiedlichen Aspekten der Softwareentwicklung entstanden; unter anderem zum Wasserfallmodell, Extrem-Programming, Rational Unified Process und Rapid Prototyping.

Modelle setzen sich in SimSE aus fünf Aspekten zusammen, die sich mittels des mitgelieferten Editors definieren, konfigurieren und verwalten lassen:

- Jede Entität des Spiels basiert auf einem **Objekttyp**. Der Modellierer definiert diese Objekttypen, indem er sie von einem der sechs möglichen Metatypen *Artefakt*, *Werkzeug*, *Mitarbeiter*, *Aktivität*, *Kunde* und *Simulationsregel* ableitet. Diese Metatypen geben die Anzeigekonzepte und Interaktionsmöglichkeiten für den Spieler vor. Ein Objekttyp definiert außerdem eine Menge getypter Attribute.

- Der **Startzustand** besteht aus einer Menge von Objekten der zuvor definierten Objekttypen mit einer konkreten Belegung der einzelnen Attribute. Er stellt die initiale Konfiguration des Modells zu Spielbeginn dar und entspricht somit in etwa einem Szenario im Simulationsmodell.

- Die Menge der **Aktionen** definiert die Aktivitäten, die den Zustand der Simulation verändern können. Für jede Aktion wird festgelegt, welche Objekte teilnehmen, und was der Auslöser für die Aktion ist. Aktionen können dabei entweder durch den Zustand des Systems, den Spieler oder durch Zufall ausgelöst werden. Jede Aktion verfügt über Vorbedingungen, die zum Auslösezeitpunkt erfüllt sein müssen. Wie lange eine Aktion ausgeführt wird, wird über einen sogenannten *Destroyer* festgelegt, der analog zum Auslöser definiert wird. Um die Ausführungsreihenfolge der Aktionen untereinander festzulegen, verfügt jede Aktion über eine Priorität. Für jede Aktion kann außerdem deren Sichtbarkeit für den Spieler festgelegt werden.

- **Regeln** können an Aktionen gebunden werden und definieren die Auswirkung der Aktion auf den Simulationszustand. Sie können dabei entweder zum Zeitpunkt des Aktionsbeginns, des Aktionsendes oder kontinuierlich über den Zeitraum, in der die Aktion aktiv ist, ausgeführt werden. Regeln können neue Objekte erstellen, bestehende Objekte zerstören, die Werte der Objektattribute verändern oder anderer Aktionen auslösen.

- Mit der **Grafischer Repräsentation** wird festgelegt, wie dem Spieler der Zustand und die Veränderung des Projekts dargestellt wird. Dazu bietet SimSE verschiedene Anzeigekonzepte wie Bilder, Sprechblasen, Animationen, etc. an.

Bei der Entwicklung von SimSE wurde der Fokus auf die Werkzeuge gelegt, mit denen Planspiele erstellt und generiert werden können. Der Entwickler kann das gesamte Planspiel mit entsprechenden Editoren erstellen und daraus per Knopfdruck ein ausführbares Planspiel generieren. Obwohl im Rahmen der Entwicklung von SimSE mehrere Planspiele mit entsprechenden Simulationsmodellen im Hintergrund entstanden sind, ist in den dazugehörigen Arbeiten nur sehr wenig zu den Modellen selbst zu finden. Deren Beschreibung beschränkt sich auf wenige Annahmen, die für das jeweilige Modell getroffen wurden, zur Umsetzung und Validierung finden sich keinerlei Angaben. Auch aus der Implementierung der Modelle selbst lassen sich nur schwer Erkenntnisse gewinnen, da diese nur als unkommentierte Menge von Tabellen, Zahlen und Formeln vorliegen. Eine Wiederverwendung der Modelle oder deren Weiterentwicklung ist damit nur schwer möglich. Sie waren daher für diese Arbeit nicht verwendbar.

5.4. Weiterführende Schritte

Diese Arbeit liefert ein vollständiges, funktionierendes Simulationsmodell, das direkt eingesetzt werden kann, um Planspiele im Bereich der Softwareentwicklung zu erstellen. Trotzdem gibt es noch einige Punkte, die durch diese Arbeit nicht abgedeckt werden konnten.

Insbesondere war es aufgrund des beschränkten zeitlichen Rahmens nicht möglich, vollständige Planspiele mit dem Modell umzusetzen, die über das in Abschnitt 4.3 beschriebene Fallbeispiel hinausgehen. Ein nächster wichtiger Schritt im Anschluss an diese Arbeit ist deshalb die Umsetzung entsprechender Planspiel mit dem Modell, und deren Evaluation.

Das Simulationsmodell ermöglicht eine sehr detailreiche Simulation einiger Aspekte des Softwareentwicklungsprozesses. Insbesondere die Individualität der Projektbeteiligten und deren psychologisch fundiertes Verhalten, was in den meisten existierenden Planspielen fehlt, werden unterstützt. Diese Arbeit konnte nicht klären, welchen Einfluss diese Details auf den Lernerfolg der Spieler haben. Entsprechend wäre die Evaluation des didaktischen Nutzens von Planspielen, die mit diesem Modell gebaut werden, ein spannendes Thema für weitere Arbeiten.

5.4.1. Verbesserungen an Framework und Simulationsmodell

Um das Simulationsmodell zu erweitern und seine Anwendung zu vereinfachen, gibt es noch einige Ansätze, die im beschränkten Rahmen dieser Arbeit nicht mehr umgesetzt werden konnten. Dazu gehören sowohl die Verfeinerungen des Modells an sich, als auch die Entwicklung neuer, sowie die Verbesserung bestehender Werkzeuge. Es sind auch einige Ideen und Konzepte entstanden, die nicht zeitlich mehr umgesetzt werden konnten, teilweise den Umfang einer eigenen Arbeit haben oder nicht in den anfänglich abgesteckten Rahmen passen. Dieser Abschnitt widmet sich all diesen Punkten.

Verfeinerung des Simulationsmodells

Bei der Anwendung und Evaluation des Modells sind einigen Punkte aufgefallen, die ergänzt oder verbessert werden können, um das Modell noch einfacher, flexibler und umfangreicher einsetzen zu können:

- Das Datenmodell erlaubt die Definition von Sympathiebeziehungen zwischen Mitarbeitern, und zwischen Mitarbeitern und Aktivitäten. Es existiert jedoch keine Möglichkeit eine **Sympathiebeziehung zwischen Mitarbeitern und Artefakten** anzugeben. Dadurch ist es nicht möglich festzulegen, dass ein Mitarbeiter bestimmte Artefakte bevorzugt. Durch eine entsprechende Ergänzung in Datenmodell und Simulationsmodell wäre es möglich, Szenarien aufzubauen, in denen die Mitarbeiter bestimmte Produktteile bevorzugt bearbeiten.

- Aktivitäten sind bei ihrer Ausführung derzeit auf ein einziges Artefakt beschränkt. In manchen Fällen wäre es hilfreich, wenn **Aktivitäten auf mehrere Artefakte gleichzeitig angewendet** werden könnten. Dadurch wäre es beispielsweise möglich, Code und Codedokumentation als getrennte Artefakte zu modellieren, und trotzdem gemeinsam zu erstellen. Dazu muss ein entsprechendes Konzept erarbeitet werden, wie die in die Aktivität investierte Zeit auf die einzelnen Artefakte aufgeteilt werden muss. Ein alternatives Konzept wäre es, zwei Aktivitäten, die jeweils auf ein Artefakt angewendet werden, durch die Rahmenanwendung in einer gemeinsamen Aufgabe zusammenzufassen.

- Eine **detailliertere Parametrisierung von Artefakten** würde es dem Spielentwickler an einigen Stellen erlauben, das Szenario genauer zu kontrollieren. So könnte beispielsweise die Einarbeitungszeit in die einzelnen Artefakte unterschiedlich ausgelegt, und so beispielsweise zwischen Dokumentation und Code unterschieden werden.

- **Erweiterung des Einflusses von Sympathie-/Antipathie.** Das VIE-Modell zur Handlungsplanung ermöglicht es Mitarbeitern andere Mitarbeiter um Hilfe bei einer Aufgabe zu bitten. Dabei wird jedoch die Sympathie-/Antipathiebeziehung zwischen den Mitarbeitern nicht berücksichtigt. Um die Handlungsplanung an dieser Stelle realistischer zu machen, sollte die Entscheidung, ob und wie lange einem Kollegen bei seiner Aufgabe geholfen wird, auch von dieser Beziehung abhängig gemacht werden. Weiterhin sollte eine Feedbackschleife entwickelt werden, welche die Sympathie-/Antipathiebeziehung auf Grundlage der gewährten oder abgelehnten Hilfe verändert.

Es ist zu erwarten, dass sich bei einem umfassenderen Einsatz von Modell und Framework noch weitere Punkte ergeben, die zu einer Verbesserung des Simulationsmodells umgesetzt werden können.

Variable Schrittweite bei der Simulationsberechnung

Der Simulator, mit dem das Simulationsmodell animiert wird, verwendet eine einheitliche Schrittweite für die Berechnung aller Zeitschritte des Projektverlaufs. Diese lässt sich zwar frei wählen, gilt dann jedoch für den gesamten Simulationsverlauf. Eine adaptive Wahl der Schrittlänge, bezogen auf ein konkretes Zeitintervall im Simulationsverlauf, findet nicht statt. Besonders in den Intervallen, in denen die Mitarbeiter nicht am Arbeitsplatz sind, beispielsweise nachts und am Wochenende, ist keine so hohe Auflösung der Berechnung notwendig, wie es während der Arbeitszeit der Fall ist. Die Entwicklung eines Verfahrens zur dynamische Anpassung der Zeitschrittlänge, die sich am tatsächlichen Bedarf orientiert, stellt einen Ansatz dar, die Performanz der Simulation zu erhöhen.

Ein erster Schritt dazu ist die Auswertung von Arbeitszeiten und Projektplan, um Zeitintervalle zu identifizieren, in denen eine solche Optimierung möglich ist, ohne das Simulationsergebnis zu beeinflussen.

Verhaltensmodelle für Kunden

Die Verhaltensmodelle beschränken sich derzeit auf die Entwickler in einem Projekt. Das Verhalten des Kunden wird dadurch nur unzureichend abgedeckt. Der Kunde ist in den meisten Fällen nicht Teil des Entwicklungsteams. Für ihn gelten andere Regeln als für die Entwickler, was das Simulationsmodell derzeit nur unzureichend realisiert. Beispielsweise kann der Projektleiter über den Kunden nicht so frei verfügen, wie über seine Teammitglieder. Vor allem die Kommunikation mit dem Kunden wird durch diese Tatsache beeinflusst, da der Kunde nicht immer für ein Gespräch zur Verfügung steht. So ist beispielsweise eine spontane Kommunikation mit dem Kunden typischerweise nicht möglich.

Der Kunde kann derzeit zwar als Projektbeteiligter im Szenario angelegt, und über das Rollenkonzept von entsprechenden Projektaktivitäten ausgeschlossen werden, er hat dadurch jedoch die meiste Zeit nichts zu tun, was wiederum entsprechende Auswirkungen, vor allem auf Motivation und Gesundheit hat. Die Annahme, dass der Kunde in der Zeit, in der er nicht am Projekt mitwirkt, keine anderen Aufgaben zu erledigen hat, ist jedoch falsch. Vielmehr ist er auch noch mit anderen Aufgaben beschäftigt, die jedoch durch das Modell nicht berücksichtigt werden. Dadurch entsteht an dieser Stelle ein fehlerhaftes Verhalten der Simulation.

Eine spezielle Entwicklung von Kundenagenten könnte an dieser Stelle Abhilfe schaffen. Dabei muss zum einen sichergestellt werden, dass der Kunde eine alternative Aktivität ausführt, solange er nicht am betrachteten Projekt arbeitet. Außerdem muss ein entsprechendes Verhalten umgesetzt werden, das verhindert, dass der Kunde jederzeit abrufbar ist. Dazu kann beispielsweise die von ihm für ein Projekt zur Verfügung gestellte Zeit limitiert werden. Der Spieler muss sich dann Gedanken darüber machen, wie er diese limitierte Zeit am effektivsten einsetzt. Insbesondere muss der Kunde von der spontanen Kommunikation innerhalb eines Projekts ausgeschlossen werden.

Werden mehrere Projekte gleichzeitig simuliert, weil beispielsweise ein Mehrspielerszenario umgesetzt wird, bei dem unterschiedliche Spieler mit ihren Teams Teile eines gemeinsamen Produkts für einen gemeinsamen Kunden realisieren, muss die verfügbare Zeit des Kunden zwischen den Projekten aufgeteilt werden. Ein solches Szenario könnte den Spielern spannende neue Perspektiven auf das Projektmanagement ermöglichen.

Arbeitsumgebung

Wie in Abschnitt 1.4 festgelegt, berücksichtigt diese Arbeit keine Aspekte der Arbeitsumgebung. Dazu gehören unter anderem die Gestaltung des Arbeitsplatzes, firmenpolitische Entscheidungen, Leitbilder und die Arbeitsbedingungen im Allgemeinen. Eine Erweiterung des Simulationsmodells um solche Aspekte würde eine Vielzahl an neuen Effekten ermöglichen. In Anhang A.5 werden einige solche Effekte beschrieben.

Ein möglicher Ansatz dazu ist die Einführung von zusätzlichen Szenarioparametern, die entsprechende Aspekte beschreiben. Auch das Parameterset der Mitarbeiter kann entsprechend erweitert werden, um mitarbeiterspezifische Eigenschaften der Arbeitsumgebung zu ermöglichen. Für den Einfluss der Parameter auf den Projektverlauf müssen neue Regeln und Teilmodelle entwickelt, und in das bestehende Simulationsmodell integriert werden. Schnittstellen liegen hierbei vermutlich vor

allem beim Motivationsmodell und den Verhaltensmodellen, aber auch die Arbeitsleistung an sich kann beispielsweise durch die Gestaltung des Arbeitsplatzes beeinflusst werden.

Um die Arbeitsumgebung in ein Planspiel zu integrieren, können entsprechende Parameter der Arbeitsumgebung beispielsweise durch Ereignisse verändert werden. Gegebenenfalls könne ausgewählte Parameter auch durch den Spieler selbst bestimmt werden. Einige davon müssen dazu eventuell an das zur Verfügung stehende Budget gekoppelt werden.

Arbeitsmarkt

Das Simulationsmodell unterstützt das Hinzufügen und Entfernen von Mitarbeitern eines Projekts während des Simulationsverlaufs. Werden Mitarbeiter hinzugefügt, müssen diese dazu zuerst erstellt werden. Dazu können Mitarbeiter, die dem Spieler zur Verfügung stehen, entweder im Szenario vordefiniert oder bei Bedarf generiert werden.

Eine umfassendere Möglichkeit ist die Umsetzung eines Arbeitsmarkts, der einen Pool von Mitarbeitern zur Verfügung stellt. Dabei werden dem Pool regelmäßig neue Mitarbeiter hinzugefügt und bestehende daraus entfernt. Dieses Konzept ermöglicht einige neue interessant Aspekte. Zum einen kann der Arbeitsmark gemeinsam von mehreren Spielern genutzt werden. Stellt ein Spieler einen Mitarbeiter ein, steht dieser den anderen Spielern nicht mehr zur Verfügung. Umgekehrt können Mitarbeiter, die kündigen oder von einem Spieler entlassen werden, von anderen Spieler eingestellt werden. Mitarbeiter können so beispielsweise die in einem Projekt erlernten Fähigkeiten in ein anderes Projekt mitnehmen.

Eine weitere interessante Möglichkeit ist es, den Arbeitsmarkt auf einem prototypischen Mitarbeiter aufzusetzen. Dieser Mitarbeiter repräsentiert den durchschnittlichen Mitarbeiter auf dem Arbeitsmarkt. Alle generierten Mitarbeiter sind Variation von ihm. Dazu werden die Attribute generierter Mitarbeiter beispielsweise zufällig mittels der Normalverteilung gewählt, wobei der Attributwert aus dem Prototyp deren Erwartungswert vorgibt. Mittels des prototypischen Mitarbeiters kann so der Arbeitsmarkt gesteuert werden. Seine Attribute können wiederum durch Ereignisse im Simulationsverlauf verändert werden, um beispielsweise einen Engpass an Softwarearchitekten zu erzeugen.

Wird ein solcher Arbeitsmarkt für ein sehr großes Planspiel mit vielen Spielern und Projekten verwendet, besteht zudem die Möglichkeit, den prototypischen Mitarbeiter, und damit den Arbeitsmarkt, auf Basis der von den Spielern eingestellten und entlassenen Mitarbeitern zu verändern. Benötigen beispielsweise in einer bestimmten Phase alle Spieler Mitarbeiter mit hohen Implementierungsfähigkeiten, stellt der Arbeitsmarkt weniger davon zur Verfügung.

Risikomanagement

Ein weiterer Aspekt, der im Simulationsmodell derzeit nicht umgesetzt wird, ist das Risikomanagement. Die Literatur zu diesem Thema enthält einige Modelle, die beispielsweise eine Einschätzung von Projekt- und Produktrisiken ermöglichen. Diese Modelle könnten als Basis einer Erweiterung des Simulationsmodelles dienen, um unerwartete Ereignisse, wie beispielsweise Ausfälle von Mitarbeitern oder Hardware zu generieren. Der Spieler kann durch eine entsprechende Risikoanalyse passende Gegenmaßnahmen planen, vorbereiten und bei Bedarf umsetzen.

Grafischer Editor für Szenarien

Derzeit werden Szenarien vor allem mittels Algorithmen im Programmcode oder über entsprechende XML-Dateien mittels Texteditoren erstellt. In Abschnitt 2.3.3 wird eine grafische Notation für die Struktur der im Simulationsmodell verwendeten Szenarien eingeführt. Diese Notation wurde für die Darstellungen in dieser Arbeit entwickelt. Sie hat jedoch auch das Potential als Basis eines grafischen Editors für Szenarien zu dienen. Ein solcher Editor könnte die Entwicklung von komplexen und umfangreichen Szenarien verbessern.

Grafische Oberfläche für das Evaluationsframework

Das Evaluationsframework ist derzeit eine Sammlung von Algorithmen, Skripten und Kommandozeilenwerkzeugen. Um die Benutzbarkeit zu erhöhen, ist es notwendig, diese in eine gemeinsame Oberfläche zu integrieren, die dem Nutzer eine bessere Übersicht über die Experimente, deren Ausführungszustand und die Ergebnisse bietet, und eine komfortable Navigation durch die entstandenen Daten ermöglicht.

Besonders die zwei folgenden Funktionen wären bei der Durchführung der Evaluation besonders hilfreich gewesen:

Das Simulationsframework berechnet derzeit zuerst alle Simulationsverläufe und erzeugt daraus anschließende die entsprechenden Korrelationskoeffizienten, Plots und weiteren Ergebnisse. Da die Ausführung der Simulationen bei einer großen Anzahl an Durchläufen oder komplexen Szenarien viel Zeit in Anspruch nimmt, wäre ein aktueller Stand nach der Durchführung jedes Simulationslaufs hilfreich, um frühzeitig Probleme und Fehler zu erkennen, die ein Fortsetzen des Experiments unnötig machen. Dazu könnte eine entsprechende Oberfläche eine Echtzeitanzeige der Ergebnisse des aktuell durchgeführten Experiments enthalten.

Liefert ein Experiment auffällige Datenpunkte oder interessante Ausreißer, so müssen deren Simulationsverläufe im Einzelnen untersucht werden. Dazu muss zum entsprechenden Datenpunkt der passende Durchlauf herausgefunden werden, was derzeit sehr aufwändig ist. Eine entsprechende Oberfläche könnte dies stark vereinfachen, in dem sie die Funktionalität bietet, das Szenario zu einem ausgewählten Datenpunkt direkt zu laden.

Des Weiteren können in eine solche Oberfläche nützliche Analysewerkzeuge, vor allem zur Unterstützung statistischer Techniken, wie beispielsweise zur Berechnung von Korrelationen und der Signifikanz von Unterschieden mehrere Testgruppen, sowie Werkzeuge zur weiteren Visualisierung der Experimente und ihrer Ergebnisse integriert werden.

Selbsterklärende Simulation

Um den Simulationsverlauf im Detail nachvollziehen zu können, ist es derzeit notwendig, sich entsprechend in das Simulationsmodell einzuarbeiten. Dem Spieler eines Planspiels ist dies nicht zumutbar. Eine wertvolle Ergänzung zur Berechnung des Simulationsverlaufs wäre die Extraktion besonderer Konstellationen, Ereignisse, Symptome und Auffälligkeiten, die dem Spieler gesondert aufgezeigt werden können. Gelingt es, zu diesen Auffälligkeiten auch die passende Erklärung zu liefern, kann dadurch der Lerneffekt erheblich verbessert werden. So könnte dem Spieler beispielsweise die folgende Meldung angezeigt werden:

„Die Produktivität von Mitarbeiter "Maier" ist im letzten Monat um 30% eingebrochen, weil Mitarbeiter "Schulz" ihn die ganze Zeit von seiner Arbeit abhält. Mitarbeiter "Schulz" tut dies, weil ihm die Aufgabe "Testen der Datenbankschnittstelle" zugeteilt wurde, welche ihn überfordert. Er holt sich deswegen Hilfe bei Mitarbeiter "Maier".“

Eine Extraktion von Auffälligkeiten und das Erstellen einer verständlichen Erklärung aus dem reinen Simulationsverlauf ist eine schwierige Aufgabe, die in einer eigenen Arbeit behandelt werden sollte. Es muss ein Weg gefunden werden, wie die einzelnen Einflüsse, die zu den gefundenen Auffälligkeiten geführt haben, bis zur Aktion des Spielers, den Bedingungen des Szenarios oder entsprechenden Ereignissen zurückverfolgt werden können. Ansätze hierfür könnten Arbeiten aus dem Gebiet der selbsterklärenden Systeme liefern [Fähndrich, 2013]. Vor allem bei verteilten Systemen die auf einem Multi-Agenten-Ansatz basieren, lassen sich Parallelen zum in dieser Arbeit beschriebenen Simulationsmodell vermuten.

