ulm university universität
uulm

Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notation

Stephan Buchwald, Thomas Bauer, Manfred Reichert

Ulmer Informatik-Berichte

Nr. 2009-12
Dezember 2009

Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

Stephan Buchwald[1], Thomas Bauer[1], Manfred Reichert[2]

[1]Abteilung für Daten- und Prozessmanagement, Daimler AG,
{stephan.buchwald, thomas.tb.bauer}@daimler.com
[2]Institut für Datenbanken und Informationssysteme, Universität Ulm
manfred.reichert@uni-ulm.de

Abstract: Häufig genannte Ziele einer Service-orientierten Architektur (SOA) sind die bessere Unterstützung und Anpassbarkeit von Geschäftsprozessen sowie das Business-IT-Alignment. Diese Ziele werden heute nicht erreicht, da die bei der Implementierung eines Fachprozesses notwendigen komplexen Transformationen in einen ausführbaren Workflow schwer nachvollziehbar sind. Dadurch gehen fachliche Anforderungen verloren und es entsteht ein hoher Aufwand bei späteren Prozessanpassungen. Dieser Beitrag führt die Ebene des Systemmodells zwischen Fachbereich (Fachmodelle) und IT-Bereich (ausführbare Modelle) ein, um Geschäftsprozess-Transformationen besser zu unterstützen und den Ergebnisprozess mit den Fachbereichen abstimmen zu können. Im vorgestellten Ansatz wird ein sog. *Abbildungsmodell* eingeführt, das Zugehörigkeiten von Aktivitäten des Fachmodells zu denen des Systemmodells (d.h. technische Spezifikation des Informationssystems) explizit dokumentiert. Dadurch werden im Software-Entwicklungsprozess automatisierte Konsistenzprüfungen zwischen den Modellebenen möglich. Werden später Prozessanpassungen erforderlich, lassen sich die zu einer fachlichen Aktivität gehörenden technischen Aktivitäten unmittelbar erkennen, was die Durchführung der Anpassung erleichtert. Ein wesentlicher Vorteil unseres Ansatzes besteht darin, dass die Erstellung des Abbildungsmodells nur einen minimalen Aufwand verursacht, da keine komplexen Regeln, sondern nur einfache Beziehungen definiert werden müssen. Der Ansatz ist damit in der Praxis gut verwendbar, was durch eine prototypische Umsetzung unterstrichen wird.

1. Motivation

Service-orientierte Architekturen (SOA) [Erl05, Jos07, BBP09] sind ein aktuelles IT-Thema in Unternehmen. Neben technischen Aspekten ist die Erhöhung der Flexibilität eines der wichtigsten Ziele einer SOA. Gemeint ist damit, dass fachliche Anforderungen schneller als bisher in entsprechende IT-Applikationen umgesetzt werden können. In diesem Zusammenhang spielen Geschäftsprozesse und ihre IT-Unterstützung eine zentrale Rolle. Hier besteht ein hoher Anpassungsbedarf, da Prozessänderungen (z.B. zur Geschäftsoptimierung oder zwecks Anpassungen der Prozesse an gesetzliche Änderungen) häufiger notwendig werden als Anpassungen feingranularer und wiederverwendbarer Geschäftsfunktionen. Wir verfolgen im Projekt *Enhanced Process Management by Service Orientation (ENPROSO)* das Ziel, eine SOA so zu gestalten, dass Geschäftsprozesse zukünftig leichter anpassbar sind als bei heutigen Architekturen.

Eine bessere Anpassungsfähigkeit von Geschäftsprozessen [Sch98, Wes07] an notwendige Änderungen lässt sich, zumindest vom Prinzip her, durch Verwendung von *Workflow-Technologie* erreichen [Wes07, RD00]. Durch Trennung der verschiedenen Workflow-Aspekte [Jab97], wie Kontroll- und Datenfluss, von der Implementierung der Anwendungslogik, entsteht ein besser strukturiertes prozessorientiertes Informationssystem.

Allerdings genügt der Einsatz von Workflow-Technologie alleine nicht, um die Anforderung einer SOA an *Flexibilität* zu erfüllen. Gegenüber der heutigen Praxis ist insbesondere eine Verbesserung des Zusammenspiels zwischen Fachbereichen und IT-Abteilungen bei der Software-Entwicklung erforderlich. Dieser Aspekt wird auch als *Business-IT-Alignment* bezeichnet [Che08]: Informationssysteme sollen die fachlichen Anforderungen und Bedürfnisse exakter treffen als dies bisher der Fall gewesen ist. Neben einer konsequenten Prozessorientierung ist es einerseits erforderlich, dass fachliche Anforderungen und Fachprozesse vollständig dokumentiert werden. Andererseits müssen Informationsverlust und Verfälschungen bei der Entwicklung des (prozessorientierten) Informationssystems reduziert werden. Falls Änderungen an den fachlichen Anforderungen erfolgen, sollen diese korrekt und unverfälscht in die Implementierung des Informationssystems einfließen. Dies soll zudem möglichst schnell geschehen. Außerdem kann es in einer SOA zu Änderungen in der Umgebung kommen, z.B. wenn Services wegfallen (d.h.

abgeschaltet werden) oder wenn diese zu einer neuen Version migrieren. Auch auf solche Szenarien muss flexibel reagiert werden können.