5.4.2. Erweiterung der Evaluation

Das Simulationsmodell wurde im Rahmen dieser Arbeit ausführlich evaluiert. Dabei wurden vor allem die in der Literatur aufgeführten Zusammenhänge und Effekte herangezogen. Eine weiterführende Evaluation würde helfen, vor allem die Parameter des Simulationsmodells noch weiter zu optimieren, und es damit vor allem quantitativ noch besser mit der Realität abzugleichen.

Vergleich mit realen Projektverläufen

Eine mögliche Vorgehensweise hierzu kann die Auswertung von Projektverläufen realer Projekte sein. Dass eine solche Evaluation schwierig und in manchen Punkten vielleicht sogar überhaupt nicht möglich ist, da entsprechende Eigenschaften nicht direkt gemessen werden können, wurde bereits in Abschnitt 3.3 besprochen. Ein erster Ansatz für eine solche Evaluation wäre es, das Simulationsmodell begleitend zu einem realen Projekt einzusetzen, indem das dort gefertigte Produkt, die entwickelnden Mitarbeiter und die verwendeten Aktivitäten in einem entsprechenden Szenario definiert werden. Der im Projekt verwendete Projektplan muss ebenfalls in das Szenario übertragen werden. Der reale und der simulierte Projektverlauf können anschließend verglichen werden, um Diskrepanzen zu identifizieren und das Simulationsmodell dahingehend anzupassen und zu verbessern.

Vergleich mit Modellen bestehender Planspiele

In einer eigenständigen derzeit an der Universität Ulm durchgeführte Arbeit, wird ein Benchmark zum Vergleich der Simulationsmodelle unterschiedlicher Planspiele entwickelt. Die Idee dabei ist, die jeweiligen Modelle von existierenden Planspielen anhand verschiedener Szenarien zu prüfen und zu bewertet. Ziel dabei ist es, einen Überblick zu bekommen, welche Mächtigkeit die einzelnen Modelle haben, welche Situationen damit erzeugt werden können und welche Effekte sie umsetzen. Auch das in dieser Arbeit entstandene Simulationsmodell nimmt an dieser Untersuchung teil. Die Ergebnisse der Untersuchung stehen derzeit noch aus.

5.4.3. Weitere Einsatzfelder für das Simulationsmodell

Das Simulationsmodell wurde geschaffen, um die Entwicklung neuer Planspiele im Bereich der Softwareentwicklung und des damit verbundenen Projektmanagements zu unterstützen. Alle Aspekte von Modell und Framework sind darauf hin ausgerichtet. Das gilt auch für die Evaluation des Modells. Trotzdem stellt sich die Frage, ob das so entstandene Simulationsmodell nicht auch für andere Zwecke eingesetzt werden kann.

Planspiele außerhalb der Softwareentwicklung

Einige Erkenntnisse, die in die Entwicklung des Simulationsmodells eingeflossen sind, beziehen sich nicht nur auf Softwareprojekte, sondern sind allgemein für die Projektarbeit gültig. Andere Erkenntnisse, die aus der Softwareentwicklungsdomäne stammen, lassen sich auf andere Projekte übertragen. Es lässt sich daher die Hypothese aufstellen, dass mit dem Simulationsmodell auch Projektverläufe anderer Projekttypen berechnet werden können, um damit Planspiele für andere Domänen zu unterstützen. Insbesondere im Bereich der Betriebswirtschaftslehre sind Planspiele sehr beliebt [Faria, 1998].

Eine gewisse Ähnlichkeit kann unter anderem zwischen Softwareprojekten und anderen ingenieurmäßig durchgeführten Projekten festgestellt werden. Dabei handelt es sich beispielsweise um Projekte im Bereich des Maschinenbaus oder der Gebäudearchitektur. Auch in diesen Bereichen werden Planspiele in der Lehre eingesetzt [Bonta, 1979]. Ein Problem bei der Anwendung des Simulationsmodells aus dieser Arbeit könnte die Erstellung von Hardwareartefakten sein, beispielsweise der Fertigung eines Karosserieprototyps oder dem Gießen eines Gebäudefundaments. Das Simulationsmodell setzt für alle Artefakte voraus, dass es sich um Dokumente handelt, die Wissen enthalten, für deren Erstellung jedoch keine handwerklichen Fähigkeiten notwendig sind. Es berücksichtigt auch nicht, dass die Erstellung bestimmter Artefakte, wie beispielsweise die Aushärtung eines Fundaments nicht nur von der investierten Arbeitszeit abhängt.

Um das Simulationsmodell auch für andere Projekte einsetzen zu können, muss daher in einem ersten Schritt ermittelt werden, inwieweit die in dieser Arbeit getroffenen Annahmen auch für andere Projekte gültig sind, und welche zusätzlichen Aspekte benötigt werden. In einem zweiten Schritt muss festgestellt werden, ob, und mit welchem Aufwand das Simulationsmodell um die fehlenden Aspekte ergänzt werden kann.

Analyse von realen Situationen

An mehreren Stellen dieser Arbeit wurde darauf hingewiesen, dass sich das Simulationsmodell nicht dazu eignet, als Vorhersagemodell zu fungieren, da es nur einen der vielen möglichen Projektverläufe berechnet. Hier stellt sich die Frage, ob dieser Umstand ausgenutzt werden kann, um eine reale Situation zu analysieren.

Lässt sich die zu betrachtende Situation in einem Szenario modellieren, so können mit dem Simulationsmodell zufällige Projektverläufe berechnet und analysiert werden. Ähnlich wie das bei der Evaluation des Simulationsmodells angewandte Vorgehen, können dabei unbekannte Parameter zufällig belegt werden. Prozesse und Managementstrategien können durch zusätzliche Agenten realisiert werden. Bei der Auswertung der so berechneten Projektverläufe können nun diejenigen bevorzugt betrachtet werden, die ein unerwartetes Ergebnis liefern, oder die sich von den anderen unterscheiden. So können Hinweise auf mögliche Probleme und Risiken im Projekt gesammelt werden, an die zuvor nicht gedacht wurde. Ob dieses Vorgehen in der Praxis funktioniert muss jedoch noch untersucht werden.

Generierung neuer Erkenntnisse im Bereich der Softwareentwicklung

Wie alle Modelle, so hat auch das Simulationsmodell das Potential unbekannte Zusammenhänge aufzuzeigen, noch nicht entdeckte Effekte offenzulegen, und so zu neuen Erkenntnissen zu führen. Da sich der gewählte Ansatz von dem der meisten in diesem Bereich existierenden Modellen unterscheidet, kann es vielleicht verwendet werden, um Theorien aus anderen Arbeiten zu überprüfen. Ein erster Schritt dazu wurde mit der Evaluation des Simulationsmodells schon getan. Dabei haben sich keine erwähnenswerten Diskrepanzen zu anderen Modellen und Theorien gezeigt. Das weist darauf hin, dass die bestehenden Theorien mit der Umsetzung des Simulationsmodells konsistent sind.

Mit jedem weiteren Aspekt, der in das Simulationsmodell eingefügt, und jeder Theorie, die mit dem Simulationsmodell überprüft wird, besteht die Möglichkeit eine Unstimmigkeit aufzudecken. Diese weist auf einen Fehler in der Theorie, im Simulationsmodell oder beidem hin, und kann dazu verwendet werden, die jeweiligen Teile zu korrigieren, zu verbessern und dabei zu einem entsprechenden Erkenntnisgewinn zu führen.

A Anhang

A.1 Übersicht über die Attribute der Szenarioelemente

A.2 Übersicht über die verwendeten mathematischen Symbole

A.3 Definition der Zustände des Motivationsmodells

A.4 Liste der bei der Evaluation verwendeten Parameter

A.5 Navarros 86 fundamentale Regeln des Software Engineering

A.1. Übersicht über die Attribute der Szenarioelemente

In diesem Abschnitt sind alle Eigenschaften, der zum Szenario gehörenden Entitäten des Simulationsmodells aufgeführt. Neben der Bezeichnung und einer kurzen Beschreibung sind, falls in dieser Arbeit verwendet, auch die dazugehörigen mathematischen Symbole genannt. Falls vorhanden, sind die Standardwerte in eckigen Klammern angegeben. Eine ausführlichere Beschreibung der einzelnen Eigenschaften findet sich in Abschnitt 2.3 und 2.4.

Artefakt

Artefakttyp		Über den *Artefakttyp* wird das Artefakt einer bestimmten Kategorie, wie beispielsweise Anforderungsdokument, Codefragment oder Komponententest, zugeordnet.
Defekte	$\Lambda = \{\lambda_1, \ldots, \lambda_n\}$	Die Menge der *Defekte* beschreibt die Fehler, die während der Bearbeitung des Artefakts gemacht wurden.
Disziplin	$D = \{d_1, \ldots, d_n\}$	Artefakten können für sie relevante *Disziplinen* zugeordnet werden, die spezielle Fähigkeiten definieren, die zur Bearbeitung des jeweiligen Artefakts notwendig sind. Für jede relevante Disziplin wird dabei deren Anteil am Artefakt, und die benötigten Fähigkeiten in Form einer Aufgabenschwierigkeit festgelegt.
Einfluss zusätzlichen Wissens	$h_*[= 1.0]$	Mit einem Faktor zwischen 0 und ∞ wird der *Einfluss zusätzlichen Wissens* auf die Bearbeitung des Artefakts festgelegt. Ein Faktor von 1 bedeutet, dass zusätzliches Wissen keinen Einfluss auf die Bearbeitung des Artefakts hat, Faktoren größer 1 geben an, dass ein Wissensvorsprung zu einer Steigerung der Produktivität beiträgt, während Faktoren kleiner 1 bedeuten, dass fehlendes Wissen über den zu bearbeitenden Teil die Produktivität herabsenkt.
Fortschritt	$c[= 0U]$	Der *Fortschritt* bzw. die *Vollständigkeit* gibt an, welcher Anteil des Artefakts schon erstellt wurde. Er wird in Arbeitseinheiten zwischen 0 und dem Umfang des Artefakts angegeben. Aus Fortschritt und Umfang kann der *relative Fortschritt* berechnet werden, der angibt, zu welchem Teil das Artefakt schon erstellt wurde.
Kontextartefakte		*Kontextartefakte* beschreiben diejenigen anderen Artefakte des Produkts, die einen Einfluss auf die Bearbeitung des betrachteten Artefakts haben. Im Gegensatz zu den Vorbedingungen, bilden Kontextartefakte keine Voraussetzungen für die Bearbeitung, sondern beeinflussen, abhängig von ihrem Zustand, die Arbeitsleistung bei der Erstellung des Artefakts. Dazu wird für jedes Kontextartefakt dessen Einfluss in Form eines Faktors zwischen 0 und 1 angegeben.

Qualität	$q[= 0.7]$	Die *Qualität* ist ein abstraktes Maß, das die Menge der Defekte in Abhängigkeit des Fortschritts des Artefakts in einer einzigen Zahl, wie in Formel 2.1 gezeigt, zwischen 0 und 1 zusammenfasst. Eine Qualität von 1 entspricht dabei einem defektfreien Artefakt, ein Wert von 0 einem aufgrund der hohen Defektzahl völlig unbrauchbaren Artefakt, und ein Wert von 0.7 einem durchschnittlichen Artefakt.
Schwierigkeit	$\theta[= 0.5]$	Die *Schwierigkeit* gibt an, wie schwer die Erstellung des Artefakts grundsätzlich ist. Sie bezieht sich entsprechend der Definition der Aufgabenschwierigkeit auf die benötigten Fähigkeiten.
Umfang	$s[= 100U]$	Der *Umfang* legt den Erstellungsaufwand des Artefakts in *Arbeitseinheiten* fest.
Vorbedingung		Mittels *Vorbedingungen* lassen sich die Voraussetzungen definieren, die gelten müssen, um ein Artefakt bearbeiten zu können. Vorbedingungen beziehen sich auf jeweils ein anderes Artefakt des Produkts.

Aktivität

Aktivitätsklasse		Basisklasse der die Aktivität angehört. Das Simulationsmodell unterstützt *Fortschrittsaktivität*, *Qualitätssicherungsaktivität* und *Kommunikationsaktivität*.
Defekttypen	$\Lambda_{\text{Typ}} = \{\lambda_{\text{Typ}_1}, \ldots, \lambda_{\text{Typ}_n}\}$	Mittels der *Defekttypen* wird festgelegt, welche Arten von Defekten eine Fortschrittsaktivität erzeugen kann, bzw. welche Arten von Defekte mit einer Qualitätsaktivität aufgedeckt werden können.
Kontextartefakttypen		Die Menge der *Kontextartefakttypen* schränkt die Kontextartefakte, die bei Ausführung der betrachteten Aktivität berücksichtigt werden anhand ihres Artefakttyps ein.
Quantitätsmodifikator	$[= 1.0]$	Mittels des *Quantitätsmodifikators* kann der Effekt auf das bearbeitete Artefakt skaliert werden. Bei Fortschrittsaktivitäten skaliert der Modifikator den erzielten Fortschritt, bei Qualitätsaktivitäten den Umfang der gefundenen Defekte, und bei Kommunikationsaktivitäten den Umfang des übertragenen Wissens.
Qualitätsmodifikator	$[= 1.0]$	Der *Qualitätsmodifikator* skaliert die Qualität, mit der eine Aktivität durchgeführt wird. Dadurch wird bei Fortschrittsaktivitäten der Umfang, der ins Artefakt eingebrachten Defekte beeinflusst. Bei Qualitätssicherungsaktivitäten hingegen beeinflusst der Modifikator, welche Defekte, bezogen auf ihre Schwierigkeit, aufgedeckt werden können.
Rollen		Die Projektbeteiligten, die eine bestimmte Aktivität ausführen dürfen, werden über die Menge der ausführenden *Rollen* beschränkt.
Zielartefakttypen		Die Menge der *Zielartefakttypen* gibt an, auf welche Artefakttypen die jeweilige Aktivität angewendet werden kann, um deren Zustand zu verändern.

Projektbeteiligter

Arbeitszeitkonto	$t_{\Delta\text{Konto}}[= 0h]$	Jeder Mitarbeiter verfügt über ein Zeitkonto auf dem seine Überstunden bzw. Minusstunden aufsummiert werden.
Artefaktwissen	$K[= \emptyset]$	Jeder Mitarbeiter verfügt zu jedem Artefakt über ein Datum, welches sein Wissen über dessen Inhalt und Struktur angibt. Dieses *Artefaktwissen* bezieht sich immer auf das maximale Wissen, das für das jeweilige Artefakt erworben werden kann, und wird als Menge von Teilintervallen des Intervalls $[0, 1]$ angegeben, welches das maximale Artefaktwissen für jeweils jedes Artefakt darstellt. So kann mittels des Werts nicht nur angegeben werden, wie viel, sondern auch welcher Teil des Wissens über das Artefakt dem Projektbeteiligten bekannt ist.
Energieniveau	$\mathrm{e}[= 1.0]$	Das *Energieniveau* gibt den Gesundheitszustand, die momentane Fitness, und das körperliche Wohlbefinden des Projektbeteiligten als einen einzelnen abstrakten Wert zwischen 0 und ∞ an. Dabei entspricht ein Wert von 1 dem Wert eines durchschnittlichen Mitarbeiters. Werte kleiner 1 stehen für eine unterdurchschnittliche, Werte größer 1 für eine überdurchschnittliche körperliche Verfassung.
Fähigkeiten (unmittelbar)	$l[= 0.5]$	Jeder Projektbeteiligte verfügt über eine Menge an verschiedenen *Fähigkeiten*, die sich jeweils auf eine bestimmte Aktivität oder eine bestimmte Disziplin beziehen. Das Fähigkeitsniveau, welches zwischen 0 und 1 angegeben wird, sagt aus, wie gut der Mitarbeiter sich im entsprechenden Fachbereich auskennt, die entsprechenden Aktivitäten beherrscht, und grundsätzliche Übung und Erfahrung in diesem Bereich hat.
Fähigkeiten (Langzeitgedächtnis)	$\underline{l}[= 0.5]$	Das Fähigkeitsniveau im *Langzeitgedächtnis* gibt an, bis auf welches Niveau eine bestimmte Fähigkeit durch Vergessens- und Verlernprozesse absinken kann.
Gehalt	$\omega[= 3000]$	Für jeden Projektbeteiligten kann ein monatliches *Gehalt* angegeben werden.
Motivation	$m[= 1.0]$	Der *Leistungseinfluss* durch *Motivation* gibt an, wie sich die aktuelle Motivation des Mitarbeiters auf seine Arbeitsleistung auswirkt. Der Wert wird aus dem Zustand des Motivationsmodells abgeleitet.
Rollen		Jedem Mitarbeiter können eine oder mehrere *Rollen* zugewiesen werden. Rollen sind frei definierbar. Anhand der Rollen kann eingeschränkt werden, welche Aktivitäten der Mitarbeiter durchführen kann.
Vertragliche Arbeitszeit	$t_{\Delta\text{Vertrag}}[= 8h]$	Die vertragliche Arbeitszeit legt fest, wie viele Stunden ein Mitarbeiter pro Tag arbeiten muss, um seinen Arbeitsvertrag zu erfüllen.

Charaktereigenschaften

Eigenschaften der Physis

Ausdauer	$e_{*-}[= 1.0]$	Der *Ausdauerparameter* beschreibt, wie schnell der Mitarbeiter im Vergleich zum Durchschnitt ermüdet, wenn er eine Aufgabe durchführt. Ein durchschnittlicher Mitarbeiter hat einen Wert von 1, Mitarbeiter die schneller ermüden besitzen einen Wert größer 1, Mitarbeiter mit überdurchschnittlicher Ausdauer haben Werte kleiner 1.
Tagesleistungskurve	P_T	Die *Tagesleistungskurve* beschreibt die durchschnittlichen persönlichen Leistungsschwankungen innerhalb eines 24-Stunden-Zyklus.
Zeitdruckschranke	$\hat{\rho}[= 1.0]$	Die *Zeitdruckschranke* gibt an, wie stark sich die Effizienz eines Projektbeteiligten durch Zeitdruck steigern lässt.

Eigenschaften der Kognition

Adaptionsfähigkeit	$\kappa_*[= 1.0]$	Die *Adaptionsfähigkeit* gibt an, wie schnell ein Projektbeteiligter Artefaktwissen anhand eines vorliegenden Artefakts oder durch Kommunikation erlernen kann. Der durchschnittliche Mitarbeiter hat eine Adaptionsfähigkeit von 1. Werte darunter zeigen eine unterdurchschnittliche, Werte größer 1 ein überdurchschnittliche Fähigkeit an.
Lernfähigkeit	$l_{*+}[= 1.0]$	Der *Lernfaktor* gibt an, wie groß die Lernfortschritte eines Projektbeteiligter sind, die er durch die Ausführung von Aktivitäten macht. Der Wert eines durchschnittlichen Mitarbeiters liegt bei 1. Werte darunter zeigen eine unterdurchschnittliche, Werte größer 1 eine überdurchschnittliche Lerngeschwindigkeit an.
Wechselfähigkeit	$j_*[= 1.0]$	Der Wert für die *Wechselfähigkeit* bestimmt, wie schnell ein Projektbeteiligter mental zwischen verschiedenen Aufgaben wechseln kann. Der Wert eines durchschnittlichen Mitarbeiters liegt bei 1. Werte darunter zeigen eine unterdurchschnittliche, Werte größer 1 eine überdurchschnittliche benötigte Zeit an. Bei einem Wert von 0 benötigt der Mitarbeiter keine zusätzliche Zeit für den mentalen Teil des Aufgabenwechsels.
Rednerfähigkeit	$\kappa_{*\mathrm{Redner}}[= 1.0]$	Die *Rednerfähigkeit* gibt an, wie gut ein Mitarbeiter sein eigenes Artefaktwissen in Worte fassen kann, um es im Rahmen einer Kommunikation an andere Mitarbeiter weiterzugeben. Der durchschnittliche Projektbeteiligte hat eine Fähigkeit von 1. Werte kleiner 1 stehen für eine unterdurchschnittliche, Werte größer 1 für eine überdurchschnittliche Fähigkeit. Besitzt ein Mitarbeiter eine Fähigkeit von 0, kann er sein Wissen nicht weitergeben.

Vergessensrate	$l_{*-}[= 1.0]$	Die *Vergessensrate* gibt an, wie schnell ein Mitarbeiter zuvor erworbenes Wissen, das noch nicht im Langzeitgedächtnis gespeichert ist, wieder vergisst. Sie wird relativ zu einem durchschnittlichen Mitarbeiter gewählt. Werte größer 1 geben ein überdurchschnittlich schnelles Vergessen, Werte kleiner 1 ein unterdurchschnittlich schnelles Vergessen an.

Emotionale Eigenschaften

Aktivitätssympathie	$\psi_a[= 0.5]$	Mit der *Aktivitätssympathie* wird angegeben, wie gerne ein Projektbeteiligter eine bestimmte Aktivität durchführt. Dazu wird ein Wert aus $[0, 1]$ verwendet, der die intrinsische Motivation der Aktivität angibt. Hierbei steht 0 für die maximale Abneigung, und 1 für die höchst mögliche Motivation, die Aktivität durchzuführen. Der neutrale Wert ist 0.5.
Bedürfnisse	$\vec{n}$ $[= (0.5, \ldots, 0.5)]$	Der *Bedürfnisvektor* gibt die persönliche Gewichtung der einzelnen Bedürfnisse (vgl. 2.6.6) an. Im Falle des Standardwerts, sind alle Bedürfnisse gleich gewichtet.
Anspruch an die Hygienefaktoren und Motivatoren	$g = \{g_1, \ldots, g_n\}$ $[= \{0.5, \ldots, 0.5\}]$	Mittels der *Anspruchswerte* wird die Erwartungshaltung für jeden *Hygienefaktor* bzw. *Motivator* angegeben. Der Anspruch gibt an, ab welchem Wert ein Hygienefaktor erfüllt ist, bzw. bis zu welchem Wert ein Motivator wirkt.
Mitarbeitersympathie	$\psi_{\mathrm{m}}[= 0.5]$	Die *Mitarbeitersympathie* gibt an, wie gerne ein Mitarbeiter mit einem anderen Mitarbeiter zusammenarbeitet. Die Bewertung der Beziehung muss zwischen den Mitarbeitern dabei nicht zwangsläufig symmetrisch sein. Die Sympathie wird mit einem Wert aus $[0, 1]$ angegeben, wobei 0 für die maximale Antipathie, und 1 für die maximale Sympathie steht. Der neutrale Wert ist 0.5.