Um die Lücke zwischen Fachbereich/Fachmodell und IT-Abteilung/ausführbarem Modell (Software-Realisierung) zu schließen, ist eine zusätzliche Modellebene notwendig, welche die Beziehungen zwischen fachlichen Anforderungen und entsprechender IT-Implementierung nachvollziehbar dokumentiert. Es existieren einige Ansätze, die die Verwendung einer solchen Modellebene vorsehen, wie z.B. MID M3 [PR05], IBM SOMA [AGA+08, Ars04], IDS Scheer AVE [Dee07, BEH+08] und Quasar Enterprise [EV08]. Wir verwenden hierfür ein Zwischenmodell, welches im Folgenden *Systemmodell* genannt wird.

In dieses Systemmodell müssen fachliche Anforderungen an einen Prozess einfließen. Nur dann werden sie in der tatsächlichen Implementierung des Informationssystems umgesetzt. Hierzu ist ein Konzept erforderlich, bei dem die Beziehung zwischen fachlichen Anforderungen und ihrer Implementierung im geplanten Informationssystem nachvollziehbar wird. So muss für jede Aktivität des fachlichen Modells (und damit für deren Eigenschaften und Anforderungen) ableitbar sein, in welche Aktivität(en) des Systemmodells sie eingeflossen ist (vgl. Abbildung 1). Dies ist normalerweise nicht ohne weiteres erkennbar, da für IT-Implementierungen meist Vorgaben für Aktivitätenbezeichnungen bestehen, die nicht mit denen der Fachabteilungen übereinstimmen.

Ein Beispiel einer einfachen Transformation zeigt Abbildung 1 mit der Umbenennung der fachlichen Aktivität `[Änderungsantrag stellen]` in `[HT_..._RequestChange]` im Systemmodell. Während solche Namensunterschiede noch einfach handhabbar sind, werden bei der Abbildung eines Fachmodells auf ein ausführbares Modell meist größere Umstrukturierungen notwendig. Grund dafür ist, dass ein Fachbereich seine Geschäftsprozesse im Regelfall auf einer anderen Abstraktionsebene beschreibt, als sie für eine IT-Spezifikation benötigt wird. So werden in einem Fachprozessmodell oftmals manuelle Einzelaktivitäten beschrieben, die nicht automatisiert werden sollen, aber für Verfahrenshandbücher oder Prozesskostenrechnungen dokumentiert werden sollen. Diese werden in einer IT-Spezifikation nicht 1:1 übernommen, sondern zusammengefasst oder gar ganz weggelassen. Ein Beispiel ist die Aktivität `[Änderung umsetzen]` in Abbildung 1. IT-unterstützte Prozessschritte hingegen sind im Fachprozess häufig nur grob beschrieben, so dass sie verfeinert oder sogar zusätzlich hinzugefügt werden müssen. Andere fachliche Aktivitäten werden in mehrere IT-gestützte Aktivitäten (Benutzerinteraktionen, Service-Aufrufe und Datentransformationen) aufgespaltet. So wird in unserem Beispiel die Aktivität `[betroffene Bauteile angeben]` in eine Aktivität mit Benutzerinteraktion (Human Task) und einen Service-Aufruf aufgespalten. Schließlich ist zu beobachten, dass Fehler- und Ausnahmefälle (z.B. Rücksprünge bei unvollständigen Daten) in Fachprozessmodellen meist nicht abgebildet werden, um die Komplexität in dieser frühen Phase gering zu halten.

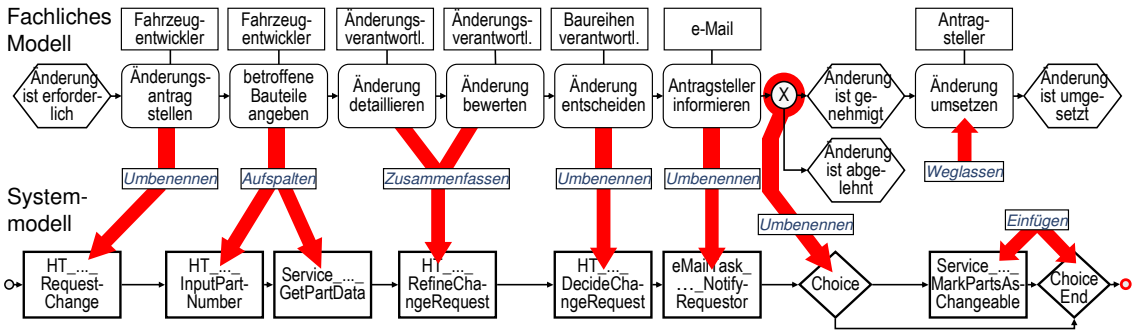


Abbildung 1: Transformationen zwischen fachlichem Modell und Systemmodell

Unter Berücksichtigung solcher Transformationen ermöglicht unser ENPROSO-Ansatz, die Nachvollziehbarkeit der Beziehungen zwischen fachlicher Aktivität und ihrer IT-Implementierung. Ebenso wird Nachvollziehbarkeit in umgekehrter Richtung unterstützt: So ist es für die robuste Ausführung einer SOA-Anwendung wichtig, die von Umgebungsänderungen betroffenen Prozesse und Aktivitäten identifizieren zu können. Nur so wird es möglich, entsprechend schnell auf anstehende Änderungen wie "Service-Abschaltungen" reagieren zu können.