A.2. Übersicht über die verwendeten mathematischen Symbole

Dieser Abschnitt enthält eine Übersicht über die in dieser Arbeit häufig verwendeten mathematischen Symbole. Für Werte werden Kleinbuchstaben verwendet, Mengen und Funktionen werden durch Großbuchstaben dargestellt. Griechische Buchstaben dienen lediglich als Ergänzung zum lateinischen Alphabet und haben keine gesonderte Bedeutung.

Maximalwerte werden als $\hat{x}$ dargestellt. Werte, die eine obere bzw. untere Schranken symbolisieren, werden als $\overline{x}$ bzw. $\underline{x}$ dargestellt. $\overline{x}$ wird auch für Durchschnittswerte verwendet.

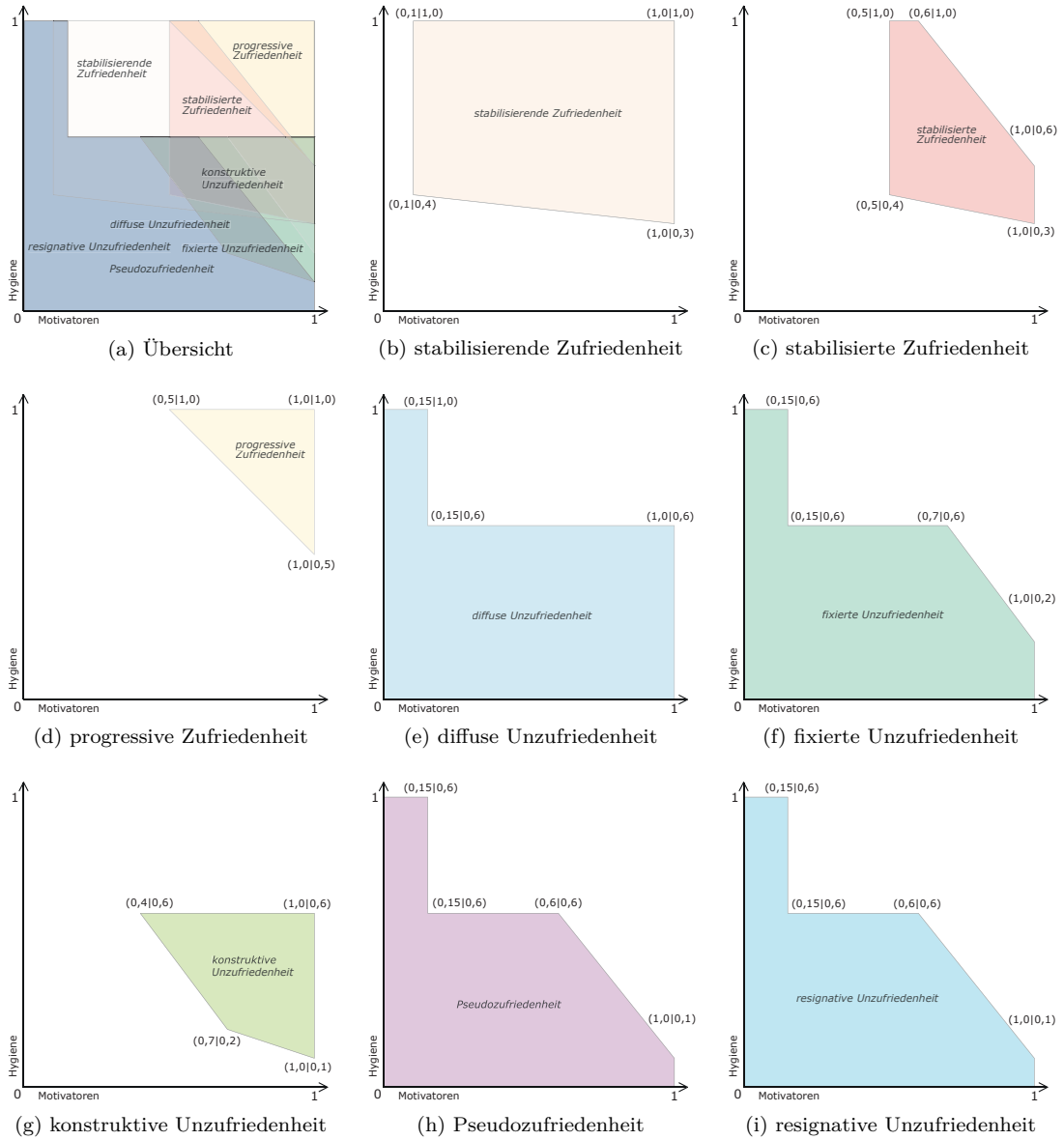
Durch tiefgestellte Symbole oder Texte werden Symbole näher spezifiziert. Differenzen werden als x_Δ, Faktoren als x_* dargestellt. Um anzuzeigen, dass ein Wert oder eine Funktion mit einer Zunahme bzw. Abnahme in Verbindung steht, wird ein tiefgestelltes $+$ bzw. $-$ verwendet.

b	Projektbeteiligter.
B	Menge von Projektbeteiligten.
c	Vollständigkeit eines Artefakts. Relativ oder absolut.
d	Disziplin.
D	Menge von Disziplinen.
e	Energieniveau.
e_{*-}	Erschöpfungsfaktor.
$E_{\Delta-}(\cdot)$	Energieverlust durch Erschöpfungsprozesse.
$E_{\Delta+}(\cdot)$	Energiegewinn durch Erholungsprozesse.
$E(\tau, \vec{r})$	Wahrscheinlichkeit, dass die Aufgabe τ zum Ergebnis $\vec{r}$ führt. Teil des VIE-Modells.
$F_a(n)$	Aktivitätsspezifische Funktion zur Berechnung des Einflusses der Gruppengröße n auf die Arbeitsleistung.
$F_k(k_\Delta)$	Aktivitätsspezifische Funktion zur Berechnung des Einflusses der Wissensabdeckung k_Δ auf die Arbeitsleistung.
g	Anspruch oder Gewicht.
G	Menge von Anspruchsfaktoren oder Gewichtsfaktoren.
θ	Schwierigkeitsniveau.
$I(\vec{r}, o_i)$	Instrumentalität im VIE-Modell. Beschreibt, wie sich das Ergebnis $\vec{r}$ auf die Folge o_i auswirkt.
j_*	Aufgabenwechselfähigkeit. Gibt an, wie schnell ein Projektbeteiligter zwischen einzelnen Tätigkeiten wechseln kann.
$\mathbb{K}$	Vollständige Wissensdomäne jedes Artefakts. Entspricht dem Intervall $[0, 1]$.
K	Artefaktwissen. Teil der vollständigen Wissensdomäne. $K \subseteq \mathbb{K}$.
k_Δ	Überdeckungsgrad zweier Artefaktwissensstände.
k_*	Einfluss von Wissen.
κ_*	Adaptionsfaktor für Wissen. Charaktereigenschaft der Mitarbeiter, die angibt, wie schnell diese Wissen aufnehmen können.
$\kappa_{*\mathrm{Redner}}$	Rednerfähigkeit. Fähigkeit Wissen in Worte zu fassen.
l	Fähigkeitsniveau.
$L_{\Delta+}(\cdot)$	Fähigkeitsgewinn durch Lernprozesse.
$L_{\Delta-}(\cdot)$	Fähigkeitsverlusts durch Vergessensprozesse.
$\underline{L}_{\Delta+}(\cdot)$	Fähigkeitsgewinns im Langzeitgedächtnis.
l_{*+}	Lernfaktor. Gibt an, wie schnell ein Mitarbeiter lernt.
l_{*-}	Vergessensfaktor. Gibt an, wie schnell ein Mitarbeiter vergisst.

$\underline{l}$	Fähigkeitsniveau im Langzeitgedächtnis.
λ	Defekt.
Λ	Menge von Defekten.
λ_{Typ}	Defekttyp.
Λ_{Typ}	Menge von Defekttypen.
m	Leistungseinfluss der Motivation.
$M(\cdot)$	Leistungseinfluss der Motivation (Funktion).
$\vec{n}$	Bedürfnisvektor im VIE-Modell.
o	Folge im VIE-Modell.
O	Menge der möglichen Folgen im VIE-Modell.
$\vec{o}$	Folgenvektor im VIE-Modell.
p	Leistungseinfluss.
$P(\cdot)$	Leistungseinfluss (Funktion).
$P_T(\cdot)$	Tagesleistungsfunktion.
ρ	Zeitdruck.
$\hat{\rho}$	Zeitdrucklimit.
$\mathrm{P}(\cdot)$	Einfluss des Zeitdrucks auf die Arbeitsgeschwindigkeit.
q	Qualität.
$Q(\cdot)$	Umrechnung von Defektmenge in Qualitätsniveau.
$R_N(\cdot)$	Normalverteilungsfunktion gemäß der gaußschen Normalverteilung.
$\vec{r}$	Ergebnisvektor einer Handlung im VIE-Modell.
h	Einflussstärke von Fähigkeiten.
s	Artefaktumfang oder Arbeitsumfang allgemein.
t	Zeitpunkt.
t_Δ	Zeitdauer.
$t_{\Delta\text{Konto}}$	Arbeitszeitkonto.
$t_{\Delta\text{Vertrag}}$	Arbeitszeit pro Tag laut Vertrag.
τ	Tätigkeit / Aufgabe.
T	Menge von Tätigkeiten / Aufgaben.
v	Arbeitsgeschwindigkeit.
ψ_a	Sympathie/Antipathie für eine Aktivität.
ψ_{m}	Sympathie/Antipathie für einen anderen Projektbeteiligten.
$V(x)$	Valenz eines gegebenen Elements x im VIE-Modell.
$W(\cdot)$	Arbeitsleistung bei der Durchführung einer Tätigkeit.
w_c	Umfang der Arbeitsleistung.
w_q	Qualität der Arbeitsleistung.
ω	Einkommen eines Projektbeteiligten.
z	Kontexteinfluss auf die Arbeitsleistung.
$Z(\cdot)$	Kontexteinfluss auf die Arbeitsleistung (Funktion).

A.3. Definition der Zustände des Motivationsmodells

In Ergänzung zu Abbildung 2.37 in Abschnitt 2.4.5 werden in Abbildung A.1 die Zustände des Zustandsraums des Motivationsmodells im Detail beschrieben. Jeder Zustand ist dabei als ein Polygon im durch die Motivatoren und Hygienefaktoren aufgespannten Zustandsraum definiert.



(a) Übersicht (b) stabilisierende Zufriedenheit (c) stabilisierte Zufriedenheit

(d) progressive Zufriedenheit (e) diffuse Unzufriedenheit (f) fixierte Unzufriedenheit

(g) konstruktive Unzufriedenheit (h) Pseudozufriedenheit (i) resignative Unzufriedenheit

Abbildung A.1.: Zustande im Zustandsraum des Motivationsmodells

A.4. Liste der bei der Evaluation verwendeten Parameter

Bei der Evaluation wird eine Vielzahl unterschiedlicher Parameter des Simulationsmodells betrachtet. Dieser Abschnitt listet die einzelnen Parameter auf und beschreibt, wie die entsprechenden Werte aus dem Szenario und dem Simulationsverlauf ausgelesen werden. Manche Parameter erlauben es außerdem automatisch gesetzt zu werden. Für die meisten Parameter lässt sich dies direkt durchführen, für einige Parameter ist jedoch ein entsprechendes Verfahren notwendig, das hier ggf. beschrieben wird.

Um Parameter besser automatisiert belegen zu können, wird der Wertebereich aller Parameter benötigt. Da einige Parameter theoretisch unbeschränkt sind, müssen deren Wertebereiche für die Evaluation sinnvoll limitiert werden. Der im Folgenden angegebene Wertebereich entspricht dem durch die Evaluation geprüften Wertebereich, die theoretisch möglichen Grenzen sind ggf. in Klammern angegeben.

Mitarbeiter

Parameter:	**Energieniveau**
Typ:	dynamisch
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittliches Energieniveau aller Projektbeteiligten.
Belegung:	Das Energieniveau aller Projektbeteiligten wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Motivation**
Typ:	dynamisch
Standardwert:	1.0
Wertebereich:	$[0, 2.25]$
Messung:	Durchschnittliche Arbeitsmotivation aller Projektbeteiligten.
Belegung:	Alle Einflussfaktoren werden auf einen entsprechenden Wert gesetzt, um die vorgegebene Motivation zu erzeugen. Dazu muss der vorgegebene Wert mit dem jeweiligen Anspruch multipliziert werden. Die Motivation lässt sich nicht direkt setzen. Der vorgegebene Wert kann deshalb nur grob angeben, wie die Motivation sein wird.
Parameter:	**Fähigkeiten**
Typ:	dynamisch
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	Durchschnittliches Fähigkeitsniveau für alle Fähigkeiten (für alle Aktivitäten und Disziplinen) aller Projektbeteiligter.
Belegung:	Das Fähigkeitsniveau für Aktivitäten und Disziplinen aller Projektbeteiligten wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Artefaktwissen**
Typ:	Ausgabe
Standardwert:	—
Wertebereich:	$[0, \infty]$
Messung:	Durchschnittliches Artefaktwissen aller Projektbeteiligter. Dazu wird die Summe allen absoluten Artefaktwissens gebildet, und durch den Umfang des Produkts geteilt.
Belegung:	— (Artefaktwissen kann nur relativ gesetzt werden)

Parameter:	**relatives Artefaktwissen**
Typ:	Eingabe
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	— (Artefaktwissen wird nur absolut gemessen)
Belegung:	Das relative Artefaktwissen jeden Mitarbeiters zu jedem Artefakt wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Adaptionsfähigkeit**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für Adaptionsfähigkeit aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Rednerfähigkeit**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für Rednerfähigkeit aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Physische Ausdauer**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für physische Ausdauer aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Zeitdruckschranke**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für die Zeitdruckschranke aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Lernfähigkeit**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für Lernfähigkeit aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Wechselfähigkeit**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für die Aufgabenwechselfähigkeit aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Vergessensrate**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für die Vergessensrate aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Aktivitätssympathie**
Typ:	Eingabe
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Charakterwert für die Aktivitätssympathie aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Anspruch (Hygiene)**
Typ:	Eingabe
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Anspruchswert aller Hygienefaktoren aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Anspruch (Motivatoren)**
Typ:	Eingabe
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der Anspruchswert aller Motivatoren aller Projektbeteiligter wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Mitarbeitersympathie**
Typ:	Eingabe
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	Durchschnittswert aller Projektbeteiligten.
Belegung:	Der jeweilige Sympathiewert für alle anderen Projektbeteiligte wird bei allen Projektbeteiligten auf den vorgegebenen Wert gesetzt.

Produkt

Parameter:	**Schwierigkeit**
Typ:	Eingabe
Standardwert:	0.5
Wertebereich:	$[0, 1]$
Messung:	Durchschnittlicher Wert über die Schwierigkeit α_θ aller Artefakte A des Projekts. Die Artefakte werde dabei mit ihrem Umfang α_s gewichtet. $$\overline{\theta} = \frac{\sum_{\alpha \in A} \alpha_\theta \cdot \alpha_s}{\sum_{\alpha \in A} \alpha_s} \quad \text{(A.1)}$$
Belegung:	Das Schwierigkeitsniveau aller Artefakte wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Artefaktumfang**
Typ:	Eingabe
Standardwert:	$100U$
Wertebereich:	$[1, 3000(\infty)]$
Messung:	Durchschnittlicher Wert aller Umfänge α_s aller Artefakte A des Projekts. $$\overline{s} = \frac{\sum_{\alpha \in A} \alpha_s}{\lvert A \rvert} \quad \text{(A.2)}$$
Belegung:	Der Umfang aller Artefakte wird auf die den vorgegebenen Wert gesetzt.
Parameter:	**Einfluss zusätzlichen Wissens**
Typ:	Eingabe
Standardwert:	0.7
Wertebereich:	$[0, 1]$
Messung:	Durchschnittlicher Wert über den Einflussfaktor α_{h_*} aller Artefakte A des Projekts. Die Artefakte werde dabei mit ihrem Umfang α_s gewichtet. $$\overline{h_*} = \frac{\sum_{\alpha \in A} \alpha_{h_*} \cdot \alpha_s}{\sum_{\alpha \in A} \alpha_s} \quad \text{(A.3)}$$
Belegung:	Der Parameter für den Einfluss zusätzlichen Wissens aller Artefakte wird auf den vorgegebenen Wert gesetzt.
Parameter:	**relative Vollständigkeit**
Typ:	Dynamisch
Standardwert:	0.0
Wertebereich:	$[0, 1]$
Messung:	Quotient aus aktuell fertiggestelltem Artefaktumfang α_c und Umfang des Produkts. $$c_{\text{rel}} = \frac{\sum_{\alpha \in A} \alpha_c}{\sum_{\alpha \in A} \alpha_s} \quad \text{(A.4)}$$
Belegung:	Die relative Vollständigkeit aller Artefakte des Produkts wird auf den vorgegebenen Wert gesetzt. Die Qualität des Artefakts wird auf 0.7 gesetzt. Dazu werden die benötigten Defekte mit dem entsprechenden Elementarmodell erzeugt (vgl. Formel 2.26).

Parameter:	**Fortschrittsänderung**
Typ:	Delta
Standardwert:	—
Wertebereich:	$[-\infty, +\infty]$
Messung:	Differenz aus dem defektfreien Fortschritt nach Ende und vor Beginn des Projekts. Der defektfreie Fortschritt berechnet sich als Summe der Artefaktfortschritte α_c, abzüglich des doppelten im Artefakt vorhandenen Defektumfangs α_λ, über alle Artefakte A des Projekts. Der Defektumfang wird doppelt abgezogen, da gleichermaßen Aufwand in die Identifikation der Defekte, wie in deren Behebung investiert werden muss, was in Summe einem Aufwand des doppelten Defektumfangs entspricht. $$c_{\text{defektfrei}} = \sum_{\alpha \in A} \alpha_c - 2 \cdot \alpha_\lambda \quad \text{(A.5)}$$
Belegung:	—

Parameter:	**Qualitätsänderung**
Typ:	Delta
Standardwert:	0.7
Wertebereich:	$[0, 1]$
Messung:	Die durchschnittliche Qualität wird als Summe der Qualität q über alle Artefakte A des Projekts, gewichtet mit dem jeweiligen Artefaktfortschritt c, berechnet. $$\overline{q} = \frac{\sum_{\alpha \in A} \alpha_q \cdot \alpha_c}{\sum_{\alpha \in A} \alpha_c} \quad \text{(A.6)}$$ Die Qualitätsänderung berechnet sich als Differenz aus der durchschnittlichen Qualität nach Projektende und vor Projektbeginn.
Belegung:	Alle Artefakte werden auf das vorgegebene Qualitätsniveau gesetzt. Dazu werden die benötigten Defekte mit dem entsprechenden Elementarmodell erzeugt (vgl. Formel 2.26).

Aktivitäten

Parameter:	**Quantitätsmodifikator für Fortschrittsaktivitäten**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittlicher Quantitätsmodifikator über alle im Szenario vorhandenen Fortschrittsaktivitäten.
Belegung:	Der Quantitätsmodifikator aller Fortschrittsaktivitäten wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Qualitätsmodifikator für Fortschrittsaktivitäten**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittlicher Qualitätsmodifikator über alle im Szenario vorhandenen Fortschrittsaktivitäten.
Belegung:	Der Qualitätsmodifikator aller Fortschrittsaktivitäten wird auf den vorgegebenen Wert gesetzt.

Parameter:	**Quantitätsmodifikator für Qualitätssicherungsaktivitäten**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittlicher Quantitätsmodifikator über alle im Szenario vorhandenen Qualitätssicherungsaktivitäten.
Belegung:	Der Quantitätsmodifikator aller Qualitätssicherungsaktivitäten wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Qualitätsmodifikator für Qualitätssicherungsaktivitäten**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittlicher Qualitätsmodifikator über alle im Szenario vorhandenen Qualitätssicherungsaktivitäten.
Belegung:	Der Qualitätsmodifikator aller Qualitätssicherungsaktivitäten wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Quantitätsmodifikator für Kommunikationsaktivitäten**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, 3(\infty)]$
Messung:	Durchschnittlicher Quantitätsmodifikator über alle im Szenario vorhandenen Kommunikationsaktivitäten.
Belegung:	Der Quantitätsmodifikator aller Kommunikationsaktivitäten wird auf den vorgegebenen Wert gesetzt.
Parameter:	**Arbeitsgeschwindigkeit (Fortschrittsaktivitäten)**
Typ:	Ausgabe
Standardwert:	1.0
Wertebereich:	$[0, \infty]$
Messung:	Die Arbeitsleistung, die im Rahmen der einzelnen Fortschrittstätigkeiten berechnet wird, wird mitprotokolliert. Daraus lässt sich die durchschnittliche Fortschrittsgeschwindigkeit berechnen, indem die Summe des geschaffenen Artefaktumfangs durch die Summe der dafür benötigten Zeit geteilt wird.
Belegung:	—
Parameter:	**Arbeitsqualität (Fortschrittsaktivitäten)**
Typ:	Ausgabe
Standardwert:	0.7
Wertebereich:	$[0, 1$
Messung:	Die Arbeitsqualität, die im Rahmen der einzelnen Fortschrittstätigkeiten berechnet wird, wird mitprotokolliert. Daraus lässt sich die durchschnittliche Qualität der Fortschrittsaktivitäten berechnen, indem die protokollierten Werte mit der jeweiligen Tätigkeitsdauer gewichtete, und anschließend der Durchschnittswert berechnet wird.
Belegung:	—

Parameter:	**Arbeitsgeschwindigkeit (Qualitätssicherungsaktivitäten)**
Typ:	Ausgabe
Standardwert:	1.0
Wertebereich:	$[0, \infty]$
Messung:	Die Arbeitsleistung, die im Rahmen der einzelnen Qualitätssicherungstätigkeiten berechnet wird, wird mitprotokolliert. Daraus lässt sich die durchschnittliche Geschwindigkeit der Qualitätssicherungsaktivitäten berechnen, indem die Summe der aufgedeckten Defekte durch die Summe der dafür benötigten Zeit geteilt wird.
Belegung:	—

Parameter:	**Arbeitsqualität (Qualitätssicherungsaktivitäten)**
Typ:	Ausgabe
Standardwert:	0.7
Wertebereich:	$[0, 1]$
Messung:	Die Arbeitsqualität, die im Rahmen der einzelnen Qualitätssicherungstätigkeiten berechnet wird, wird mitprotokolliert. Daraus lässt sich die durchschnittliche Qualität der Qualitätssicherungsaktivitäten berechnen, indem die protokollierten Werte mit der jeweiligen Tätigkeitsdauer gewichtete werden, und daraus anschließend der Durchschnittswert berechnet wird.
Belegung:	—

Parameter:	**Arbeitsgeschwindigkeit (Kommunikation)**
Typ:	Ausgabe
Standardwert:	$^{50.0}/_{h}$
Wertebereich:	$[0, \infty]$
Messung:	Die Arbeitsleistung, die im Rahmen der einzelnen Kommunikationstätigkeiten berechnet wird, wird mitprotokolliert. Daraus lässt sich die durchschnittliche Kommunikationsgeschwindigkeit berechnen, indem die Summe des übertragenen Artefaktwissens durch die Summe der dafür benötigten Zeit geteilt wird.
Belegung:	—

Projekt

Parameter:	**Produktivität des Teams**
Typ:	Ausgabe
Standardwert:	1.0
Wertebereich:	$[-\infty, +\infty]$
Messung:	Die Produktivität ist der Quotient aus defektfreiem Arbeitsfortschritt und der dafür benötigen Zeit. Der defektfreie Arbeitsfortschritt berechnet sich wie in Formel A.5 angegeben. $$\frac{\overline{c}_{\text{defektfrei}_{t_{\text{Ende}}}} - \overline{c}_{\text{defektfrei}_{t_{\text{Beginn}}}}}{t_{\text{Ende}} - t_{\text{Beginn}}} \quad \text{(A.7)}$$
Belegung:	—

Parameter:	**Produktivität pro Mitarbeiter**
Typ:	Ausgabe
Standardwert:	1.0
Wertebereich:	$[-\infty, +\infty]$
Messung:	Die durchschnittliche Produktivität der Mitarbeiter berechnet sich wie die *Produktivität des Teams*. Das Ergebnis wird anschließend durch die Teamgröße geteilt, um die Produktivität pro Mitarbeiter zu erhalten.
Belegung:	—

Parameter:	**Teamgröße**
Typ:	Eingabe
Standardwert:	1
Wertebereich:	$[1, 15(\infty)]$
Messung:	Anzahl der Projektbeteiligten.
Belegung:	Es werden dem Projekt entsprechend viele Projektbeteiligte hinzugefügt, die nach den jeweiligen Vorschriften des Szenarios erstellt werden, bzw. zufällig gewählte Projektbeteiligte aus dem Projekt entfernt, bis die Vorgabe erreicht ist.

Parameter:	**relativer Zeitaufwand**
Typ:	Ausgabe
Standardwert:	1.0
Wertebereich:	$[0, \infty]$
Messung:	Benötigte Zeit pro Arbeitseinheit defektfreiem Produktfortschritts. Der defektfreie Fortschritt berechnet sich wie in Formel A.5 beschrieben. Der defektfreie Fortschritt wird zu Beginn und am Ende des Projekts gemessen und die Differenz gebildet. Diese wird in Relation zur aufgewendeten Zeit gesetzt. $$\frac{t_{\text{Ende}} - t_{\text{Beginn}}}{\overline{c}_{\text{defektfrei}_{t_{\text{Ende}}}} - \overline{c}_{\text{defektfrei}_{t_{\text{Beginn}}}}} \quad \text{(A.8)}$$
Belegung:	—

Parameter:	**relativer Zeitaufwand pro Mitarbeiter**
Typ:	Ausgabe
Standardwert:	1.0
Wertebereich:	$[0, \infty]$
Messung:	Investierte Mannstunden pro geschaffener Arbeitseinheit. Die Messung wird wie beim *relativen Zeitaufwand* vorgenommen (vgl. Formel A.8). Das Ergebnis wird mit der Teamgröße multipliziert, um den Zeitaufwand pro Mitarbeiter zu bekommen.
Belegung:	—

Parameter:	**Zeitdruck**
Typ:	Eingabe
Standardwert:	1.0
Wertebereich:	$[0, \infty]$
Messung:	Der im Projekt herrschende Zeitdruck wird im Rahmen der einzelnen Tätigkeiten mitprotokolliert. Daraus lässt sich der durchschnittliche Zeitdruck berechnen, indem der Mittelwert der mit der Dauer der jeweiligen Tätigkeiten gewichteten Zeitdruckwerte berechnet wird.
Belegung:	Der Zeitdruck wird über den Projektplan gesteuert. Der Projektleiter-Agent wählt die Dauer aller Tätigkeiten so, dass ein entsprechender Zeitdruck erzeugt wird.

A.5. Navarros 86 fundamentale Regeln des Software Engineering

Folgende Aufzählung enthält alle in [Navarro, 2006] im *Anhang A – "The Fundamental Rules of Software Engineering"* aufgezählten Regeln. Einige davon wurden in einzelne Unterpunkte aufgeteilt, um die darin enthaltenen Aussagen einzeln betrachten zu können.

Im Rahmen dieser Arbeit wurden die einzelnen Aussagen nach zwei unterschiedlichen Arten kategorisiert (vgl. Abschnitt 3.1). Im Folgenden werden zu jeder Aussage die gewählte Kategorien angegeben, und kurz erläutert, warum die Aussage so eingeordnet wurde.

1. »*If you don't do a system architectual design with well-defined interfaces, integration will be a big mess.*« [Navarro, 2006, S. 262] (Entwicklungsprozess, modellierbar)

   Sowohl der Schnittstellenentwurf, als auch die Integration werden im Simulationsmodell als Artefakte dargestellt. Zwischen Artefakten können Abhängigkeiten modelliert werden. Die Aussage kann daher, wie in Abschnitt 4.4.1 gezeigt, modelliert werden.

2. »*Design before coding*« [Navarro, 2006, S. 262] (Entwicklungsprozess, modellierbar)

   Im Simulationsmodell werden Entwurf und Codierung durch unterschiedliche Artefakte repräsentiert. Die Aussage kann durch Abhängigkeiten zwischen diesen Artefakten umgesetzt werden. Dabei stehen zwei Möglichkeiten zur Verfügung: Durch das Einfügen einer strikten Vorbedingung wird die Bearbeitung eines Codeartefakts erst möglich, wenn die entsprechenden Entwurfsartefakte fertig gestellt sind. Alternativ kann durch die Deklaration der Entwurfsartefakte als Kontext der Codeartefakte, eine Beziehung zwischen Qualität und Vollständigkeit des Entwurfs und den Ergebnissen der Codierung erzeugt werden.

   Es können, wie in Abbildung A.2 gezeigt, auch beide Abhängigkeiten gleichzeitig gelten, um sowohl die Reihenfolge zu erzwingen, als auch die Ergebnisse der Implementierung von der Qualität des Entwurfs abhängig zu machen.

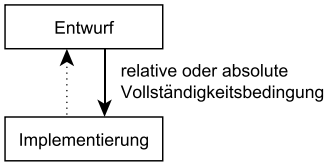


Abbildung A.2.: Modellierung der Abhängigkeiten zwischen Entwurf und Implementierung

3. »*If a project is late and you add more people, the project will be even later.*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

   Diese Aussage wird mit Hypothese 13 direkt am Simulationsmodell gezeigt.

4. »*Team members that are new to a project are less productive (1/3 to 2/3 less) than the adequately trained people.*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

   Die erforderlichen Messergebnisse können im Rahmen der Experimente zu Hypothese 13 erhoben werden. Dazu wird die durchschnittliche Produktivität der neuen Mitarbeiter in einem kurzen Zeitraum nach ihrem Eintreten in das Projekt gemessen.

5. »*The average newly hired employee is about half as productive as an experienced employee.*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

   Diese Aussage deckt sich mit Aussage 4 und kann entsprechend geprüft werden.

6.1 »*Two factors that affect productivity are work force experience level [...].*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

   Diese Aussage entspricht Hypothese 1 und kann wie in Abschnitt 3.4.1 beschrieben geprüft werden.

6.2 »*Two factors that affect productivity are [...] and level of project familiarity due to learning-curve effects.*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

Die Vertrautheit mit dem Projekt wird im Simulationsmodell über das Artefaktwissen der Projektbeteiligten abgebildet.

Hypothese 2 betrachtet den Zusammenhang zwischen dem Artefaktwissen des Mitarbeiters und seiner Produktivität.

Im Rahmen von Hypothese 6 wird untersucht, wie die Produktivität von der jeweiligen Lernfähigkeit des Mitarbeiters abhängt, die wiederum die Lernkurve definiert. Die Ergebnisse zeigen, dass sich das Simulationsmodell entsprechend der obigen Aussage verhält.

7. »*Developers' productivity varies greatly depending on their individual skills (experience concerning a development activity, knowledge of the tools, methods, and notations used, etc.).*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

Im Simulationsmodell können Fähigkeiten bezüglich der Entwicklungsaktivitäten direkt als Aktivitätsfähigkeit modelliert werden. Erfahrung mit Werkzeugen, Methoden, und Notationen, etc. können als fachliche Fähigkeit modelliert werden. Hypothese 1 untersucht, ob Mitarbeiter, die auf einem entsprechenden Gebieten bessere Fähigkeiten haben, dort eine höhere Produktivität erreichen.

8. »*Using better and fewer people is more productive than using more less qualified people.*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

Im Rahmen von Hypothese 12 wird untersucht, wie sich die Veränderung der Mitarbeiterzahl und die Veränderung deren Fähigkeiten auf die Produktivität auswirkt.

9. »*The greater the number of developers working on a task simultaneously, the faster that task is finished, but more overall effort is required due to the growing need for communication among developers. Thus, the productivity of the individual developer decreases.*« [Navarro, 2006, S. 262] (Personalmanagement, validierbar)

Die Experimente zu Hypothese 10 zeigen, dass größere Mitarbeiterteams zu einer niedrigeren Projektdauer führen.

Die Experimente zu Hypothese 11 zeigen, dass die Effizienz jedes einzelnen Mitarbeiters jedoch in größeren Teams geringer ist.

10. »*The earlier problems are discovered, the less the overall cost will be.*« [Navarro, 2006, S. 262] (Entwicklungsprozess, modellierbar)

Fehler, die im frühen Projektverlauf gemacht werden, bzw. Probleme die nicht rechtzeitig erkannt und adäquat behandelt werden, verursachen typischerweise einen größeren Schaden, als wenn sie im späten Projektverlauf passieren. Grund dafür ist, dass die einzelnen Teile eines Projekts aufeinander aufbauen und Fehler sich somit fortsetzen. Je früher ein Fehler passiert, bzw. je länger er unerkannt bleibt, umso mehr Artefakte entstehen auf Basis des fehlerhaften Artefakts. Diese Artefakte sind dann meist auch fehlerhaft. Häufig entstehen auf Basis eines Artefakts gleich mehrere weitere Artefakte, die Fehlerfortpflanzung ist deshalb häufig nicht linear, sondern exponentiell. Je mehr fehlerhafte Artefakte im Entwicklungsprozess entstanden sind, umso höher sind auch die dadurch entstandenen Kosten.

Hypothese 7 sagt aus, dass das Ergebnis eines Arbeitsschritts auch von den gegebenen Voraussetzungen abhängt. In Abschnitt 3.4.1 wird gezeigt, dass diese Hypothese für das Simulationsmodell akzeptiert werden muss.

Als Voraussetzungen werden die Vollständigkeit, sowie die Qualität, der als Kontext angegebenen Artefakte bewertet. Die oben genannte Aussage kann modelliert werden, indem die Artefakte des Projekts mittels Kontextverknüpfungen in einen geeigneten Zusammenhang gebracht werden. Dadurch sinkt die Qualität und Geschwindigkeit des Bearbeitungsprozesses bezogen auf das betrachtete Artefakt, wenn dessen Kontextartefakte eine schlechte Qualität

oder Unvollständigkeit aufweisen. Dies führt typischerweise wiederum zu höheren Entwicklungskosten.

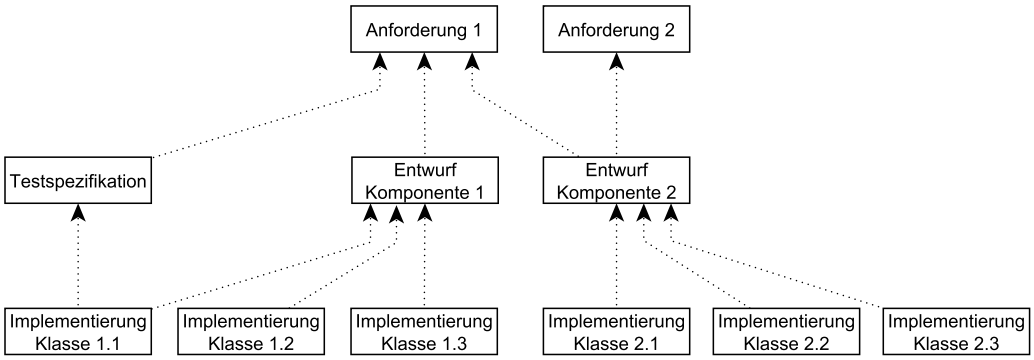


Abbildung A.3.: Produktstruktur mit Kontextabhängigkeiten

Abbildung A.3 zeigt ein Beispiel für eine mögliche Umsetzung einer entsprechenden Produktstruktur. Die vorgesehene Bearbeitungsreihenfolge ist von oben nach unten dargestellt. Ist hier beispielsweise *Anforderung 1* fehlerhaft und hat somit eine niedrige Qualität, so wirkt sich das direkt auf den *Entwurf von Komponenten 1* und *2* aus. Da die Implementierung der Klassen wiederum auf dem jeweiligen Entwurf basiert, wirkt sich der Defekt in der Anforderung indirekt auch hier aus. Die *Testspezifikation* für *Klasse 1.1* basiert ebenfalls auf der fehlerhaften Anforderung, und ist daher wahrscheinlich auch fehlerhaft. Wird der Defekt in *Anforderung 1* schon während der Anforderungsanalyse entdeckt, muss lediglich dieses Artefakt korrigiert werden. Wird der Defekt erst spät, beispielsweise während den Tests gefunden, so müssen in diesem Beispiel 10 Artefakte korrigiert werden, was erheblich mehr Kosten verursacht als die reine Korrektur von *Anforderung 1*.

11.1 »*The error detection effectiveness of reviews depends greatly on the qualifications and preparations of the reviewers [. . .].*« [Navarro, 2006, S. 263] (Personalmanagement, validierbar)

Diese Aussage wird in der allgemeineren Form im Rahmen von Hypothese 1 untersucht. Dabei wird gezeigt, dass die Fähigkeiten, und damit die Qualifikation, eines Mitarbeiters Auswirkungen auf seine Produktivität haben.

Je nachdem, um welche Art der Vorbereitung es sich handelt, entspricht die Vorbereitung im Simulationsmodell dem Projektwissen, z. B. der Kenntnis über den Aufbau des Prüflings, oder den fachlichen Fähigkeiten, z. B. dem Anwendungsdomänenwissen des Inspektors.

11.2 »*The error detection effectiveness of reviews depends greatly on [. . .] and the completeness and correctness of the documents used as a reference.*« [Navarro, 2006, S. 263] (Entwicklungsaktivität, modellierbar)

Diese Aussage lässt sich, wie in Abschnitt 4.4.2 gezeigt, modellieren, indem ein oder mehrere Referenzdokumente in Form von einzelnen Artefakten angelegt werden. Diese bilden einen Teil des Kontextes des Prüflings.

Im Rahmen von Hypothese 7 wird gezeigt, dass die Produktivität von der Qualität und Vollständigkeit der Kontextdokumente abhängt.

12. »*Reviews of non-technical documents (e. g., requirements specification, user manual) are more effective if the customer is involved.*« [Navarro, 2006, S. 263] (Entwicklungsaktivität, modellierbar)

Diese Aussage gilt, weil der Kunde typischerweise mehr Wissen über das von ihm erwartete Ergebnis hat, als die Entwickler. Dadurch ist er dazu prädestiniert, Ergebnisse, beispielsweise in Form einer Inspektion zu beurteilen.

Dieser Sachverhalt kann, wie in Abschnitt 4.4.3 ausführlicher gezeigt, modelliert werden, indem für das entsprechende Artefakt eine fachliche Fähigkeit (z. B. Anwendungsdomänenwissen)

gefordert wird, die nur der Kunde besitzt. Der Kunde selbst gilt als Projektbeteiligter, der an bestimmten Aktivitäten, wie beispielsweise einer Inspektion teilnehmen kann.

Eine höhere Produktivität bei der Durchführung der Inspektion ergibt sich dann entsprechend Hypothese 1, durch die höhere fachliche Fähigkeit. Werden für das zu begutachtende Artefakt neben dem Anwendungsdomänenwissen noch weitere Fähigkeiten benötigt, beispielsweise technische Fähigkeiten, die die Entwickler besitzen, der Kunde jedoch nicht, kann die Produktivität noch weiter gesteigert werden, indem Kunde und Entwickler gemeinsam am Artefakt arbeiten, und sich in ihren Fähigkeiten entsprechend ergänzen.

13. »*Develop tests before doing the coding.*« [Navarro, 2006, S. 263] (Entwicklungsaktivität, modellierbar)

    Diese Aussage wurde im Rahmen der testgetriebenen Entwicklung gemacht, und kann in einem passenden Prozessmodell umgesetzt werden. Ob testgetriebene Entwicklung, immer der beste Weg ist, soll hier nicht diskutiert werden. In Abschnitt 4.4.4 wird gezeigt, wie sich testgetrieben Entwicklung mit dem Simulationsmodell umsetzen lässt.

14. »*Extreme time pressure leads to decreased productivity.*« [Navarro, 2006, S. 263] (Personalmanagement, validierbar)

    Diese Aussage, die auf einer Veröffentlichung von Anke Drappa basiert, bezieht sich vermutlich nur auf Zeitdruck, der über einen längeren Zeitraum hinweg aufrechterhalten bleibt. Dies wird aus der Quelle jedoch nicht eindeutig klar.

    Im Rahmen von Hypothese 19 wird untersucht, wie sich das Simulationsmodell für einen erhöhten Zeitdruck hinweg verhält. Es lässt sich zeigen, dass die Produktivität der Mitarbeiter langfristig sinkt.

15. »*Extreme time pressure leads to a faster rate at which errors are made, which leads to a further delay in completion date.*« [Navarro, 2006, S. 263] (Personalmanagement, validierbar)

    Im Rahmen der Experimente zu Hypothese 19 und Hypothese 20 wird unter anderen auch die Auswirkung von Zeitdruck auf die Qualität der Arbeitsergebnisse untersucht. Dabei hat sich gezeigt, dass das Simulationsmodell einen entsprechenden Zusammenhang zwischen Zeitdruck und Qualität der Arbeitsergebnisse umsetzt.

16. »*Error correction is most efficiently done by the document's author(s).*« [Navarro, 2006, S. 263] (Personalmanagement, validierbar)

    Die hier gemachte Aussage wird durch Hypothese 16 beschrieben, und kann, wie in Abschnitt 3.4.1 gezeigt, am Simulationsmodell nachgewiesen werden.

17. »*The more errors a document from a previous phase contains, the more errors will be passed on to the next document.*« [Navarro, 2006, S. 263] (Entwicklungsprozess, validierbar)

    Wird diese Aussage umgekehrt als „Je weniger Fehler ein Dokument einer früheren Phase enthält, umso weniger Fehler werden an das nachfolgende Dokument weitergegeben.“ formuliert, entspricht sie Hypothese 7, die wiederum für das Simulationsmodell, wie in Abschnitt 3.4.1 gezeigt, bestätigt werden kann.

    Vergleiche auch Aussage 10 dieser Liste.

18. »*Always test everything.*« [Navarro, 2006, S. 263] (Entwicklungsaktivität, irrelevant)

    Diese Aussage ist eine Handlungsanweisung und beschreibt keinen Sachverhalt, den das Simulationsmodell umsetzen kann. Sie ist daher für dessen Evaluation irrelevant.

    Da ungetestete Artefakte typischerweise ungenügende Qualität aufweisen, werden Qualitätssicherungsaktivitäten in Planspielen, die mit dem Simulationsmodell umgesetzt werden, dadurch motiviert, dass sie zu einer höheren Produktqualität und einer höheren Entwicklungsgeschwindigkeit führen können. Siehe hierzu Hypothese 7 und Hypothese 17.

19. »*Talk to users, not to customers to verify the prototype.*« [Navarro, 2006, S. 263] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf die konkrete Durchführung der Verifikation eines Prototyps und ist daher zu speziell, um sie in einem allgemeinen Modell für Planspiele umzusetzen. Es ist jedoch möglich, sowohl Nutzer als auch Kunde als Projektbeteiligte im Simulationsmodell zu modellieren, und anhand ihres Fach- und Artefaktwissens zu unterscheiden. Über gemeinsame Kommunikationsaktivitäten mit den Nutzern, bzw. deren Teilnahme an Inspektionen, entsteht so die Möglichkeit, einen Vorteil durch den Nutzer zu erlangen, der über den Kunden nicht möglich ist.

20. »*Inspection is the most cost-effective measure of finding problems in software.*« [Navarro, 2006, S. 263] (Entwicklungsprozess, modellierbar)

    Ob diese Aussage allgemein gültig ist, kann zur Diskussion gestellt werden. Unabhängig davon, kann der Sachverhalt modelliert werden, indem der Quantitäts- und Qualitätsmodifikator der Inspektionsaktivität im Verhältnis zu den anderen Qualitätssicherungsaktivitäten eines Szenarios höher gesetzt werden.