In Kapitel 2 stellen wir die Anforderungen an das zu entwickelnde Konzept vor, insbesondere hinsichtlich der zu unterstützenden Prozesstransformationen. Kapitel 3 skizziert das Abbildungsmodell unseres

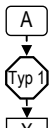
ENPROSO-Ansatz und beschreibt die dazu notwendigen Prozesstransformationen. Anschließend wird der Ansatz in Kapitel 4 detailliert. Außerdem wird diskutiert, welche Notationen und Prozess-Modellierungswerkzeuge zu seiner Realisierung geeignet sind. Ferner werden in Kapitel 5 für mehrere ausgewählte Notationen exemplarische Umsetzungen für ein Prozessbeispiel vorgestellt. Kapitel 6 diskutiert verwandte Ansätze, bevor mit einer Zusammenfassung und einem Ausblick geschlossen wird.

2. Rahmenbedingungen und Anforderungen

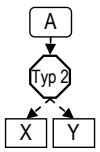
Die Modellierung von Geschäftsprozessen aus fachlicher Sicht dient, neben der Prozessanalyse und Prozessoptimierung, vor allem der Dokumentation fachlicher Anforderungen seitens der Endanwender und Prozessverantwortlichen. Da letztere meist wenig oder keinen IT-Hintergrund haben, werden die Inhalte eines *fachlichen Modells* nicht formal beschreiben. Stattdessen werden einfache graphische Notationen und textuelle Beschreibungen verwendet, wie sie von Geschäftsprozess-Modellierungswerkzeugen angeboten werden (z.B. erweiterte ereignisgesteuerte Prozessketten (eEPK) in ARIS [Sch98]). Wie erwähnt, sind in dieser frühen Phase noch nicht alle Aspekte im Detail bekannt bzw. sollen aus Komplexitätsgründen noch nicht modelliert werden, so dass das fachliche Prozessmodell (kurz: Fachprozess) an bestimmten Stellen bewusst vage gehalten wird. Diese Unvollständigkeit betrifft die Prozessstruktur (d.h. den Kontrollfluss) selbst, aber auch andere Aspekte (z.B. erfolgt im Fachprozess noch keine detaillierte Festlegung von Datenstrukturen). Hingegen muss ein implementierter Workflow von einer Workflow-Engine ausgeführt werden, weshalb die entsprechende technische Prozessbeschreibung (*ausführbares Modell*) vollständig und formal sein muss. Außerdem muss sie den Vorgaben des von der Workflow-Engine verwendeten Metamodells genügen (z.B. BPEL [OAS07], BPMN [OMG09]). Welches Metamodell jeweils relevant ist, hängt von der gewählten Ausführungsumgebung ab. So verwendet der IBM WebSphere Process Server (WPS) eine Erweiterung von BPEL, die sich in der Version 6.2 stark an BPMN orientiert [IBM09].

Bei der *Transformation* eines fachlichen Prozessmodells in ein Systemmodell ist es wichtig, dass das Systemmodell geeignet angepasst (d.h. umstrukturiert) wird. Da die durchgängige Realisierung solcher Umstrukturierungen den Schwerpunkt dieses Beitrags bildet, werden die verschiedenen Typen von Strukturänderungen entlang des in Abbildung 1 dargestellten (stark vereinfachten) Antragsprozess für Produktänderungen in der Automobilindustrie motiviert. Dieser stellt sicher, dass Änderungsvorhaben an Bauteilen vor ihrer eigentlichen Umsetzung bewertet, genehmigt und entsprechend dokumentiert werden. Ein Änderungsvorhaben wird angelegt, indem die Änderung in Form eines Antrages beschrieben wird. Da sich Änderungen meist auf mehrere Bauteile auswirken, müssen Informationen über die von ihnen betroffenen Bauteile eingeholt werden. Anschießend wird die Änderung detailliert und bewertet. Abhängig von dieser Bewertung wird entschieden, ob das Änderungsvorhaben umgesetzt wird oder nicht.

Im Folgenden werden Transformationstypen beschrieben, die in der Praxis häufig vorkommen.

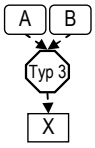


Typ 1 (Übernahme einer Aktivität inkl. Umbenennung): Im einfachsten Fall wird eine Aktivität des Fachprozesses auf genau eine Aktivität des Systemmodells abgebildet. So kann eine Benutzerinteraktion zum Ausfüllen eines Formulars z.B. als *Human Task* [Agr07] in einem BPEL-Prozess realisiert werden. Da für Aktivitäten auf IT-Ebene aber häufig Namenskonventionen existieren (z.B. um die zugehörige Applikation im IT-Betrieb erkennen zu können), ergeben sich meist unterschiedliche Namen für Aktivitäten und Daten im fachlichen Modell und im Systemmodell. Zwecks Nachvollziehbarkeit müssen wir solche Anpassungen explizit verwalten. Beispielsweise wird die fachliche Aktivität `[Änderungsantrag stellen]` durch die Human Task `[HT_..._ChangeAppl_-ProductDevelopment_RequestChange]` im Systemmodell realisiert (vgl. Abbildung 1).