21. »*Software inspections find a high percentage of errors early in the development life cycle.*« [Navarro, 2006, S. 263] (Entwicklungsprozess, modellierbar)

    Der Sachverhalt kann modelliert werden, indem die Parameter der Inspektion, insbesondere deren Quantitätsmodifikator, entsprechend gesetzt werden. Durch die Einschränkung der Aktivitäten auf die Artefakttypen kann realisiert werden, dass in den ersten Phasen der Entwicklung, keine anderen Qualitätssicherungsaktivitäten, wie beispielsweise Tests, eingesetzt werden können. Die konkrete Umsetzung hängt jedoch vom eigentlichen Szenario und den Lernzielen des Planspiels ab.

22. »*The use of inspections can lead to defect prevention, because developers get early feedback with respect to the types of mistakes they are making.*« [Navarro, 2006, S. 263] (Entwicklungsprozess, fehlend)

    Diese Aussage wird vom Simulationsmodell nicht direkt unterstützt. Inspektionen können, wie jede andere Aktivität, zwar zu einer Steigerung relevanter Fähigkeiten führen, einen Einfluss auf die Defekttypen, die der entsprechende Mitarbeiter anschließend erzeugt, hat dies jedoch nicht. Bei der Durchführung einer Aktivität wird weiterhin das gesamte dieser Aktivität zugewiesene Spektrum an Defekttypen erzeugt.

23. »*Every group has one programmer that is 10 times more productive than everyone else.*« [Navarro, 2006, S. 263] (Personalmanagement, modellierbar)

    Dieser Sachverhalt lässt sich über die Fähigkeiten der Entwickler modellieren. Dazu werden die Parameter der Entwickler, insbesondere die Fähigkeit, das Energieniveau, Motivation, und Ausdauer entsprechend gesetzt.

24. »*If you disable Internet surfing, productivity will go down.*« [Navarro, 2006, S. 263] (Arbeitsumgebung, irrelevant)

    Dieser Aspekt gehört zur Gestaltung der Arbeitsumgebung, beziehungsweise zu den Werkzeugen, die Teil dieser Umgebung sind. Die Arbeitsumgebung ist nicht Teil des Simulationsmodells, und muss bei Bedarf zusätzlich implementiert werden. Diese Aussage ist für die Evaluation des Simulationsmodells daher nicht relevant.

25. »*The structure of the software reflects the structure of the organization that developed it.*« [Navarro, 2006, S. 264] (Arbeitsumgebung, irrelevant)

    Organisationsstrukturen sind nicht Teil des Simulationsmodells, und häufig auch nicht Teil eines Planspiels im Bereich der Softwaretechnik. Die Struktur der Software wird im Simulationsmodell über das Szenario vorgegeben. Im Rahmen seiner Entwicklung kann so eine Produktstruktur erstellt werden, die zu einer bestimmten Unternehmensstruktur passt. Das Simulationsmodell selbst, erzeugt jedoch keine solche Struktur.

26. »*Changing requirements are inevitable. Anticipating change with open architectures, adaptable designs, and flexible planning can help to mediate some of the ill effects of these changes.*« [Navarro, 2006, S. 264] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich zum einen auf die konkrete Gestaltung der Softwarearchitektur, zum anderen auf die Projektplanung. Die Architektur ist in Form des Produkts und seiner Artefakte durch das Szenario vorgegeben, und wird durch die Simulation nicht verändert. Sie wird typischerweise passend zu Storyline und die den verbundenen Projektbedingungen vorgegeben.

    Sich ändernde Anforderungen werden ebenfalls durch die Storyline in Form von expliziten Ereignissen umgesetzt, und ergänzen, beziehungsweise verändern das Szenario während des Spiels. Da dieser Teil zur Rahmenimplementierung gehört, ist er für die Evaluation des Simulationsmodells irrelevant.

    Der Aspekt der flexiblen Planung, das Einplanen von Pufferzeiten etc., ist Aufgabe des Projektleiters, und damit des Spielers. Somit gibt es auch hier keine Notwendigkeit, diesen Aspekt im Simulationsmodell zu evaluieren.

27. »*Design for change/variability.*« [Navarro, 2006, S. 264] (Entwicklungsaktivität, irrelevant)

    Diese Aussage entspricht der vorherigen. Siehe Punkt 26.

28. »*Use defensive programming.*« [Navarro, 2006, S. 264] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf die konkrete Durchführung der Implementierungsaktivität und ist somit nicht Teil des Simulationsmodells. Dieses behandelt die Aktivitätsdurchführung auf einer abstrakteren Ebene und beschränkt sich dabei auf deren Ergebnisse in Form von Fortschritt und Qualität. Die konkreten Techniken, die dabei eingesetzt werden, sind durch die Aktivität abstrahiert. Der hier angesprochene Aspekt ist Teil der Programmierausbildung und nicht des Projektmanagements.

29. »*Configuration management is good.*« [Navarro, 2006, S. 264] (Entwicklungsprozess, irrelevant)

    Konfigurationsmanagement gehört zu den Rahmenbedingungen, unter denen ein Projekt durchgeführt wird, und ist damit nicht Teil des Simulationsmodells. Die Entscheidung über die Verwendung von Konfigurationsmanagement in einem Planspiel kann gegebenenfalls als Teil der Storyline umgesetzt werden. Der Einfluss der Entscheidung auf die Simulationsergebnisse kann wiederum über das Setzen entsprechender Parameter des Szenarios, beispielsweise die Quantitäts- und Qualitätsmodifikatoren bestimmter Aktivitäten, realisiert werden.

30. »*Successful software is designed by people who understand the application of the software (e. g., a well-designed missile control program was designed by someone who understood missiles).*« [Navarro, 2006, S. 264] (Personalmanagement, modellierbar)

    Dieser Aspekt ist über Disziplinfähigkeiten modellierbar, das für bestimmte Artefakte notwendig beziehungsweise hilfreich ist. Diese können sich auch, wie hier gefordert, auf die Anwendungsdomäne beziehen. Im Rahmen von Hypothese 1 wird gezeigt, dass im Simulationsmodell eine höhere Fähigkeit zu einer höheren Produktivität führt.

31. »*Software development requires a substantial time commitment to learning the application domain.*« [Navarro, 2006, S. 264] (Personalmanagement, modellierbar)

    Dieser Aspekt lässt sich auf zwei unterschiedliche Arten realisieren. Zum einen kann das Wissen über die Anwendungsdomäne in Form von Disziplinfähigkeiten repräsentiert werden. Die Entwickler müssen sich diese Fähigkeiten zuerst aneignen, bevor sie mit ihrer eigentlichen Arbeit beginnen können. Dies sollte hauptsächlich über geeignete Schulungen passieren, die nicht Teil des Simulationsmodells sind, sondern als der Teil Rahmenimplementierung umgesetzt werden müssen. Zum anderen kann das benötigte Domänenwissen, wie in Abschnitt 4.4.3 erläutert, in Form eines Artefakts repräsentiert werden.

32. »*Broad application knowledge is acquired more through relevant experience than through training.*« [Navarro, 2006, S. 264] (Personalmanagement, fehlend)

    In der Simulation gewinnen die Projektbeteiligten Erfahrung durch die Tätigkeiten, die sie durchführen. Da explizites Training nicht Teil des Simulationsmodells ist, kann der Vergleich zwischen implizit gewonnener Erfahrung durch die tägliche Arbeit, und explizit erworbener Erfahrung durch Training, nicht gezeigt werden.

33. »*The more bugs you find, the more buggy the rest of your program will likely be.*« [Navarro, 2006, S. 264] (Entwicklungsprozess, validierbar)

    Wird der Sachverhalt umgekehrt formuliert „Je mehr Fehler in einem Programm sind, umso mehr werden gefunden.“, deckt er sich mit der Aussage von Hypothese 14. In Abschnitt 3.4.1 wird gezeigt, dass diese für das Simulationsmodell akzeptiert werden muss.

34. »*Tests reveal errors in the code. The better a test is prepared for, the higher amount of detected errors.*« [Navarro, 2006, S. 264] (Entwicklungsaktivität, modellierbar)

    Diese Aussage kann modelliert werden, indem für die Testaktivitäten ein entsprechendes Kontextartefakt verlangt wird, das durch die Testvorbereitung erzeugt wird. Dadurch werden während der Testaktivität umso mehr Defekte gefunden, je besser und umfangreicher diese Vorbereitung war. Der Zusammenhang zwischen den Voraussetzungen eines Arbeitsschritts und dessen Ergebnissen werden mit den Experimenten im Rahmen von Hypothese 7 gezeigt.

35. »*Sticking with a too-tight schedule increases cost due to a large work force.*« [Navarro, 2006, S. 264] (Personalmanagement, validierbar)

    Die eigentliche Aussage dieses Punkts besteht darin, dass größere Teams zwar durch ihre höhere Arbeitskraft mit einem engeren Projektplan zurechtkommen, jedoch die pro Kopf Produktivität, mit der Teamgröße sinkt, und dadurch insgesamt höhere Kosten entstehen.

    Im Rahmen von Hypothese 10 und Hypothese 11 werden diese beiden Aspekte am Simulationsmodell gezeigt.

36.1 »*Motivation is increased through monetary incentives (profit sharing, pay for performance, merit pay, work measurement with incentives, and moral measurement), creating a positive frame of mind at work (employee involvement in wellness programs and creating fun at work), encouraging a feeling of commitment and responsibility (worker participation in descition-making, getting employees to think like owners, self-managing work teams, commitment to productivity breakthroughs, and provding an environment with more freedom and less restrictions), and increasing schedule pressure (using visibile milestones and setting individual goals.)*« [Navarro, 2006, S. 264] (Personalmanagement, fehlend)

    Das Simulationsmodell setzt die in dieser Aussage festgehaltenen Faktoren nicht in dem hier beschriebenen Detailgrad um. Einige, wie beispielsweise die Auswirkungen von Zeitdruck, oder der Spaß bei der Arbeit sind im Motivationsmodell jedoch in abstrakter Weise vorhanden.

    Diese Aussage liefert einen guten Ansatz für die Konzeption einer Erweiterung des Simulationsmodells um eine Arbeitsumgebung. Diese kann als Teil des Simulationsmodells oder als Rahmenimplementierung umgesetzt werden, und entsprechende Auswirkungen auf die Motivationsparameter der Mitarbeiter haben. Dabei kann es notwendig sein, das Motivationsmodell entsprechend zu erweitern oder komplett auszutauschen.

36.2 »*Increased motivation leads to increased productivity which reduces cycle time.*« [Navarro, 2006, S. 264] (Personalmanagement, validierbar)

    Im Rahmen von Hypothese 8 wird zeigt, dass eine höhere Motivation der simulierten Mitarbeiter deren Produktivität erhöht.

37. »*Improving the work environment is done by making ergonomic considerations, giving employees enclosed offices to reduced background noise and interruptions, and giving employees*

access to required resources, such as computers, software tools, support staff, and information. Improving the work environment leads to increased productivity, which reduces cycle time.« [Navarro, 2006, S. 265] (Arbeitsumgebung, irrelevant)

Diese Aussage bezieht sich auf die Arbeitsumgebung, die nicht Teil des Simulationsmodells ist. Sie ist daher für dessen Evaluation irrelevant.

Die meisten der Aspekte können mit einer entsprechenden Rahmenimplementierung im Zuge der Planspielerstellung umgesetzt werden. Die dort angelegten Mechanismen können wiederum die Parameter des Szenarios beeinflussen, sodass die Entscheidungen des Spielers bezüglich der Arbeitsumgebung den Verlauf der Simulation beeinflussen.

38. »*Getting the most out of employees can be done by utilizing experts, employee training, skills assessment and job matching, and reducing turnover. Getting the most out of employees leads to increased productivity, which leads to decreased cycle time.*« [Navarro, 2006, S. 265] (Personalmanagement, teilweise validierbar)

    Das Erwerben neuer Fähigkeiten findet im Simulationsmodell nur über die Durchführung von Aktivitäten statt. Explizites Training der Mitarbeiter ist dort nicht umgesetzt, und muss bei Bedarf durch eine geeignete Erweiterung ergänzt werden. Im Rahmen von Hypothese 1 wird am Simulationsmodell gezeigt, dass Mitarbeiter mit höheren passenden Fähigkeiten eine höhere Produktivität haben.

    Die passende Auswahl der Mitarbeiter obliegt dem Spieler in seiner Rolle als Projektleiter, der die Entscheidungen hinsichtlich seines Personals zu treffen hat.

    Das Simulationsmodell unterstützt eine Kündigung durch die Mitarbeiter, löst diese jedoch nicht selbstständig aus. Dieses Verhalten muss über die Rahmenimplementierung hinzugefügt werden. Dazu liefert das Simulationsmodell im Rahmen der Motivationsberechnung Indikatoren, die verwendet werden können, um zu entscheiden, ob ein Mitarbeiter kündigen, oder krankheitsbedingt ausfällt. Bei krankheitsbedingten Ausfällen sollte auch das Energielevel des Mitarbeiters ausgewertet werden.

39. »*Improving the software development process can be done by formalizing the process, controlling quality, and taking advantage of tools. Improving the software process increase employees' motivation, which also increases their productivity.*« [Navarro, 2006, S. 265] (Entwicklungsprozess, fehlend)

    Das Simulationsmodell enthält keine Aspekte, die sich mit der Formalisierung des Prozesses beschäftigen.

    Die Verbesserung des Entwicklungsprozesses durch Werkzeuge ist ebenfalls kein Teil des Simulationsmodells, kann jedoch gegebenenfalls über eine entsprechende Rahmenimplementierung ergänzt werden. Der Spieler kann dann beispielsweise über den Einsatz von Werkzeugen entscheiden. Diese Entscheidungen können wiederum die Parameter, insbesondere Quantitäts- und Qualitätsmodifikatoren von Aktivitäten beeinflussen, um entsprechende Auswirkungen im Simulationsverlauf zu erreichen.

    Eine Steigerung der Motivation durch die Verbesserung des Entwicklungsprozesses ist nicht im Simulationsmodell abgebildet. Im Rahmen von Hypothese 8 wird gezeigt, dass eine Motivationssteigerung zu einer Steigerung der Produktivität führt.

40. »*Rework is usually due to customer requirements, product flaws, and communication breakdown between project members. Improving the process to reduce rework can be done by using prototyping and evolutionary development and by using formal specification methods, modern programming practices, and inspections. Reducing rework increases productivity.*« [Navarro, 2006, S. 265] (Entwicklungsprozess, teilweise validierbar)

    Nacharbeit ist typischerweise dann notwendig, wenn die Qualität der schon geleisteten Arbeit nicht ausreichend ist. Grund dafür kann entweder die Tatsache sein, dass das Ergebnis aufgrund von Mängeln im Erstellungsprozess eine unzureichende Qualität aufweist, oder dass das falsche

Ergebnis erzeugt wurde. Beide Aspekte werden im Simulationsmodell über das Qualitätsattribut ausgedrückt.

Die in dieser Aussage beschriebenen Ursachen und möglichen Lösungen sind im Simulationsmodell in dieser Form nicht abgebildet. Sich ändernde Anforderungen können durch entsprechende Spielereignisse eingefügt werden. Fehlende Kommunikation zwischen den Entwicklern kann zu einem Wissensdefizit führen, das wiederum die Produktivität verringert, wie im Rahmen von Hypothese 2 gezeigt wird.

Da der Prozess nicht Teil der Simulation ist, sondern entweder durch das entsprechende Szenario, oder direkt durch die Entscheidungen des Spielers umgesetzt wird, können keine Aussagen über die allgemeine Gültigkeit der oben aufgezählten Lösungsansätze bezüglich des Simulationsergebnisses gemacht werden. Einige Aspekte beziehen sich auf die Art und Weise, wie eine Aktivität durchgeführt wird. Da das Simulationsmodell von der konkreten Durchführung von Aktivitäten abstrahiert, sind die betreffenden Aussagen für die Evaluation irrelevant.

41. »*Design complexity can be reduced by using object-oriented design techniques. Reducing design complexity reduces product complexity, which increases productivity.*« [Navarro, 2006, S. 266] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf konkrete Designtechniken, und damit auf die Art und Weise wie eine Aktivität durchgeführt wird. Das Simulationsmodell abstrahiert von der konkreten Durchführung der Aktivitäten, die Aussage ist daher für dessen Evaluation nicht relevant.

42. »*Code complexity can be reduced by using modularization and object-oriented programming techniques. Reducing code complexity reduces product complexity, which increases productivity.*« [Navarro, 2006, S. 266] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf konkrete Designtechniken, und damit auf die Art und Weise, wie eine Aktivität durchgeführt wird. Das Simulationsmodell abstrahiert von der konkreten Durchführung der Aktivitäten. Die Aussage ist daher für dessen Evaluation nicht relevant.

43. »*Cognitive complexity can be reduced by modularization, multiple levels of abstraction, simulation, and prototyping. Reducing cognitive complexity reduces product complexity, which increases productivity.*« [Navarro, 2006, S. 266] (Entwicklungsaktivität, modellierbar)

    Die Modularität des Produkts wird im Simulationsmodell durch die Aufteilung auf die einzelnen Artefakte und deren Abhängigkeiten untereinander abgebildet. Im Rahmen von Hypothese 24 wird am Simulationsmodell gezeigt, dass sich die Produktivität nicht mit der Aufteilung des Produkts in kleinere Artefakte ändert. Da die Komplexität eines Artefakts durch seine Schwierigkeit dargestellt wird, muss bei einer Zerlegung zusätzlich dieser Parameter reduziert werden, um den hier beschriebenen Vorteil der Modularisierung zu erreichen.

44. »*Test complexity can be reduced by using testing tools, building the product with testing in mind, and testing for the type of environment a product will be used in. Reducing test complexity reduces product complexity, which increases productivity.*« [Navarro, 2006, S. 266] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf die konkrete Art der Durchführung der Testaktivität, beziehungsweise deren Berücksichtigung bei der Erstellung von Code. Das Simulationsmodell abstrahiert von der konkreten Durchführung der Aktivitäten. Diese Aussage ist für die Evaluation daher irrelevant.

    Der Vorteil von Werkzeugen kann über das Anlegen unterschiedlicher Testaktivitäten, mit und ohne Werkzeug nachgebildet werden, die sich hinsichtlich ihrer Parameter unterscheiden, um die oben beschriebenen Vorteil umzusetzen.

45. »*Management complexity can be reduced by using project management planning tools and methods. Reducing management complexity reduces product complexity, which increases productivity.*« [Navarro, 2006, S. 266] (Entwicklungsprozess, irrelevant)

    Diese Aussage bezieht sich auf das Projektmanagement, das nicht Teil des Simulationsmodells ist, sondern vom Spieler durchgeführt werden soll. Sie ist somit für die Evaluation des Simulationsmodells irrelevant.

46. »*Tasks can be elimanted or simplified by using automation of tasks (e. g., code generators, automated testing) and eliminating non-value added activities and low-priority tasks. This leads to increased productivity.*« [Navarro, 2006, S. 266] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf die konkrete Durchführung von Aktivitäten und ist somit für die Evaluation des Simulationsmodells irrelevant, da dieses von der konkreten Durchführung der Aktivitäten abstrahiert.

47.1 »*Nine ways to reduce cycle time are: increase the productivity, [...]*« [Navarro, 2006, S. 266] (Entwicklungsprozess, validierbar)

    Das Simulationsmodell definiert Produktivität als den Quotienten aus erzeugtem Fortschritt abzüglich der gemachten Defekte und der dafür investierten Zeit. Dass die Erhöhung der Produktivität zu einer kürzeren Entwicklungszeit führt, wird im Rahmen von Hypothese 4 gezeigt.

47.2 »*Nine ways to reduce cycle time are: [...], reduce rework, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, irrelevant)

    Da das Überarbeiten von Produktteilen zusätzliche Zeit in Anspruch nimmt, ist es zunächst naheliegend, dass sich hier Zeit einsparen lässt. Jedoch hängt dies auch vom angewendeten Prozess ab. Beim evolutionären Prototyping oder der Testgetriebenen Entwicklung ist das Überarbeiten der Produktteile ein wichtiger Bestandteil.

47.3 »*Nine ways to reduce cycle time are: [...], maximize software reuse, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, irrelevant)

    Die Wiederverwendung von Komponenten kann entweder innerhalb eines Projekts, oder über mehrere Projekte hinweg erfolgen. Hier ist vor alle die Gestaltung des Entwurfs ausschlaggebend, die im Simulationsmodell durch die Aufteilung des Produkts auf die einzelnen Artefakte festgelegt wird. Diese Aufteilung geschieht durch den Entwickler des Szenarios und nicht durch die Simulation.

47.4 »*Nine ways to reduce cycle time are: [...], reduce product complexity, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, validierbar)

    Die Produktkomplexität wird im Simulationsmodell durch die Schwierigkeit der Produktartefakte abgebildet. Im Rahmen von Hypothese 3 wird gezeigt, dass eine höhere Schwierigkeit zu einer niedrigeren Produktivität führt.

47.5 »*Nine ways to reduce cycle time are: [...], eliminate or simplify tasks, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, fehlend)

    Ob das Eliminieren oder Vereinfachen von Aufgaben möglich und sinnvoll ist, hängt von der konkreten Situation im Projekt ab. Dieser Aspekt wird im Simulationsmodell nicht umgesetzt.

47.6 »*Nine ways to reduce cycle time are: [...], maximize task concurrency, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, validierbar)

    Durch die parallele Bearbeitung von Aufgaben lässt sich die Entwicklungszeit reduzieren, vorausgesetzt, es stehen genügend entsprechende Mitarbeiter zur Verfügung. Eine wichtige Voraussetzung dafür ist die Usnabhängigkeit der Aufgaben. Dieser Aspekt lässt sich am Simulationsmodell im Rahmen von Hypothese 25 beobachten.

47.7 »*Nine ways to reduce cycle time are: [...], reduce undiscovered work, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, irrelevant)

Das Identifizieren aller Aufwände ist eine Grundvoraussetzung für eine optimale Gestaltung der Zeitpläne und Arbeitsabläufe. Es ist Aufgabe des Projektleiters, und damit des Spielers, die Zeitpläne zu entwickeln. Das Simulationsmodell enthält dazu keine entsprechenden Aspekte. Gegebenenfalls kann diese Aufgabe für den Spieler durch eine entsprechende Spielumsetzung und Rahmenimplementierung gefordert, erschwert, oder anderweitig beeinflusst werden.

47.8 »*Nine ways to reduce cycle time are: [...], reduce risk, [...].*« [Navarro, 2006, S. 266] (Entwicklungsprozess, fehlend)

Die Reduzierung von Risikofaktoren macht das Projekt beherrschbarer, führt jedoch nur zu einer kürzeren Entwicklungszeit, wenn die Eintrittswahrscheinlichkeit oder die Schwere der Auswirkungen im Eintrittsfall tatsächlich reduziert werden kann. Das Simulationsmodell enthält keine Aspekte zum Risikomanagement.

47.9 »*Nine ways to reduce cycle time are: [...], and use process models aimed at cycle time reduction.*« [Navarro, 2006, S. 266] (Entwicklungsprozess, irrelevant)

Dass Prozessmodelle die auf die Reduzierung von Entwicklungszeiten ausgelegt sind, diesen Zweck auch erfüllen, ist anzunehmen. Prozessmodelle werden entweder über das Szenario vorgegeben, oder sie werden durch die Entscheidung des Spielers in die Simulation eingebracht. In beiden Fällen sind sie kein Teil des Simulationsmodells.

48.1 »*Productivity is increased by increasing motivation, [...].*« [Navarro, 2006, S. 267] (Personalmanagement, validierbar)

Im Rahmen von Hypothese 8 wird gezeigt, dass eine höhere Motivation zu einer höheren Produktivität führt.

48.2 »*Productivity is increased by [...], improving the work environment, [...].*« [Navarro, 2006, S. 267] (Arbeitsumgebung, irrelevant)

Die Arbeitsumgebung ist nicht Teil des Simulationsmodells. Daher ist diese Aussage für dessen Evaluation nicht relevant.

48.3 »*Productivity is increased by [...], getting the best people for the job, [...].*« [Navarro, 2006, S. 267] (Personalmanagement, validierbar)

Die Experimente im Rahmen von Hypothese 1 zeigen, dass Mitarbeiter mit besseren Fähigkeiten eine höhere Produktivität haben.

48.4 »*Productivity is increased by [...], and maximizing reuse.*« [Navarro, 2006, S. 267] (Entwicklungsprozess, irrelevant)

Diese Aussage ist äquivalent zu Punkt 46 dieser Liste.

49. »*Product complexity can be reduced by reducing code complexity, design complexity, cognitive complexity, test complexity, and management complexity.*« [Navarro, 2006, S. 267] (Entwicklungsaktivität, irrelevant)

Die durch die Aufgabenstellung gegebene Komplexität des Produkts und seiner Artefakte wird über die Schwierigkeit der Artefakte im Szenario festgelegt (vgl. Punkt 43). Die Komplexität, die aus der Art und Weise, wie die einzelnen Artefakte erstellt werden, resultiert, wird vom Simulationsmodell nicht behandelt, da dieses die Entwicklungsaktivitäten nur abstrakt darstellt.

50. »*Decisions made in the upstream portion of the software development process (requirements and design) impact productivity, quality, and costs throughout the life cycle more than the other portions.*« [Navarro, 2006, S. 267] (Entwicklungsprozess, modellierbar)

Da Anforderungs- und Designartefakte häufig den Kontext für die Entwicklung weiterer Artefakte bilden, haben sie einen entsprechenden Einfluss auf die Produktivität bei der Erstellung

dieser nachfolgenden Artefakte. Der Zusammenhang zwischen Qualität und Vollständigkeit von Kontextartefakten, und der Produktivität bei der Erstellung weiterer Artefakte, wird im Rahmen von Hypothese 7 gezeigt.

Das Simulationsmodell deckt keine Aspekte der Betriebs- und Wartungsphase ab. Daher können auch die Auswirkungen von Entscheidungen in den frühen Phasen auf diesen Lebensabschnitt des Produkts nicht an der Simulation beobachtet werden. Die Qualität der einzelnen Artefakte gibt jedoch auch Aufschluss darüber, wie gut das Produkt hinsichtlich seiner Wartungsfreundlichkeit ist.

Die Simulation abstrahiert von der Art und Weise der Durchführung der einzelnen Entwicklungsaktivitäten. Der Spieler hat somit keine Möglichkeit, die oben beschriebenen Entscheidungen im Detail richtig oder falsch zu machen. Er kann lediglich über die Mitarbeiterauswahl und die Art der Aktivitäten, die durchgeführt werden, Einfluss darauf nehmen, und die Ergebnisse anhand der Qualität der erstellten Artefakte überprüfen.

51. »*The thin spread of application domain knowledge is a major phenomenon that greatly reduces software productivity and quality.*« [Navarro, 2006, S. 267] (Personalmanagement, modellierbar)

    Domänenwissen kann als fachliche Fähigkeit modelliert werden. Dadurch hat fehlendes Domänenwissen eine entsprechende Auswirkung auf die Produktivität der Mitarbeiter. Die Experimente im Rahmen von Hypothese 1 zeigen den Zusammenhang zwischen den Fähigkeiten und der Produktivität eines Mitarbeiters.

52. »*Specification mistakes often occure when designers do not have sufficient application knowledge to interprete the customer's intention from the requirements documents.*« [Navarro, 2006, S. 267] (Entwicklungsaktivität, modellierbar)

    Domänenwissen kann mittels Disziplinfähigkeiten modelliert werden (siehe Punkte 31 und 51). Der Zusammenhang zwischen Fähigkeiten und der Qualität darauf basierender Arbeitsschritte wird im Rahmen von Hypothese 1 gezeigt.

53. »*Requirements will appear to fluctuate when the development team lacks application knowledge and performs an incomplete analysis of the requirements.*« [Navarro, 2006, S. 267] (Personalmanagement, teilweise modellierbar)

    Die eigentliche Fluktuation wird vom Simulationsmodell nicht abgebildet. Jedoch hat das fehlende Domänenwissen, welches als Disziplinfähigkeiten modelliert werden kann (siehe Punkte31 und 51), einen Einfluss auf die Qualität der entsprechenden Artefakte.

    Eine explizite Fluktuation der Anforderungen, unabhängig von den Disziplinfähigkeiten der Entwickler, kann in einer entsprechenden Storyline, beispielsweise durch entsprechende Ereignisse während des Simulationsverlaufs umgesetzt werden.

54. »*Coordinating understanding of an application and its environment requires constant communication between customers and developers.*« [Navarro, 2006, S. 267] (Entwicklungsprozess, fehlend)

    Dieser Aspekt kann mit dem Simulationsmodell im hier geforderten Detailgrad nicht abgebildet werden. Zwar kann Kommunikation zum Wissensaustausch simuliert werden, diese bezieht sich jedoch primär auf die Entwickler. Eine Kommunikation mit dem Kunden ist nur insofern vorgesehen, dass im Szenario definierte Informationen vom Kunden zum Entwickler fließen. Eine Art von Feedback des Kunden zu den erstellten Inhalten gibt es nicht.

    Eine Möglichkeit den Kunden in den Entwicklungsprozess miteinzubeziehen ist es, ihn mit Inspektionsaktivitäten zu betrauen, die er aufgrund seines entsprechenden Fachwissens, das in Form von Disziplinfähigkeiten modelliert ist, besser durchführen kann als die Entwickler, um so effektiver Defekte in den erstellten Artefakten finden zu können.

55. »*Specification are almost always incomplete and fraught with ambiguities. Constant contact with the customer is required to abtain the correct requirements. Without this communication,*

the developers tend to make incorrect assumptions about what the customer wants.« [Navarro, 2006, S. 267] (Entwicklungsprozess, fehlend)

Siehe Punkt 54.

56. »*Fluctuating and conflicting requirements is a major phenomenon that greatly reduces software productivity and quality.*« [Navarro, 2006, S. 268] (Entwicklungsprozess, modellierbar)

Diese Aussage bezieht sich auf konkrete Inhalte bestimmter Artefakte. Das Simulationsmodell abstrahiert von solchen Inhalten. Fluktuationen und Konflikte von Anforderungen innerhalb eines Anforderungsartefakts können durch dessen Schwierigkeitsniveau ausgedrückt werden. Wie im Rahmen von Hypothese 1 gezeigt, führt eine höhere Schwierigkeit zu einer niedrigeren Produktivität und Qualität des Ergebnisses. Bauen weitere Artefakte auf den Anforderungsartefakten auf, so wirkt sich eine niedrigere Qualität auch entsprechend auf diese Artefakte aus, was im Rahmen von Hypothese 7 untersucht wird.

57. »*Communication and coordination breakdown is a major phenomenon that greatly reduces software productivity and quality.*« [Navarro, 2006, S. 268] (Entwicklungsprozess, validierbar)

Das Simulationsmodell umfasst Kommunikationsaktivitäten, die sowohl implizit als auch explizit stattfinden können. Hypothese 22 zeigt, dass fehlende Kommunikation zu einer niedrigeren Produktivität führt.

58. »*Truly exceptional designers that are extremly familiar with the application domain, skilled at communicating their technical vision to other project members, posses an exceptional ability to map between the behavior required of the application system and the computational structures that implement the behavior, and are recognized as the "intellectual core" of the project are a sacred resource.*« [Navarro, 2006, S. 268] (Personalmanagement, teilweise validierbar)

Fachwissen kann als Disziplinfähigkeiten modelliert werden. Die Relevanz von Fähigkeiten wird im Rahmen von Hypothese 1 gezeigt. Die Möglichkeit der Kommunikation von Domänenwissen hängt von der Gestaltung des Szenarios ab. Wird Domänenwissen als Disziplinfähigkeiten modelliert, kann es nicht durch Kommunikation geteilt werden. Es kann jedoch auch als Artefaktwissen modelliert werden, indem zusätzliche Artefakte dafür angelegt, und diese in den Kontext der jeweils darauf basierenden Artefakte gestellt werden. Den entsprechenden Mitarbeitern kann das dazugehörige Artefaktwissen zugewiesen werden, dass so wiederum mittels Kommunikation mit weiteren Mitarbeitern geteilt werden kann. Damit wirkt sich die Kommunikationsfähigkeit der Mitarbeiter indirekt auf die Produktivität des Teams aus.

59. »*New requirements frequently emerge during development since they could not be identified until portions of the system had been designed or implemented.*« [Navarro, 2006, S. 268] (Entwicklungsprozess, modellierbar)

Dieser Aspekt ist über Anforderungsartefakte die von Entwurf- bzw. Implementierungsartefakten abhängen, jedoch gleichzeitig eine Voraussetzung für diese darstellen, wie in 4.4.5 gezeigt, modellierbar. Dazu können Artefakt beispielseweise wechselseitig durch entsprechende Vorbedingungen miteinander verknüpft werden.

60. »*Besides a developer's ability to design and implement programs, his skills in resolving conflicting requirements, negotiating with the customer, ensuring that the development staff shares a consistent understanding of the design, and providing communications between two contending groups are crucial to project performance.*« [Navarro, 2006, S. 268] (Personalmanagement, modellierbar)

Diese Aussage umfasst viele unterschiedliche Aspekte, die in dieser Form nicht direkt durch das Simulationsmodell abgebildet werden. Sie betreffen jedoch konkrete Aktivitäten im Entwicklungsprozess, und können so mittels der entsprechenden Aktivitäts- und Disziplinfähigkeiten abgebildet werden, die sich wiederum, wie im Rahmen von Hypothese 1 gezeigt, auf die Produktivität auswirken.

61. »*Undiscovered work (work that was not considered in initial planning estimates) can be reduced by using formal methods, analysis of PERT sizing metrics, the Spiral life cycle model, and prototyping. Reducing undiscovered work leads to increased productivity.*« [Navarro, 2006, S. 268] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf die konkrete Aktivität der Aufwandsschätzung. Diese ist Teil des Projektmanagements, somit Aufgabe des Spielers, und nicht Teil des Simulationsmodells.

62. »*Risk can be reduced by using risk management techniques. Reducing risk leads to increased productivity.*« [Navarro, 2006, S. 269] (Entwicklungsprozess, fehlend)

    Risiken und Risikomanagement werden im Simulationsmodell nicht umgesetzt, und müssen über eine Erweiterung oder eine entsprechende Rahmenimplementierung hinzugefügt werden.

63. »*Inspections should be thought of as part of the development process, and time must be set aside accordingly. Once this is done, inspections can have a significant improvement in the development organization's ability to meet internal schedules.*« [Navarro, 2006, S. 269] (Entwicklungsprozess, validierbar)

    Dieser Aspekt kann als konsequente Qualitätssicherung während des gesamten Entwicklungsprozesses, insbesondere auch in den frühen Phasen, interpretiert werden. Welche Möglichkeiten der Qualitätssicherung in den einzelnen Phasen angewendet werden können, ist über das Szenario zu definieren.

    Im Rahmen von Hypothese 17 wird gezeigt, dass eine kontinuierliche Qualitätssicherung zu einer höheren Produktivität führt. Wie sich diese wiederum auf die Einhaltung von Projektplänen auswirkt, hängt von den Entscheidungen des Projektleiters ab.

64. »*Proper use of inspections can even shorten life cycle.*« [Navarro, 2006, S. 269] (Entwicklungsprozess, validierbar)

    Siehe Punkt 63.

65. »*Participants in the inspection team get a high degree of product knowledge, which leads to higher productivity.*« [Navarro, 2006, S. 269] (Personalmanagement, validierbar)

    Im Rahmen von Hypothese 15 wird gezeigt, dass Mitarbeiter ihr Artefaktwissen durch die Teilnahme an Qualitätssicherungsaktivitäten steigern können. Die Experimente zu Hypothese 2 zeigen, dass größeres Artefaktwissen zu einer höheren Produktivität führt.

66. »*Slower programmers show a great deal of improvement when using inspections.*« [Navarro, 2006, S. 269] (Personalmanagement, teilweise validierbar)

    Diese Aussage entspricht im Wesentlichen Punkt 65. Laut ihr profitieren jedoch besonders langsame Mitarbeiter durch eine Teilnahme an Inspektionen. Hier ist jedoch zu unterscheiden, ob das langsame Arbeiten durch mangelndes Wissen und Fähigkeiten, oder durch andere Gründe bedingt ist. Im zweiten Fall führt die Teilnahme an Inspektionen nicht zu einer Steigerung der Produktivität.

    Der Erwerb von Produktwissen durch das Inspizieren eines Artefakts wird von der Simulation abgedeckt und führt zur gewünschten Produktivitätssteigerung, wie im Rahmen von Hypothese 2 für das Simulationsmodell gezeigt wird. Eine Steigerung der Fähigkeiten, die zur Erstellung des inspizierten Artefakts beigetragen haben, wird vom Simulationsmodell jedoch nicht umgesetzt. Beispielsweise verbessert der Mitarbeiter seine Entwurfsfähigkeiten durch das Betrachten einer guten Systemarchitektur nicht, was jedoch zu erwarten wäre. Eine entsprechende Umsetzung dieses Aspekts fehlt im Simulationsmodell.

67. »*A new project assignee does not become productive until six month to a year into the project.*« [Navarro, 2006, S. 269] (Personalmanagement, validierbar)

    Die Experimente im Rahmen von Hypothese 13 zeigen, dass neu hinzugefügten Mitarbeiter die Produktivität des Teams zunächst senken.

68. »*Collaborators use hand gestures to uniquely communicate significant informations.*« [Navarro, 2006, S. 269] (Entwicklungsaktivität, irrelevant)

    Diese Aussage bezieht sich auf die konkrete Durchführung einer Aktivität und ist daher für die Evaluation des Simulationsmodells irrelevant.

69. »*Employers often limit the number of hours employees can work, resulting in further pressure to finish a project as quickly as possible.*« [Navarro, 2006, S. 269] (Personalmanagement, irrelevant)

    Dieser Aspekt ist Teil des Projektmanagements und wird gegebenenfalls durch den Spieler initiiert, oder durch das Szenario, beziehungsweise die Rahmenimplementierung vorgegeben.

70. »*The customer often changes deadlines to be earlier than originally agreed-upon, requiring negotiation with the customer for either allowing some deliverables to be delivered at the earlier date, with the rest being delivered later, or dropping some deliverables or requirements altogether.*« [Navarro, 2006, S. 269] (Projektrahmen, irrelevant)

    Dieser Aspekt ist Teil des Projektrahmens und kann gegebenenfalls als Element der Storyline umgesetzt werden. Er hat keinen direkten Einfluss auf die Simulation, sondern lediglich auf die vom Spieler durchgeführten Managementtätigkeiten. Durch eine geeignete Rahmenimplementierung oder entsprechende Ereignisse im Szenario, können die oben beschriebenen Situationen erzeugt werden, sodass der Spieler gezwungen wird sich damit zu beschäftigen.

71. »*Code comments and documentation are often produced at the end of a project, creating major problems when a team member is lost at short notice, leaving others to continue their work. This can be alleviate by heaving quality auditors require inspections at very short notice.*« [Navarro, 2006, S. 270] (Entwicklungsprozess, fehlend)

    Die hier erwähnte Dokumentation bezieht sich hier auf die Informationen, die im Simulationsmodell mittels Artefaktwissen dargestellt werden. Dieses Wissen wird entweder durch direkte Kommunikation weitergegeben, oder durch das Studium des Artefakts aus dem Artefakt entnommen. Dabei spiegelt sich der Umfang und die Qualität der Dokumentation, neben einer Reihe von anderen Einflussfaktoren, in der Artefaktqualität wider, die sich wiederum auf die Geschwindigkeit bei der Einarbeitung in das Artefakt auswirkt.

    Das Simulationsmodell unterscheidet nicht zwischen dem eigentlichen Artefakt und seiner Dokumentation. Eine besondere Betrachtung des Dokumentationsaspekts kann daher nicht stattfinden. Die einzige Möglichkeit, die das Simulationsmodell an dieser Stelle bietet, ist die Modellierung des Dokumentationsanteils als eigenständige Artefakte mit entsprechenden Kontextverknüpfungen.

72. »*Teams often change during projects (members are added and/or removed).*« [Navarro, 2006, S. 270] (Personalmanagement, irrelevant)

    Dieser Aspekt ist nicht Teil des Simulationsmodells, sondern gehört zum Szenario oder der Rahmenimplementierung. Da aus der Simulation jederzeit Mitarbeiter entfernt oder hinzugefügt werden können, lässt sich ein Wechsel im Team leicht durch entsprechende Ereignisse im Szenario umsetzen. Die Auswirkungen werden in der Simulation direkt sichtbar, ohne das weitere Anpassungen notwendig sind.

73. »*Sometimes the software used for development is upgraded to a new version during development, and despite claims that it is fully backward-compatible and won't affect their work, it usually introduces new problems.*« [Navarro, 2006, S. 270] (Arbeitsumgebung, irrelevant)

    Hier kommt es darauf an, welche Probleme durch eine solche Umstellung verursacht werden. Handelt es sich dabei um Mehrarbeit, die durch die Anpassung der Artefakte an die neuen Werkzeuge verursacht wird, können, beispielsweise durch Ereignisse in der Storyline, entweder die Fortschritte der Artefakte verringert, oder der Umfang des Artefakts entsprechend erhöht werden.

    Soll eine Verzögerung durch die Einarbeitung in die neue Software simuliert werden, kann dies durch das Hinzufügen von neuen fachlichen Fähigkeiten, die das Werkzeug betreffen, oder

durch eine manuelle Verzögerung in Form einer Schulung geschehen. Beides muss in einer entsprechenden Rahmenimplementierung umgesetzt werden und ist nicht Teil des Simulationsmodells.

74. »*Hardware crashes, and customers are often unsympathetic to this kind of delay.*« [Navarro, 2006, S. 270] (Arbeitsumgebung, irrelevant)

    Explizite Verzögerungen durch Hardwareausfälle müssen durch die Rahmenimplementierung realisiert werden. Dazu müssen lediglich die Arbeitsprozesse für den Zeitraum des simulierten Hardwareausfalls blockiert werden, indem beispielsweise die dazugehörigen Aktivitäten für diesen Zeitraum aus dem Szenario entfernt werden.

    Soll zusätzlich ein Verlust von Daten simuliert werden, kann bei den entsprechenden Artefakten der Fortschritt reduziert werden. Nicht reduziert wird dabei das Wissen der Mitarbeiter über den schon fertig gestellten, jedoch verlorengegangenen Teil. Dadurch kann dieser schneller neu erstellt werden, was im Rahmen von Hypothese 23 gezeigt wird.

    Die Auswirkungen der durch den Hardwareausfall entstandenen Verzögerung ergibt sich automatisch aus der Situation. Die Reaktion des Kunden muss in einer entsprechenden Rahmenimplementierung umgesetzt werden.

75. »*When a project is in its later stages of development, the development hardware and software tend to be under the greatest demand, and performance starts to suffer with lengthy compilations, builds, and test runs.*« [Navarro, 2006, S. 270] (Arbeitsumgebung, fehlend)

    Dieser Aspekt ist im Simulationsmodell nicht umgesetzt, da benötigte externe Ressourcen wie Hardware nicht Teil der Simulation sind.

76.1 »*Matching the tasks to the skills [. . .] of the people available increases productivity.*« [Navarro, 2006, S. 270] (Personalmanagement, validierbar)

    Im Rahmen von Hypothese 1 wird gezeigt, dass die Fähigkeiten des Mitarbeiters direkte Auswirkungen auf seine Produktivität hat. Da Fähigkeiten aufgabenspezifisch sind, wird der hier genannte Effekt adäquat umgesetzt.

76.2 »*Matching the tasks to [. . .] and the motivation of the people available increases productivity.*« [Navarro, 2006, S. 270] (Personalmanagement, validierbar)

    Im Rahmen von Hypothese 5 wird gezeigt, dass eine höhere Sympathie für eine Aufgabe zu einer höheren Produktivität führt.