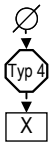


Typ 2 (Aufspalten einer Aktivität): Service-orientierte Workflow-Engines erfordern eine strikte Unterscheidung von Aktivitäten mit Benutzerinteraktion (Human Tasks) und Service-Aufrufen (BPEL Invoke). Diese Aufspaltung ist eine Neuerung bei Service-orientierten Workflow-Engines. Klassische Workflow-Engines, wie IBM WebSphere MQ Workflow [IBM05], haben Aktivitäten als größere Einheiten betrachtet, die sowohl mit dem Benutzer interagieren als auch Daten mit Backend-Systemen austauschen. Da solche Einheiten aber kaum wiederverwendbar sind, entsprechen sie nicht der Grundphilosophie einer SOA. Im Beispiel aus Abbildung 1 ist es deshalb notwendig, die fachlich zusammengehörige Aktivität `[betroffene Bauteile angeben]` in eine Benutzerinteraktion zur Eingabe der Bauteilnummern und einen Service-Aufruf zur Ermittlung der restlichen Bauteildaten aus einem Produktdaten-Management (PDM)-System aufzuspalten. Es gibt durchaus auch Fälle, in denen mehr als ein Serviceaufruf entsteht, z.B. wenn vor der Benutzerinteraktion mittels eines Services anzuzeigende Daten beschafft werden müssen, oder Daten mittels mehrerer

Services aus unterschiedlichen Backend-Systemen gelesen oder geschrieben werden müssen.



Typ 3 (Zusammenfassen von Aktivitäten): Bei der fachlichen Analyse von Geschäftsprozessen werden logisch zusammengehörige Aufgaben ermittelt, die mittels separaten Aktivitäten modelliert werden. Wenn solche Aktivitäten z.B. unmittelbar aufeinander folgend stets von derselben Person erledigt werden, kann es Sinn machen, diese im Systemmodell zu einer Aktivität zusammenzufassen, um sie dann durch eine Sequenz von Bildschirmmasken zu bearbeiten. Im dargestellten Beispiel wird es dem Änderungsverantwortlichen dadurch erspart, nach Beendigung der Detaillierung, dieselbe Workflow-Instanz in seiner Arbeitsliste zu suchen, um anschließend die Bewertung der Änderung durchzuführen.



Typ 4 (Einfügen zusätzlicher Aktivitäten in das Systemmodell): Nachdem der Baureihenverantwortliche eine Entscheidung über die Änderung getroffen hat und der Antragsteller entsprechend informiert ist, kann die Änderung durchgeführt werden. Damit dies im PDM-System tatsächlich möglich ist, müssen dort die entsprechenden Bauteile in den Zustand `Veränderbar` versetzt werden. Dies erfolgt durch den Serviceaufruf `[Service_MarkPartsAsChangable]`, der deshalb zusätzlich in das Systemmodell über eine entsprechende Abbildung eingefügt wird. Oftmals sind auch zusätzliche Aktivitäten zur Protokollierung relevanter Aktionen oder eingetretener Fehler auf Workflow-Ebene erforderlich. Das Einfügen zusätzlicher Aktivitäten wird auch deshalb erforderlich, um Fehlererkennungen und Behandlungen durchführen zu können, die fachlich noch nicht modelliert werden sollen (z.B. automatische Aktivität zur Sicherstellung der Datenqualität). Kann ein Fehler beispielsweise nur behoben werden, indem an eine frühere Stelle im Prozess zurückgesprungen wird (so dass Teile wiederholt werden), werden auch noch zusätzliche Knoten und die zugehörigen Kanten in den Prozess eingefügt.

Typ 5 (Weglassen von Aktivitäten aus dem fachlichen Modell): In Fachprozessen können sich häufig Aktivitäten befinden, deren Ausführung nicht vom Workflow-System gesteuert bzw. überwacht werden soll. So erfolgt in unserem Beispiel die Umsetzung einer Änderung selbstständig durch den Antragssteller, nachdem dieser informiert wurde, dass sein Antrag genehmigt wurde. Die Modellierung dieser Aktivität im Fachprozess ist zwar sinnvoll, da sie nicht unerheblich zu den Prozesskosten und -durchlaufzeiten beiträgt (sie fließt z.B. bei einer Prozess-Simulation ein), jedoch für die Workflow-Ebene nicht relevant. Ähnliche Szenarien gibt es für Aktivitäten die das Begrüßen eines Kunden oder das Führen eines Verkaufsgesprächs realisieren.

Zusätzlich können bei Bedarf weitere Transformationstypen für Aktivitäten definiert werden, etwa die Transformation von *m* Aktivitäten des Fachmodells in *n* Aktivitäten des Systemmodells.