77. »*Employee motivation is the strongest influence of productivity.*« [Navarro, 2006, S. 270] (Personalmanagement, validierbar)

    Dass Motivation der stärkste Einflussfaktor auf die Produktivität der Mitarbeiter ist, lässt sich am Simulationsmodell nicht nachweisen. Im Rahmen von Hypothese 8 wird jedoch gezeigt, dass die Motivation eines Mitarbeiters einen deutlichen Einfluss auf seine Produktivität hat.

78. »*Above a certain threshold, work conditions are not a powerful motivator, but below that threshold, they are a powerful de-motivator.*« [Navarro, 2006, S. 270] (Personalmanagement, validierbar)

    Die Experimente im Rahmen von Hypothese 9 zeigen, dass es einen Punkt gibt, an dem eine Steigerung der Arbeitsbedingungen keine Steigerung der Motivation mehr zur Folge hat.

79. »*The training of new employees is usually done by the "old-timers", which results in a reduced level of productivity an the "old-timers" part. Specifically, on the average, each new employee consumes in training overhead 20% of an experienced employee's time for the duration of the training or assimilation period.*« [Navarro, 2006, S. 270] (Personalmanagement, validierbar)

    Dieser Aspekt resultiert aus dem Informationsdefizit der neuen Mitarbeiter, welches diese versuchen durch Kommunikation mit den erfahrenen Mitarbeitern auszugleichen. Dadurch

werden diese von ihrer eigentlichen Arbeit abgelenkt, was zum oben beschriebenen Produktivitätsverlust führt. Im Rahmen von Hypothese 13 wird dieser Effekt am Simulationsmodell nachgewiesen. Der konkrete Wert des Mehraufwands durch die Einarbeitung der neuen Mitarbeiter hängt jedoch von unterschiedlichen Faktoren, wie beispielsweise der Auffassungsgabe der Mitarbeiter ab.

80. »*The average assimilation delay, the period of time it takes for a new employee to become fully productive, is 80 days.*« [Navarro, 2006, S. 271] (Personalmanagement, validierbar)

Diese Aussage ist inkonsistent zu der Aussage in Punkt 67, welche behauptet, dass das Erreichen der vollen Produktivität zwischen 6 und 12 Monaten benötigt. Die genaue Quantifizierung scheint hier umstritten, der eigentliche Effekt wird jedoch im Rahmen von Hypothese 13 gezeigt.

81. »*As schedule pressure increases, quality assurance activities (especially walkthroughs and inspections) are often relaxed or suspended altogether.*« [Navarro, 2006, S. 271] (Entwicklungsprozess, fehlend)

Dieser Aspekt betrifft einen Teil des Projektmanagements. Die Agenten haben darüber in der Simulation keine Kontrolle. Daher ist dieser Aspekt auch nicht Teil der Evaluation des Simulationsmodells, sondern betrifft ausschließlich die Entscheidungen des Spielers in einer solchen Situation.

Ein entsprechendes Verhalten der Agenten könnte, durch ihr jeweiliges Verhaltensmodell realisiert werden, das zusätzlich zu den Entscheidungen des Projektleiters wirkt. Die Verhaltensmodelle des Simulationsmodells verursachen jedoch kein solches Verhalten.

82. »*In the absence of schedule pressure, a full-time employee allocated, on average, 60% of his working hours to the project (the rest is slack time: reading mail, personal activities, non-project related company business, etc).*« [Navarro, 2006, S. 271] (Personalmanagement, validierbar)

Im Rahmen von Hypothese 21 wird gezeigt, dass die Produktivität der Mitarbeiter bei niedrigem Zeitdruck bis auf in etwa die Hälfte abfällt.

83. »*Under schedule pressure, people tend to increase their percentage of working hours spent on the project by as much as 100%, due to spending less time on off-project activities, such as personal business and non-project communication, and/or working overtime.*« [Navarro, 2006, S. 271] (Personalmanagement, validierbar)

Die Experimente zu Hypothese 21 zeigen, dass die Produktivität vom Zeitdruck abhängt, der auf die Mitarbeiter wirkt. Je höher der Zeitdruck ist, umso höher ist auch die Produktivität der Mitarbeiter.

84.1 »*The three "resource-type" variables that have the greatest impact on programmer productivity are the availybility of programming tools, [. . .].*« [Navarro, 2006, S. 271] (Arbeitsumgebung, modellierbar)

Die explizite Auswirkung der Verwendung von Werkzeugen ist nicht teil des Simulationsmodells. Dieser Aspekt kann über die Gestaltung der Aktivitäten eingeführt werden, indem unterschiedliche Aktivitäten für die Durchführung ohne und mit Werkzeug angelegt werden.

84.2 »*The three "resource-type" variables that have the greatest impact on programmer productivity are [. . .], the availability of programming practices, and programmer experience.*« [Navarro, 2006, S. 271] (Personalmanagement, validierbar)

Die Auswirkungen von Praxis und Erfahrung spiegelt sich in den Aktivitätsfähigkeiten, Disziplinfähigkeiten, und dem Produktwissen der Mitarbeiter wider. Die positiven Auswirkungen von Fähigkeiten und Wissen auf die Produktivität werden am Simulationsmodell im Rahmen von Hypothese 1 und 2 gezeigt.

85. »*The two "task-type" variables that have the greatest impact on programmer productivity are the programming language and the quality of external documentation.*« [Navarro, 2006, S. 271] (Projektrahmen, teilweise modellierbar)

    Dieser Aspekt ist teilweise modellierbar. Externe Dokumentationen können als Artefakte dargestellt werden, die in einem Kontext zu den zu dokumentierenden Artefakten stehen und schon zu Beginn des Projekts vollständig vorhanden sind. Die Experimente zu Hypothese 7 zeigen den positiven Zusammenhang zwischen der Qualität der Kontextartefakte und der Produktivität.

    Der Einfluss durch die Auswahl der Programmiersprache muss über eine entsprechende Rahmenimplementierung umgesetzt werden. Dazu werden die Quantitäts- und Qualitätsmodifikatoren der entsprechenden Aktivitäten in Abhängigkeit der Spielerwahl verändert.

86. »*The average full-time employee misses 13–15 days of work per year (not counting vacation time).*« [Navarro, 2006, S. 271] (Personalmanagement, fehlend)

    Der Ausfall von Mitarbeitern ist im Simulationsmodell nicht umgesetzt, lässt sich jedoch über eine Rahmenimplementierung in Abhängigkeit der Attribute der Mitarbeiter umsetzen. Dabei sollte vor allem das Energieniveau und der Indikator für Fehlzeiten des Motivationsmodells verwendet werden, um Mitarbeiterausfälle in Form von Ereignissen zu generieren.

A »*A four-person inspection team is ideal, and is twice as effective as a three-person team. [. . .] A larger inspection team dows not necessarily equal a more productive inspection team.*« [Navarro, 2006, S. 124] (Personalmanagement, validierbar)

    Diese Aussage ist nicht Teil der Liste in [Navarro, 2006], Anhang A, sondern findet sich als Aussage zum Inspektionsmodell in Kapitel 7.2 dieser Arbeit.

    Die Experimente im Rahmen von Hypothese 18 zeigen, dass ein Inspektionsteam mit vier Mitarbeitern am effizientesten arbeitet. Die Steigerung der Arbeitsgeschwindigkeit bei einem Anstieg der Teamgröße von drei auf vier Mitarbeiter beträgt jedoch nur ca. 50%.

Zuordnung der Aussagen zu den Hypothesen

Im vorherigen Abschnitt wurde einige der in [Navarro, 2006] identifizierten Aussagen über grundlegende Aspekte des Software-Engineering auf die in Abschnitt 3.1 beschriebenen Hypothesen übertragen. Tabelle A.12 gibt einen Überblick über diese Zuordnung und zeigt, welche Aussagen durch die Überprüfung welcher Hypothesen für das Simulationsmodell belegt oder zurückgewiesen werden können.

Aussage	Hypothese																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
3													×												
4													×												
5													×												
6.1	×																								
6.2		×				×																			
7	×																								
8												×													
9										×	×														
10							×																		
11.1	×																								
11.2							×																		
12	×																								
14																			×						
15																			×	×					
16																×									
17							×																		
18							×										×								
30	×																								
33														×											
34							×																		
35										×	×														
36.2								×																	
38	×																								
39								×																	
40		×																							
43																								×	
47.1				×																					
47.4			×																						
47.6																									×
48.1								×																	
48.3	×																								
50							×																		
51	×																								
52	×																								
53	×																								
56	×						×																		
57																						×			
58	×																								
60	×																								
63																	×								
64																	×								
65		×													×										
66		×													×										
67													×												
74																							×		
76.1	×																								
76.2					×																				
77								×																	
78									×																
79													×												
80													×												
82																					×				
83																					×				
84.2	×	×																							
85							×																		
A																		×							

Tabelle A.12.: Zuordnung der Hypothesen zu den “Fundamental Rules of Software Engineering“

Abbildungsverzeichnis

2.1 Übersicht über den groben Aufbau des Simulationsmodells . . . 19
2.2 UML-Paketdiagramm zur Paketstruktur des Frameworks . . . 23
2.3 Klassenstruktur des Pakets Ausführungsumgebung . . . 24
2.4 Operationen auf Artefaktwissen . . . 28
2.5 UML-Klassendiagramm zum Aufbau des Simulationszustands . . . 29
2.6 Notation für die Struktur von Szenarien . . . 34
2.7 Struktur eines Beispielszenarios . . . 35
2.8 Verfeinerung des Artefakts Pflichtenheft (Erstveröffentlichung in [Nassal, 2017]) . . . 36
2.9 Verfeinerung der Kundenkommunikation . . . 36
2.10 Erweiterung durch manuelle Tests . . . 36
2.11 Erweiterung durch automatisierte Tests . . . 37
2.12 Notationselemente für Modelle und deren Kombination . . . 38
2.13 Glättungsfunktion . . . 39
2.14 REFA-Normkurve [Seiwert, 1999] . . . 40
2.15 Änderung der Arbeitsgeschwindigkeit in Abhängigkeit des Zeitdrucks . . . 43
2.16 Berechnung der physischen Grundleistung eines Mitarbeiters . . . 43
2.17 Berechnung der Erschöpfung . . . 45
2.18 Negative Wirkung von Arbeit auf das Energieniveau . . . 46
2.19 Berechnung der Erholung . . . 47
2.20 Verlauf des Energieniveaus während der Erholungsphase . . . 47
2.21 Berechnung des Einflusses der Fähigkeiten auf die Produktivität . . . 49
2.22 Produktivität in Abhängigkeit der vorhandenen Fähigkeiten . . . 49
2.23 Berechnung zum Fähigkeitsgewinn durch Aufgabenbearbeitung . . . 50
2.24 Verlauf des Fähigkeitsniveaus bei Lernaktivitäten mit unterschiedlichen Lernfaktoren 51
2.25 Verlauf des Fähigkeitsniveaus bei Lernaktivitäten . . . 51
2.26 Berechnung zum Fähigkeitsverlust . . . 52
2.27 Fähigkeitsverlust in Abhängigkeit der Zeit . . . 53
2.28 Berechnung des Wissenszuwachses bei der Aufgabenbearbeitung . . . 53
2.29 Berechnung des Wissenszuwachses durch das Studium eines Artefakts . . . 54
2.30 Berechnung zum Wissensverlust . . . 55
2.31 Vergleich der zufallsbasierten und intervallbasierten Wissensreduktion . . . 55
2.32 Berechnung des direkten Wissenstransfers . . . 57
2.33 Berechnung der Arbeitsleistung einer Teilaufgabe . . . 61
2.34 Anwendung der Arbeitsleistung aus Fortschrittsaktivitäten auf ein Artefakt . . . 64
2.35 Berechnung der aus einem Artefakt zu entfernenden Defekte . . . 66
2.36 Wechsel zwischen den verschiedenen Zuständen der Arbeitszufriedenheit . . . 68
2.37 Visualisierung des Zustandsraums im Motivationsmodell . . . 76
2.38 Beispiel zu einem zeitlichen Ablauf einer Aufgabe . . . 82
2.39 Schätzprozess im Rahmen der Wahrnehmung . . . 91
2.40 Verhaltensmodelle des Simulationsmodells . . . 92
2.41 Motivationsprozess nach Pritchard und Ashwood (inklusive Modifikationen) [Nassal, 2016] . . . 98
2.42 Rahmenszenario für VIE-Beispiele (Erstveröffentlichung in [Nassal, 2016]) . . . 110
2.43 UML-Klassendiagramm zum Aufbau der Modellarchitektur . . . 117

3.1 Aufteilung der Aussagen auf die beschriebenen Kategorien . . . 121
3.2 Einstufung der Aussagen bezüglich des Simulationsmodells . . . 122
3.3 Zusammenhang zwischen Fähigkeitsniveau und Produktivität . . . 132
3.4 Produktivität in Abhängigkeit des Produktwissens . . . 134
3.5 Produktivität in Abhängigkeit der Artefaktschwierigkeit . . . 135
3.6 Zusammenhang zwischen Produktivität und Projektlaufzeit . . . 136
3.7 Produktivität in Abhängigkeit der Aktivitätssympathie . . . 137
3.8 Zusammenhang zwischen Adaptionsfähigkeit und Produktivität . . . 138
3.9 Arbeitsgeschwindigkeit, Arbeitsqualität und Produktivität in Abhängigkeit von Kontextqualität und Kontextvollständigkeit . . . 139
3.10 Produktivität in Abhängigkeit der Motivation . . . 140
3.11 Motivation in Abhängigkeit der relativen Arbeitsbedingungen . . . 142
3.12 Produktivität in Abhängigkeit der Teamgröße . . . 143
3.13 Effizienz in Abhängigkeit der Teamgröße . . . 143
3.14 Produktivität in Abhängigkeit der Teamgröße (Ergebnisse der ersten 10 Sets) . . . 144
3.15 Änderung der durchschnittlichen Produktivität durch das Hinzufügen neuer Mitarbeiter (Erstveröffentlichung in [Nassal, 2017]) . . . 146
3.16 Veränderung der Projektdauer durch das Hinzufügen von neuen Mitarbeitern . . . 146
3.17 Zusammenhang zwischen Defektanzahl und Aufdeckungsgeschwindigkeit . . . 147
3.18 Zusammenhang zwischen erlerntem Produktwissen und investierter Zeit . . . 148
3.19 Rahmenszenarien zur Überprüfung von Hypothese 16 . . . 149
3.20 Vergleich Defektbehebung . . . 150
3.21 Produktivität mit und ohne kontinuierliche Qualitätssicherung . . . 151
3.22 Effizienz bei Inspektionen in Abhängigkeit der Größe des Inspektionsteams . . . 153
3.23 Produktivität in Abhängigkeit des Zeitdrucks . . . 154
3.24 Produktivität in Abhängigkeit der Wirkdauer von Zeitdruck . . . 154
3.25 Produktivität in Abhängigkeit des fehlenden Zeitdrucks . . . 155
3.26 Produktivität mit und ohne Kommunikation . . . 156
3.27 Produktivität mit und ohne Kommunikation . . . 157
3.28 Produktivität in Abhängigkeit des Artefaktumfangs . . . 158
3.29 Benötigte Zeit in Abhängigkeit der Anzahl paralleler Aufgaben . . . 159
3.30 Benötigte Zeit in Abhängigkeit Anzahl paralleler Aufgaben und der Teamgröße . . . 159
3.31 Streudiagramm über den Zusammenhang zwischen Fähigkeit und Produktivität im Rahmen der weiteren Analyse der Parameterabhängigkeiten . . . 162

4.1 Grafische Oberfläche zur Simulationsanalyse . . . 170
4.2 Detailinformationen zu Artefakten . . . 171
4.3 Visualisierung des VIE-Entscheidungsprozesses . . . 172
4.4 Visualisierung des Motivationsmodells . . . 173
4.5 Aufbau des Pflichtenhefts (Erstveröffentlichung in [Nassal, 2017]) . . . 182
4.6 Fähigkeitenprofil des Architekten (Erstveröffentlichung in [Nassal, 2017]) . . . 183
4.7 Steuerung der Simulation (Erstveröffentlichung in [Nassal, 2015]) . . . 184
4.8 Informationen zu einem Artefakt (Erstveröffentlichung in [Nassal, 2015]) . . . 184
4.9 Vorgangsplanung in Microsoft Project (Erstveröffentlichung in [Nassal, 2015]) . . . 185
4.10 Artefakte des Integrationsszenarios und deren Beziehungen (Erstveröffentlichung in [Nassal, 2017]) . . . 188
4.11 Artefakte des Inspektionsszenarios . . . 190
4.12 Veränderung des Prüflings während eines Inspektionsvorgangs (Erstveröffentlichung in [Nassal, 2017]) . . . 191
4.13 Ausschnitt aus einem Szenario zur testgetriebenen Entwicklung . . . 192
4.14 Aufteilung des Produkts zur Realisierung sich ändernder Anforderungen . . . 193
4.15 Verknüpfung zweier gegenseitig voneinander abhängiger Artefakte . . . 193
4.16 Mögliches Szenario für ein Planspiel zu Scrum . . . 194

5.1 Ausschnitt aus einem System Dynamics Modell zum Einfluss neuer Mitarbeiter auf die Produktivität [Madachy, 2007, S. 106] 204

A.1 Zustande im Zustandsraum des Motivationsmodells 223
A.2 Modellierung der Abhängigkeiten zwischen Entwurf und Implementierung 232
A.3 Produktstruktur mit Kontextabhängigkeiten 234

LITERATURVERZEICHNIS

[Abdel-Hamid, 1991] Tarek K. Abdel-Hamid und Stuart E. Madnick. *Software Project Dynamics: An Integrated Approach*. Prentice Hall, 1991. ISBN 978-0138220402.

[Abramowitz, 1974] Milton Abramowitz. *Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables*. Dover Publications Inc., 1974. ISBN 978-0486612720.

[Baker, 2005] Alex Baker, Emily Oh Navarro, und André van der Hoek. *An experimental card game for teaching software engineering processes*. In *Journal of Systems and Software*, Band 75 (1–2) Seite 3–16, Elsevier, 2005. ISSN 0164-1212.

[Balzert, 2008] Helmut Balzert. *Lehrbuch der Software-Technik: Softwaremanagement*. Spektrum, 2. Ausgabe, 2008. ISBN 978-3827411617.

[Blötz, 2008] Ulrich Blötz und Bundesinstitut für Berufsbildung. *Planspiele in der beruflichen Bildung: Auswahl, Konzepte, Lernarrangements, Erfahrungen*. Berichte zur beruflichen Bildung. Bertelsmann, 2008. ISBN 978-3763911141.

[Boehm, 1981] Barry W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, 1. Ausgabe, 1981. ISBN 978-0138221225.

[Bonta, 1979] Juan P. Bonta. *Simulation Games in Architectural Education*. In *Journal of Architectural Education*, Band 33 (1) Seite 12–18, Taylor & Francis, Ltd., Association of Collegiate Schools of Architecture, Inc., 1979. ISSN 0149-2993.

[Brooks, 1995] Frederick P. Brooks. *The Mythical Man-Month (Jubiläumsausgabe)*. Addison-Wesley Longman Publishing, 1995. ISBN 978-0201835953.

[Bruggemann, 1975] Agnes Bruggemann, Peter Groskurth, und Eberhard Ulich. *Arbeitszufriedenheit*. Schriften zur Arbeitspsychologie. Hans Huber, 1975. ISBN 978-3456801889.

[Caruso, 2004] Claire C. Caruso, Centers for Disease Control, Prevention (U.S.), und National Institute for Occupational Safety and Health. *Overtime and Extended Work Shifts: Recent Findings on Illnesses, Injuries, and Health Behaviors*. CDC Workplace Safety and Health. Centers for Disease Control and Prevention, US Department of Health and Human Services, 2004. ISBN 978-1481228220.

[Cohen, 1988] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Taylor & Francis Inc., 1988. ISBN 978-0805802832.

[Coleman, 1994] Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, und Paul Jeremaes. *Object-oriented Development: The Fusion Method*. Prentice Hall, Inc., 1994. ISBN 978-0133388237.

[Curtis, 1988] Bill Curtis, Herb Krasner, und Neil Iscoe. *A field study of the software design process for large systems*. In *Communications of the ACM*, Band 31 (11) Seite 1268–1287, ACM, November 1988. ISSN 0001-0782.

[Dawson, 2000] Ray Dawson. *Twenty dirty tricks to train software engineers.* In *Proceedings of the 22nd International Conference on Software Engineering*, ICSE '00, Seite 209–218. ACM, 2000. ISBN 1-58113-206-9.

[de Oliveira Barros, 2006] Márcio de Oliveira Barros, Alexandre R. Dantas, Gustavo O. Veronese, und Cláudia M. L. Werner. *Model-driven Game Development: Experience and Model Enhancements in Software Project Management Education.* In *Software Process: Improvement and Practice*, Band 11 (4) Seite 411–421, John Wiley & Sons, Inc., 2006. ISSN 1099-1670.

[DeMarco, 1982] Tom DeMarco. *Controlling Software Projects: Management, Measurement & Estimation.* Yourdon Press, 1982. ISBN 978-0917072321.

[DeMarco, 2002] Tom DeMarco. *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency.* Broadway Books, 2002. ISBN 0767907698.

[Drappa, 2000a] Anke Drappa. *Quantitative Modellierung von Softwareprojekten.* Berichte aus der Softwaretechnik. Shaker, 2000. ISBN 978-3826571305.

[Drappa, 2000b] Anke Drappa und Jochen Ludewig. *Simulation in Software Engineering Training.* In *Proceedings of the 22nd International Conference on Software Engineering*, ICSE '00, Seite 199–208. ACM, 2000. ISBN 1-58113-206-9.

[Dudenredaktion, 2006] Dudenredaktion. *Duden - Die deutsche Rechtschreibung. Bd. 1.* Dudenverlag, Bibliographisches Institut & F.A. Brockhaus, 24. Ausgabe, 2006. ISBN 978-3411040148.

[Ebbinghaus, 1992] Hermann Ebbinghaus. *Über das Gedächtnis. Untersuchungen zur experimentellen Psychologie.* Bibliothek Klassischer Texte. Wissenschaftliche Buchgesellschaft, 1992. ISBN 978-3534052554.

[Fähndrich, 2013] Johannes Fähndrich, Sebastian Ahrndt, und Sahin Albayrak. *Self-Explaining Agents.* In *Jurnal Teknologi (Science & Engineering)*, Band 3 (63) Seite 53–64, Penerbit UTM Press, 2013. ISSN 0127-9696.

[Faria, 1998] Anthony J. Faria. *Business Simulation Games: Current Usage Levels – An Update.* In *Simulation & Gaming*, Band 29 (3) Seite 295–308, SAGE Publishing Inc., 1998.

[Flores, 2011] Petro Flores und Yoonsik Cheon. *PWiseGen: Generating Test Cases for Pairwise Testing Using Genetic Algorithms.* In *2011 IEEE International Conference on Computer Science and Automation Engineering*, Band 2, Seite 747–752, Juni 2011. ISBN 978-1424487271.

[Forrester, 1968] Jay W. Forrester. *Principles of Systems.* Pegasus Communications, 1968. ISBN 978-1883823412.

[Forrester, 1972] Jay W. Forrester. *Grundzüge einer Systemtheorie: Ein Lehrbuch.* Gabler, 1972. ISBN 978-3409313353.

[Forrester, 1980] Jay W. Forrester und Peter M. Senge. *Tests for building confidence in system dynamics models.* In *TIMS Studies in the Management Sciences (System Dynamics)*, Band 14, Seite 209–228. North Holland Publishing Company, 1980.

[Graf, 1960] Otto Graf. *Arbeitsphysiologie.* Gabler Verlag, 1960. ISBN 978-3663004042.

[Hanakawa, 2002] Noriko Hanakawa, Ken ichi Matsumoto, und Koji Torii. *A Knowledge-Based Software Process Simulation Model*. In *Annals Software Engineering*, Band 14 (1) Seite 383–406, Kluwer Academic Publishers, 2002. ISSN 1573-7489.

[Herzberg, 1959] Frederick Herzberg, Bernard Mausner, und Barbara Bloch Snyderman. *The Motivation to Work*. Wiley, Chapman & Hall New York, London, 2. Ausgabe, 1959.

[Hunecker, 2009] Felix Hunecker. *A generic process simulation-model for educational simulations and serious games*. In *On the Horizon*, Band 17 (4) Seite 313–322, Emerald, September 2009. ISSN 1074-8121.

[Jaber, 2016] Mohamad Y. Jaber. *Learning Curves: Theory, Models, and Applications*. Industrial Innovation Series. CRC Press, 2016. ISBN 978-1439807408.

[Jain, 2006] Apurva Jain und Barry W. Boehm. *SimVBSE: Developing a Game for Value-Based Software Engineering*. In *19th Conference on Software Engineering Education & Training (CSEET'06)*, Seite 103–114. IEEE Computer Society, 2006. ISBN 0-7695-2557-1.

[Jennings, 1998] Nicholas R. Jennings und Michael J. Wooldridge (Herausgeber). *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag New York, Inc., 1998. ISBN 978-3662036785.

[Johnson-Laird, 1983] Philip N. Johnson-Laird. *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Harvard University Press, 1983. ISBN 0-674-56882-6.

[Knowles, 1984] Malcolm S. Knowles. *Andragogy in Action: Applying Modern Principles of Adult Learning*. Jossey-Bass Higher & Adult Education. Wiley, 1984. ISBN 978-0875896212.

[Kolb, 1984] David A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development*. Prentice Hall, 1984. ISBN 978-0132952613.

[Lord, 1968] Frederic M. Lord und Melvin R. Novick. *Statistical Theories of Mental Test Scores*. Addison-Wesley Educational Publisher Inc., 1968. ISBN 978-0201043105.

[Ludewig, 1992] Jochen Ludewig, Thomas Bassler, Marcus Deininger, Kurt Schneider, und Jürgen Schwille. *SESAM – Simulating Software Projects*. In *Software Engineering and Knowledge Engineering, 1992. Proceedings., Fourth International Conference on Software Engineering and Knowledge Engineering*, Seite 608–615, Juni 1992. ISBN 0-8186-2830-8.

[Ludewig, 1994] Jochen Ludewig, Marcus Deininger, Anke Drappa, Li Jinhua, Horst Lichter, Kurt Schneider, und Jürgen Schwille. *SESAM : Software-Engineering-Simulation durch animierte Modelle*. Technischer Bericht Informatik 1994/05, Universität Stuttgart, Fakultät Informatik, Deutschland, Mai 1994.

[Madachy, 1994] Raymond J. Madachy. *A Software Project Dynamics Model for Process Cost, Schedule and Risk Assessment*. Dissertation, University of Southern California, 1994.

[Madachy, 2007] Raymond J. Madachy. *Software Process Dynamics*. Wiley, 2007.

[Martin, 1994] Hans Martin. *Grundlagen der menschengerechten Arbeitsgestaltung: Handbuch für die betriebliche Praxis*. Bund Verlag, 1994. ISBN 978-3766324528.

[Maslow, 1943] Abraham H. Maslow. *A Theory of Human Motivation*. In *Psychological Review*, Band 50 (4), American Psychological Association, 1943.

[McGraw, 2006] Gary McGraw. *Software Security: Building Security In*. Addison-Wesley Professional, 1. Ausgabe, 2006. ISBN 978-0321356703.

[Michel, 1990] Christian Michel und Felix Novak. *Kleines psychologisches Wörterbuch*. Herder Taschenbuch. Herder, 1990. ISBN 978-3451086908.

[Mitchell, 2004] Mark L. Mitchell und Janina M. Jolley. *Research Design Explained*. Thomson Wadsworth, 2004. ISBN 978-0534617318.

[Myers, 2015] David G. Myers, Siegfried Hoppe-Graff, Barbara Keller, Ü.T.T Tübingen, Matthias Reiss, Laura Dörrenbächer, Katharina Eilers, Stephanie Eilers, Theresa Fehn, Tatajana Gackstatter, et al. *Psychologie*. Springer-Lehrbuch. Springer Berlin Heidelberg, 2015. ISBN 978-3642407826.

[Nassal, 2014] Alexander Nassal und Marcel Dausend. *A Board Game for Educational Purposes in Software Engineering*. In Georg Hagel und Jürgen Mottok (Herausgeber), *European Conference Software Engineering Education (ECSEE) 2014*, Seite 235–239. Shaker, November 2014. ISBN 978-3844030679.

[Nassal, 2015] Alexander Nassal. *Projektmanagement spielend lernen*. In Axel Schmolitzky und Anna S. Hauptmann (Herausgeber), *Tagungsband des 14. Workshops "Software Engineering im Unterricht der Hochschulen" (SEUH) 2015, Dresden, Deutschland, 26.–27. Februar 2015*, Band 1332 der *CEUR Workshop Proceedings*, Seite 53–64. CEUR-WS.org, 2015. ISSN 1613-0073.

[Nassal, 2016] Alexander Nassal und Matthias Tichy. *Modeling Human Behavior for Software Engineering Simulation Games*. In *Proceedings of the 5th International Workshop on Games and Software Engineering*, GAS '16, Seite 8–14. ACM, 2016. ISBN 978-1450341608.

[Nassal, 2017] Alexander Nassal und Matthias Tichy. *Ein Framework zur Erstellung von Planspielen zur Softwaretechnik*. In Bernd Bruegge und Stephan Krusche (Herausgeber), *Tagungsband des 15. Workshops "Software Engineering im Unterricht der Hochschulen" (SEUH) 2017, Hannover, Deutschland, 22.–23. Februar 2017*, Band 1790 der *CEUR Workshop Proceedings*, Seite 51–62. CEUR-WS.org, 2017. ISSN 1613-0073.

[Navarro, 2006] Emily Navarro. *SimSE: A Software Engineering Simulation Environment for Software Process Education*. Dissertation, University of California, Irvine, 2006. ISBN 978-0542997716.

[Naylor, 1980] James C. Naylor, Robert D. Pritchard, und Daniel R. Ilgen. *A Theory of Behavior in Organizations*. Academic Press, 1980. ISBN 978-0125144506.

[Neuberger, 1974] Oswald Neuberger. *Theorien der Arbeitszufriedenheit*. Nummer 507 in Urban-Taschenbücher. Kohlhammer, 1974. ISBN 978-3170020955.

[Pieper, 2013] Jöran Pieper. *Alles nur Spielerei? Neue Ansätze für digitales spielbasiertes Lernen von Softwareprozessen*. In Andreas Spillner und Horst Lichter (Herausgeber), *Tagungsband des 13. Workshops "Software Engineering im Unterricht der Hochschulen" (SEUH) 2013, Aachen*, Band 956 der *CEUR Workshop Proceedings*, Seite 131–139. CEUR-WS.org, 2013. ISSN 1613-0073.

[Prammer, 2012] Elisabeth Prammer. *Boreout - Biografien der Unterforderung und Langeweile: Eine soziologische Analyse*. Springer Fachmedien Wiesbaden, 2012. ISBN 978-3658005030.

[Prensky, 2007] Marc Prensky und Sivasailam Thiagarajan. *Digital Game-Based Learning*. Paragon House Publishers, 2007. ISBN 978-1557788634.

[Pressman, 1997] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., 4. Ausgabe, 1997. ISBN 978-0070521827.

[Pritchard, 2008] Robert D. Pritchard und Elissa L. Ashwood. *Managing Motivation: A Manager's Guide to Diagnosing and Improving Motivation*. Taylor & Francis, 2008. ISBN 978-0203888346.

[Reich, 2008] Kersten Reich. *Konstruktivistische Didaktik: Lehr- und Studienbuch mit Methodenpool*. Beltz Pädagogik. Beltz, 2008. ISBN 978-3407254924.

[Rohmert, 1983] Walter Rohmert, Joseph Rutenfranz, et al. *Praktische Arbeitsphysiologie*. Georg Thieme Verlag, 1983. ISBN 13: 9783133701037.

[Röhner, 2012] Jessica Röhner und Astrid Schütz. *Psychologie der Kommunikation*. Basiswissen Psychologie. Springer VS, 2012. ISBN 978-3531167077.

[Schneider, 1994] Kurt Schneider. *Ausführbare Modelle der Software-Entwicklung: Struktur und Realisierung eines Simulationssystems*. Dissertation, Universität Stuttgart, 1994. ISBN 978-3728121486.

[Schuler, 2006] Heinz Schuler. *Lehrbuch der Personalpsychologie*. Hogrefe, 2006. ISBN 978-3801719340.

[Seiwert, 1999] Lothar J. Seiwert. *30 Minuten für optimales Zeitmanagement*. GABAL Verlag GmbH, 1999. ISBN 978-3930799862.

[Shannon, 1963] Claude E. Shannon und Warren Weaver. *The Mathematical Theory of Communication*. Nummer 11 in Illini Books. University of Illinois Press, 1963. ISBN 978-0252725487.

[Sharp, 2000] Helen Sharp und Pat Hall. *An Interactive Multimedia Software House Simulation for Postgraduate Software Engineers*. In Carlo Ghezzi, Mehdi Jazayeri, und Alexander L. Wolf (Herausgeber), *Journal of Interactive Media in Education*, Seite 688–691. ACM, 2000. ISBN 1-58113-206-9.

[Shaw, 2005] Katherine Shaw und Julian Dermoudy. *Engendering an empathy for software engineering*. In *Proceedings of the 7th Australasian Conference on Computing Education*, Band 42, Seite 135–144. Australian Computer Society, Inc., 2005. ISBN 1-920682-24-4.

[Sommerville, 2007] Ian Sommerville. *Software Engineering*. Pearson Studium, 8. Ausgabe, 2007. ISBN 978-3827372574.

[Stachowiak, 1973] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, 1973. ISBN 978-3211811061.

[Taran, 2007] Gil Taran. *Using Games in Software Engineering Education to Teach Risk Management*. In *20th Conference on Software Engineering Education Training (CSEET'07)*, Seite 211–220. IEEE Computer Society, 2007. ISBN 0-7695-2893-7. ISSN 1093-0175.

[Tillmann, 2013] Nikolai Tillmann, Jonathan De Halleux, Tao Xie, Sumit Gulwani, und Judith Bishop. *Teaching and Learning Programming and Software Engineering via Interactive Gaming.* In *Proceedings of the 2013 International Conference on Software Engineering*, Seite 1117–1126. IEEE Press, Mai 2013. ISBN 978-1467330763.

[Tvedt, 1996] John D. Tvedt. *An Extensible Model for Evaluating the Impact of Process Improvements on Software Development Cycle Time.* Dissertation, Arizona State University, 1996.

[Urban, 2000] Christoph Urban. PECS: A Reference Model for the Simulation of Multi-Agent Systems. In Ramzi Suleiman, Klaus G. Troitzsch, und Nigel Gilbert (Herausgeber), *Tools and Techniques for Social Science Simulation*, Seite 83–114. Physica-Verlag HD, 2000. ISBN 978-3790812657.

[Urban, 2005] Christoph Urban. *Das Referenzmodell PECS: agentenbasierte Modellierung menschlichen Handelns, Entscheidens und Verhaltens.* Dissertation, Universität Passau, 2005.

[Vroom, 1994] Victor H. Vroom. *Work and Motivation.* Jossey-Bass, 1. Ausgabe, 1994. ISBN 978-0787900304.

[Weller, 1993] Edward F. Weller. *Lessons from three years of inspection data.* In *IEEE Software*, Band 10 (5) Seite 38–45, IEEE Computer Society Press, September 1993. ISSN 0740-7459.

[Winter, 2013] Eggert Winter (Herausgeber). *Gabler Wirtschaftslexikon.* Gabler, 18. Ausgabe, 2013. ISBN 978-3834934642.

[Ye, 2007] En Ye, Chang Liu, und Jennifer A. Polack-Wahl. *Enhancing Software Engineering Education Using Teaching Aids in 3-D Online Virtual Worlds.* In *37th Annual Frontiers In Education Conference – Global Engineering: Knowledge Without Borders, Opportunities Without Passports.* IEEE, 2007. ISBN 978-1424410835. ISSN 0190-5848.

Index

Abbildcharakter, 20
Abbildungsmerkmal, 20
Adaptionsfähigkeit, 54, 57, 219
Aktivität, 30, 32, 34, 116, 171
    Fortschritts-, 32, 116, 217
    Kommunikations-, 32, 116, 217
    Qualitätssicherungs-, 32, 116, 217
Aktivitätssympathie, 220
Aktivitätstyp, 34
Anerkennung, 107
Anspruch, 220
Arbeitsbedingungen, 72
Arbeitseinheit, 26, 217
Arbeitsfortschritt, 101
Arbeitsgeschwindigkeit, 26
Arbeitsinhalte, 74, 86
Arbeitsleistung, 60, 67
Arbeitsmarkt, 210
Arbeitsplatz betreten, 79
Arbeitsplatz verlassen, 79
Arbeitsumfang, 26
Arbeitsumgebung, 120, 209
Arbeitsunzufriedenheit, 67
Arbeitszeit, 79
Arbeitszeitkonto, 72
Arbeitszeitplanung
    direkte, 93
    dynamische, 94
Arbeitszufriedenheit, 67
Artefakt, 30, 34
Artefaktbearbeitung, 37, 59
Artefakttyp, 30, 32, 34, 171, 203, 216
Artefaktwissen, 27, 48, 53, 218
Aufgabe, 33, 34
Aufgabenschwierigkeit, 27
Aufgabenwechsel, 37, 77
Ausdauer, 219
Ausführungsumgebung, 24

Bedürfnis, 108
Bedürfnisbefriedigung, 98
Bedürfnisvektor, 220
Beförderung, 107
Begutachtung, 130
besonderer Einfluss, 31
Beziehung, 106
Boreout, 74

Charakter, 33
Chaturanga, 6

Datenmodell, 26
Defekt, 27, 31, 216
Defekttyp, 33, 217
deskriptives Modell, 20
Differenz, 27
Disziplin, 30, 34, 171, 216

Eclipse Process Framework, 28
Effekt, 13
effektorientierte Modellierung, 205
Einfluss zusätzlichen Wissens, 31, 60
Elementarmodell, 19
Emotion, 33
Encoder-/Decoder-Modell, 56
Energieniveau, 39, 40, 218
Entwicklungsaktivität, 120
Entwicklungsprozess, 120
Ereignis, 22
Erfahrungsbasiertes Lernen, 5
Erfolgserlebnisse, 73
Ergebnis, 98, 100
Erholung, 40, 44, 79, 80
Erschöpfung, 40, 44
Evaluation, 24, 98, 173
Experiment, 174

Fähigkeit, 26, 48, 50, 60, 103, 218
    unmittelbare, 52
feedback loops, 22
Fehlzeiten, 67
Fernplanspielwettbewerb, 7
Finanzen, 109
Folge, 98, 106
Fortschritt, 31, 216
    relativer, 31, 216
Framework, 23
Freiraum, 106
Freizeit, 79, 108
Function Points, 26
Fusion, 180

Gamification, 6
Gantt-Diagramm, 172
Gehalt, 70, 218
Geschwindigkeit, 26

grafische Oberfläche, 24, 170
Gruppenplanspiel, 7

Handlung, 98
Hygienefaktor, 68, 220

Individualplanspiel, 7
Inspektion, 116

Kündigung, 76
Kognition, 33
kognitive Modelle, 37
Kommunikation, 37
    einfache, 56
    erweiterte, 58
    Konferenz-, 58
Kommunikationsmodell, 56
    Sender-Empfänger, 56
    Shannon/Weaver, 56
Komplement, 27
Kontextartefakt, 31, 216
Kontextartefakttyp, 217
Kontexteinfluss, 60
Kontingenz, 98
Korrelationsanalyse, 130
krankheitsbedingte Ausfälle, 76
Kunde, 209

Langzeitgedächtnis, 52, 218
Lehre, 7
Leistung, 73, 76
Leistung im Tagesverlauf, 39
Leistung und Erfolgserlebnisse, 86
Leistungseinfluss, 218
Leistungsmodell, 37
Lernen, 50
Lernfaktor, 50, 219
Lernfortschritt, 103
Lernziele, 175
Lines of Code, 26
Lohn, 70, 107

Makroperformanz, 43
mentales Modell, 89
Microsoft Project, 180, 181
Mikroperformanz, 43
Mitarbeiter, 33, 172
    aktiver, 81
    passiver, 81
Mitarbeitersympathie, 220
MO-SEProcess, 11
Modell, 20
    qualitativ, 22
    quantitativ, 22
Modellstruktur, 124
Modellumgebung, 22
Modellverhalten, 125
Monitor, 24, 174
Motivation, 37, 67, 218
    extrinsische, 67
    intrinsische, 67, 105, 107
Motivationsmodell, 173
Motivator, 68, 220

Notation, 34

Operationale Semantik, 19
Original, 21
OSS, 9

Paarprogrammierung, 82
Parameter, 160
    Ausgabe-, 129
    Delta-, 130
    dynamischer, 130
    Eingabe-, 129
persönliches Wachstum, 72, 86, 107
Persistierung, 24, 169
Personalfluktuation, 67
Personalmanagement, 120
Pex4Fun, 8
Physis, 33, 37
physische Ausdauer, 45
Planspiel, 6, 7
Pragmatisches Merkmal, 21
Produkt, 30, 171, 176
Projektbeteiligter, 30, 33, 34, 172
    durchschnittlicher, 26
Projektplan, 33, 102, 172
Projektrahmen, 120
Prozessmodell, 30, 171, 176
Prozessmodellierung, 202

Qualität, 27, 32, 217
Qualitätsänderung, 101
Qualitätsmodifikator, 33, 64, 217
Quantitätsmodifikator, 33, 64, 217

R, 166
Realzeit, 7
Rednerfähigkeit, 57, 219
REFA-Normkurve, 40
Regelimplikation, 126
Retrospektive, 196
Risikomanagement, 210
Rolle, 32, 34, 171, 203, 217, 218

S-Kurve, 51
Schachspiel, 6
Schnittmenge, 27
Schrittweite, 209
Schwierigkeit, 27, 30, 49, 60, 217

Scrum, 194
Second Life, 11
SESAM, 9, 12, 205
Sicherheit, 70, 107, 109
SimjavaSP, 9
SimSE, 10, 12, 205, 206
Simulation, 24
    ereignisorientierte, 22
    Multi-Agenten-, 23
Simulationskern, 24
Simulationsmodell, 20
SimVBSE, 10
sozialer Status, 33
Spaß, 108
Spielmodus, 176
spontane Kommunikation, 87, 93
Störung, 104
Statische Struktur, 19, 26, 28
Status, 108
System Dynamics, 22, 203
Systemgrenze, 21
Systemumgebung, 21
Szenario, 7, 28
    Ausgangs-, 28

Tätigkeit, 79
Tätigkeitsmodell, 19
Tagesleistungskurve, 40, 219
Tagesverlauf, 40
Team, 30, 176
Test, 123
Testen, 116
testgetriebene Entwicklung, 192
The Incredible Manager, 11

Umfang, 27, 30, 217
unproduktive Zeit, 104
User Story, 194

Validität, 165
    externe, 167
    interne, 167
    Konstrukt-, 165
    statistische, 166
Verantwortung, 109
verbreitetes Wissen, 104
Vereinigung, 27
Vergessen, 55
Vergessensrate, 52, 220
Vergleich, 131
Verhaltensmodell, 19, 88
Verkürzungsmerkmal, 21
Verlernen, 52
VIE, 96
Vollständigkeit, 31, 216
Vorbedingung, 31, 217
Vorbildcharakter, 20

Wahrnehmung
    Selbst-, 91
    Umgebungs-, 90
Wechselfähigkeit, 219
Weiterentwicklung, 109

XML-Format, 169
XML-Schema, 169

Zeitablauf, 176
Zeitdruck, 41, 72
Zeitdrucklimit, 42
Zeitdruckschranke, 219
Zielartefakttyp, 217
zusätzliches Wissen, 62, 216
Zwei-Faktoren-Theorie, 68
zwischenmenschliche Beziehung, 71, 86