Die Beispiele zeigen, dass es im Nachhinein keineswegs trivial ist, die Beziehungen zwischen fachlichen Aktivitäten und ihrer IT-Realisierung zu erkennen. Deshalb müssen die durchgeführten Abbildungen und Transformationen explizit gemacht und aufbewahrt werden – wie dies realisiert werden kann wird in den folgenden Kapiteln beschrieben. Dadurch wird die Nachvollziehbarkeit verbessert, so dass (z.B. in Projekt-Reviews) aufgezeigt werden kann, wie die fachlichen Anforderungen tatsächlich umgesetzt wurden.

3. Konzept für die Prozesstransformation

Wir entwickeln nun ein grundlegendes Konzept, um ausgehend von einem fachlichen Prozessmodell die erwähnten Transformationen in einem Systemmodell durchführen zu können. Hierzu wird eine mehrstufige Vorgehensweise vorgestellt: Zuerst wird der Fachprozess in einen korrespondierenden Prozess auf Systemmodellebene (Systemprozess) transformiert. Aus diesem wird dann das ausführbare Prozessmodell abgeleitet. Außerdem werden Möglichkeiten aufgezeigt, wie Beziehungen zwischen diesen Ebenen in Prozessmodellierungsumgebungen explizit dokumentiert werden können. Dieser Aspekt wird in Kapitel 4 detailliert.

3.1. Ebenen der Prozessmodellierung

Der Fachprozess dient der Dokumentation von fachlichen Anforderungen an das zu realisierende Informationssystem. Fachliche Anforderungen werden meist durch Befragungen von Endanwendern und Prozessverantwortlichen bzw. -eignern ermittelt. Häufig wird diesen Personen der Fachprozess graphisch angezeigt, so dass sie ihre eigenen Prozessschritte in den Fachprozess einordnen oder Fehler erkennen können. Deshalb ist die wichtigste Anforderung an ein fachliches Modell, dass es leicht verständlich ist. Zudem liegt die Verantwortung für die Erstellung des *Fachmodells* meist bei den jeweiligen Fachbereichen, auch wenn die operative Umsetzung dieser Aufgabe oftmals durch (externe) Berater erfolgt.

Für die Inhalte sind die Fachbereiche verantwortlich, da nur diese das entsprechende Fachwissen haben. Bei der Modellierung werden primär die Struktur des Prozessablaufs (Kontrollfluss) mit seinen Aktivitäten, deren Ein- und Ausgabedaten sowie zuständigen Bearbeitern festgelegt.

Die Verantwortung für die Erstellung des *Systemmodells* liegt beim IT-Bereich. Durchgeführte Änderungen am Systemmodell sollten vom zuständigen Fachbereich bestätigt werden. Deshalb muss die Darstellung des Systemmodells (z.B. graphische Notation, Struktur) so gewählt werden, dass es für Fachanwender verständlich ist. Die zugehörigen Inhalte sind dieselben wie im fachlichen Modell. Allerdings müssen diese ausreichend detailliert, vollständig und formal sein, um die Basis einer plattformunabhängigen IT-Spezifikation bilden zu können. Das bedeutet, vage textuelle Beschreibungen und offene Aspekte aus dem Fachmodell müssen ersetzt worden sein.

Die Software-Realisierung (*ausführbares Modell*) hingegen wird von IT-Implementieren erstellt. Diese treffen keine (fachlich relevanten) Entscheidungen bei der Erstellung des ausführbaren Modells, sondern übernehmen stattdessen die Struktur und Inhalte des Systemmodells 1:1. Für das ausführbare Modell sind zahlreiche weitere zielplattformabhängige Informationen notwendig, wie Datenobjekte, implementierte Services, Benutzerschnittstellen (z.B. als Maskenentwurf), Geschäftsregeln und zugrundeliegende Organisationsmodelle. Aufgrund des hierfür notwendigen formalen Charakters und der technischen Detaillierung kann es ausschließlich von IT-Spezialisten erstellt werden. Da solche in Fachbereichen nicht verfügbar sind, liegt die Verantwortung für diese Modellebene beim IT-Bereich. In vielen Unternehmen, die sich als IT-Anwender verstehen, gibt es solche Verantwortlichen nicht. Deshalb wird die Erstellung des ausführbaren Prozessmodells häufig an externe Software-Hersteller vergeben (vgl. Offshoring).

Wie Abbildung 2 zeigt, beinhalten alle drei Modellebenen relevante Aspekte wie Datenobjekte, Geschäftsregeln, Services oder Organisationsmodelle [Jab97]. Diese werden in den unterschiedlichen Ebenen (ausgehend vom Fachmodell) verfeinert. Änderungen an diesen Aspekten werden durch Fachbereiche bestätigt, und schließlich durch den IT-Bereich implementiert. Außerdem stehen die verschiedenen Objekttypen in Beziehung zueinander: So erzeugt ein Prozess verschiedene Datenobjekte, verwendet Geschäftsregeln und ruft Services auf. Da die Umstrukturierung des Kontrollflusses und einzelnen Aktivitäten im Fachmodell (vgl. Kapitel 2) die größte Herausforderung bzgl. Modelltransformationen darstellt, fokussieren wir im Folgenden darauf.

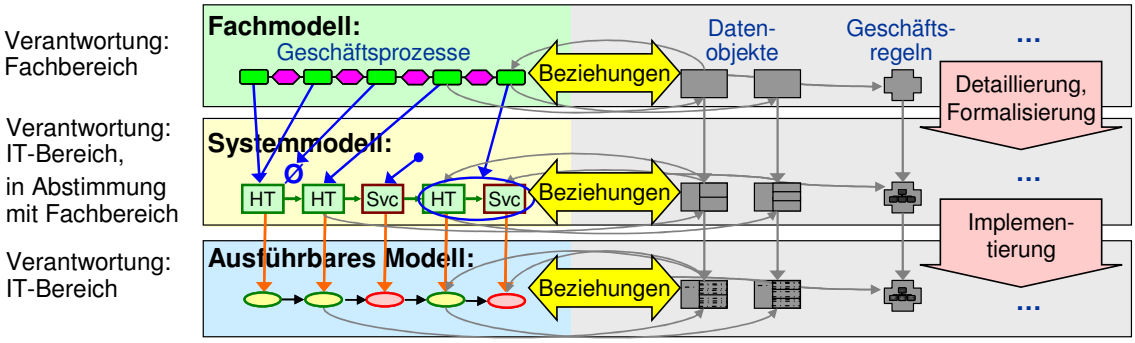


Abbildung 2: Ebenen der Modellierung von Prozessen und auch anderer Aspekte

3.2. Beziehung zwischen Fach- und Systemprozess

Eine wesentliche Anforderung ist die Nachvollziehbarkeit von Änderungen. Insbesondere ist eine explizite Bestätigung aller im Systemmodell durchgeführten Veränderungen durch den Fachbereich gefordert, um eine korrekte Abbildung fachlicher Prozesse auf Prozesse des Systemmodells (Systemprozess) sicherzustellen. Im Folgenden untersuchen wir, welche Varianten bei der Transformation eines Fachprozesses in einen Systemprozess prinzipiell möglich sind:

Variante 1 (Kopieren des Fachprozesses): Der Fachprozess wird in das Systemmodell[1] kopiert und dort angepasst (d.h. umstrukturiert). Da Systemmodelle typischerweise in einem anderen Werkzeug als Fachmodelle erstellt werden, erfolgt an dieser Stelle ein Werkzeugwechsel (z.B. von einem Geschäfts-

[1] Selbst wenn kein Systemmodell erstellt wird, sind die entsprechenden Transformationen durchzuführen. Sie werden dann bei der Erstellung des ausführbaren Modells oder einer textuell beschriebenen IT-Spezifikation durchgeführt – ebenfalls ohne jede Art einer methodisch unterstützten Nachvollziehbarkeit.

prozessmodellierungswerkzeug in ein UML-Tool). Durch diesen Werkzeugwechsel wird ein direktes Kopieren des Fachprozesses in das Systemmodell erschwert, so dass entsprechende Import-Funktionalitäten notwendig werden. Aufgrund der unterschiedlichen Metamodelle auf Fach- und Systemmodell-Ebene (z.B. eEPKs und UML Activity Diagram) sowie beschränkter Import-Funktionalität der verwendeten Werkzeuge, kommt es beim Import vielfach zu Informationsverlusten. Oftmals ist es dann sogar sinnvoller, den Fachprozess manuell zu übernehmen, d.h. den Systemprozess entsprechend dessen Metamodell und Werkzeug nach zu modellieren.

Unabhängig davon, welcher Weg zur Modellübernahme gewählt wird, existiert nach Veränderung des Fachprozesses keine Information dazu, welche Transformationen an welchen Stellen des Systemmodells durchgeführt werden müssen. Diese Variante ist gerade bei Einbeziehung künftig notwendiger Änderungen des Fachprozesses ungeeignet, da die Nachvollziehbarkeit zwischen den Modellen nicht sichergestellt wird.

Variante 2 (Verwendung von Sub-Prozessen): Eine Möglichkeit zur Verfeinerung von Prozessinformationen, die auch von heutigen Werkzeugen unterstützt wird, ist die Verwendung mittels Sub-Prozessen. Wie in Abbildung 3a dargestellt, kann dadurch z.B. eine Aktivität aufgespaltet werden. Die Beziehung zwischen fachlicher Aktivität im Fachprozess und Sub-Prozess im Systemmodell bleibt erkennbar. Dies kann im ARIS-Werkzeug z.B. dadurch realisiert werden, dass ein Sub-Prozess als sog. „Hinterlegung“ der Original-Aktivität referenziert wird. Ist zwischen den Modellebenen eine Werkzeuggrenze zu überwinden, muss analog zu Variante 1 der Fachprozess importiert werden.

Mit diesem Ansatz ist lediglich das Umbenennen (vgl. Kapitel 2, Typ 1), Aufspalten (Typ 2) und Weglassen (Typ 5) von Aktivitäten realisierbar. Ein Zusammenfassen (Typ 3) ist dagegen nicht möglich, da der Subprozess-Mechanismus nur auf Einzelaktivitäten anwendbar ist, d.h. nicht auf mehrere zu verschmelzende Aktivitäten des Fachprozesses (gemeinsam). Ebenso wenig ist ein Einfügen von Aktivitäten (Typ 4) realisierbar, da kein zu verfeinerndes Objekt im Fachprozessmodell existiert. Außerdem ist der Gesamtprozess auf Systemmodell-Ebene nicht mehr vorhanden, sondern nur mittels Fachprozess und Verfeinerungen rekonstruierbar. Dies ist nicht nur sehr unübersichtlich, sondern erschwert auch die Ableitung des ausführbaren Modells; BPEL-Generatoren etwa gehen von einem zusammenhängenden Quellprozess aus. Wegen dieser Nachteile scheidet diese Variante aus.

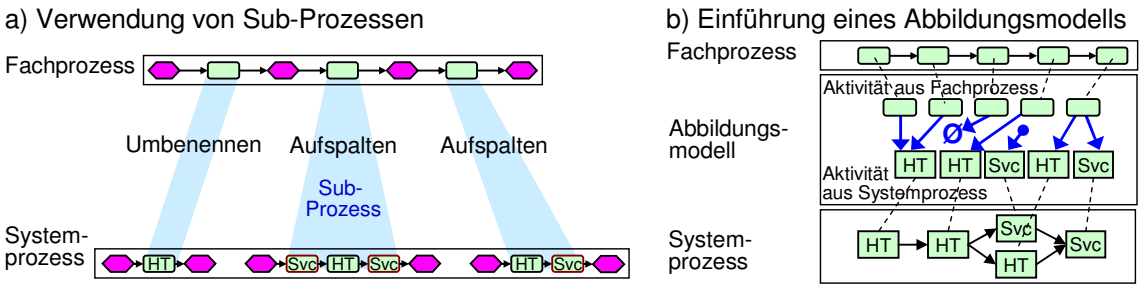


Abbildung 3: Varianten zur Verwaltung der Beziehung zwischen Fach- und Systemmodell

Variante 3 (Einführung eines Abbildungsmodells): Um die Vielfältigkeit entsprechender Transformationstypen vernünftig zu handhaben, ist eine explizite Repräsentation der Transformationsschritte zur Abbildung von Fachprozessen in Systemprozesse (und umgekehrt) nötig. Dazu führen wir im Folgenden einen neuen Modelltyp ein, dessen Instanzen wir als *Abbildungsmodell* bezeichnen. Es zeigt für alle Aktivitäten des Fachprozesses auf, wie sie in den Systemprozess überführt werden. Ebenso ist für alle Aktivitäten des Systemprozesses erkennbar, aus welchen fachlichen Aktivitäten sie entstanden sind. Wie in Abbildung 3b angedeutet, lässt sich so ein Umbenennen (Typ 1), Aufspalten (Typ 2), Zusammenfassen (Typ 3), Einfügen (Typ 4) und Weglassen (Typ 5) dokumentieren.

Zweck des Abbildungsmodells ist Beziehungen zwischen Fachmodell und Systemmodell zu dokumentieren. Deshalb beschreibt das Abbildungsmodell keine Reihenfolge zwischen Aktivitäten des Fachmodells. Muss zwischen diesen Ebenen eine Werkzeuggrenze überwunden werden, so müssen nur fachliche Aktivitäten in das (Modellierungs-) Werkzeug des Systemmodells importiert werden (und kein Kontrollfluss, vgl. Abbildung 3b). Dies ist mit einfachen Mitteln möglich, da hier keine Überführung des Fachprozesses in ein anderes Metamodell notwendig ist.

Mit Variante 3 lassen sich alle Transformationstypen dokumentieren. Der Ansatz ist noch erweiterbar,

indem zusätzliche Transformationstypen für evtl. zukünftig benötigte weitere Arten von Transformationen definiert werden. Alle durchgeführten Veränderungen sind unmittelbar erkennbar und die Beziehungen von Aktivitäten des Fach- und Systemmodells sind bidirektional nachvollziehbar. Prozessmodellierungswerkzeuge unterstützen im Regelfall eine Navigation in beide Richtungen, auch wenn die Beziehungen selbst unidirektional sind. Ein Nachteil dieser Variante ist es, dass ein zusätzliches Abbildungsmodell benötigt wird, welches manuell gepflegt werden muss. Da alle anderen Varianten gravierende Nachteile aufweisen, entscheiden wir uns für Variante 3.

3.3. Abbildung des Systemmodells auf das ausführbare Modell

Wie erwähnt, soll die Transformation des Systemprozesses in das ausführbare Modell so einfach wie möglich sein, da diese Aufgabe häufig von externen Dienstleistern, ohne Kenntnis der Fachprozesse, erledigt wird. Die Überführung der Information aus dem Systemmodell in ein ausführbares Modell findet über eine 1:1 Abbildung statt. Dabei werden alle im Systemmodell beschriebenen Sachverhalte übernommen und entsprechend der Zielplattform technisch detailliert. So ist ein Systemmodell-Prozess z.B. als UML Activity Diagram dokumentiert und soll als BPEL-Prozess implementiert werden. Dazu werden die im Systemmodell dokumentierten manuellen Aktivitäten etwa als Human Tasks [Agr07] realisiert. Außerdem müssen geeignete BPEL-Konstrukte (z.B. While, Parallel-ForEach) verwendet werden, um den im Systemmodell dokumentierten Kontrollfluss in einem BPEL-Prozess zu realisieren.

Die Nachvollziehbarkeit zwischen den Modellen ist leicht möglich, da jeder Sachverhalt aus dem Systemmodell direkt in das ausführbare Modell überführt und mit zielplattformabhängiger Information angereichert wird. Eine Zuordnung der einzelnen Objekte (z.B. Aktivitäten) aus den beiden Modellen ist mittels ihrer Namen möglich. Lediglich für den Sonderfall, dass im Systemmodell noch nicht die (endgültigen) Namenkonventionen der Zielplattform angewandt werden, ist noch eine Abbildungstabelle zu erstellen. Diese hat eine einfache Struktur, da eine Aktivität des Systemmodells stets genau einer Aktivität des ausführbaren Modells zugeordnet ist.

Da die Abbildung zwischen Systemmodell und ausführbarem Modell keine relevanten Herausforderungen bzgl. der Nachvollziehbarkeit bietet, wird dieser Aspekt im Folgenden nicht weiter betrachtet.

4. Gestaltung des Abbildungsmodells

Die bei der fachlichen Prozessmodellierung zu verwendenden Tools und Notationen (z.B. eEPK [Sch98], BPMN) werden meist bestimmt durch die Gegebenheiten im jeweiligen Fachbereich. Ebenso besteht meist kein Freiheitsgrad bei der Auswahl der Implementierungsplattform und -sprache (z.B. IBM WPS und BPEL [OAS07]). Hingegen muss das Systemmodell zwar gewisse Anforderungen wie Verständlichkeit erfüllen, der IT-Bereich hat aber prinzipiell mehrere Modellierungssprachen zur Auswahl (z.B. eEPK, BPMN), sofern nicht Unternehmensrichtlinien die Verwendung einer bestimmten Notation oder eines Modellierungswerkzeugs vorgeben. Sprache und Werkzeug haben gewisse Auswirkungen auf die Qualität des Abbildungsmodells, auf die in Abschnitt 4.3 eingegangen wird.

Ein Abbildungsmodell, wie in Abbildung 3b skizziert, stellt das Verbindungsglied zwischen Fachprozess und Systemprozess dar. Es wird von heutigen Prozessmodellierungswerkzeugen nicht direkt unterstützt. Wir zeigen im Folgenden, wie ein Abbildungsmodell prinzipiell gestaltet werden muss, damit die Beziehung zwischen Fachprozess und Systemprozess nachvollziehbar ist. Schließlich wird noch aufgezeigt, wie sich das Abbildungsmodell für das in Abbildung 1 eingeführte Beispiel mit ausgewählten Notationen realisieren lässt.

4.1. Erstellung und Speicherung

Das Abbildungsmodell wird während der Erstellung des Prozesses auf Systemmodell-Ebene (vgl. Abbildung 2) erzeugt. Da Abbildungs- und Systemmodell in der Verantwortung des IT-Bereichs liegen, sollte dasselbe Modellierungswerkzeug und möglichst dieselbe Notation verwendet werden. Allgemein betrachtet, definiert das Abbildungsmodell eine Menge von Relationen, die Aktivitäten des Fachprozesses auf Aktivitäten des Systemmodells abbilden. Jede dieser Relationen entspricht genau einem Transformationstyp (vgl. Kapitel 2). In der Modellierungsumgebung sollte eine Relation durch jeweils ein Objekt realisiert werden, dem außer dem Transformationstyp auch Attribute wie Name, Beschreibung oder der Verantwortliche aus dem Fachmodell zugeordnet sind.

Da heutige Modellierungswerkzeuge das Konzept des Abbildungsmodells nicht unterstützen, muss bei

ihrer Verwendung für diesen Zweck ein existierender Modelltyp verwendet werden (z.B. eEPK, BPMN oder UML Activity Diagram). Dieser muss Aktivitäten aus dem Fach- und Systemprozess sowie Relationen zwischen diesen darstellen können. Je nach Notation und Werkzeug kann hiervon ein Modelltyp für Abbildungsmodelle abgeleitet werden. In diesem abgeleiteten Modell können dann spezielle Knotentypen für Transformationen (Transformationsknoten) sowie Kantentypen für Transformationskanten verwendet werden. Welche Notationen in welchen Fällen geeignet sind, betrachten wir detailliert in Abschnitt 4.3. Abbildung 4 zeigt exemplarisch das zu dem in Abbildung 1 eingeführten Beispiel gehörende Abbildungsmodell in einer „neutralen" Notation. Hier werden die Transformationsknoten durch Achtecke repräsentiert, um sie sofort von Aktivitäten unterscheiden zu können; Transformationskanten werden gestrichelt dargestellt.

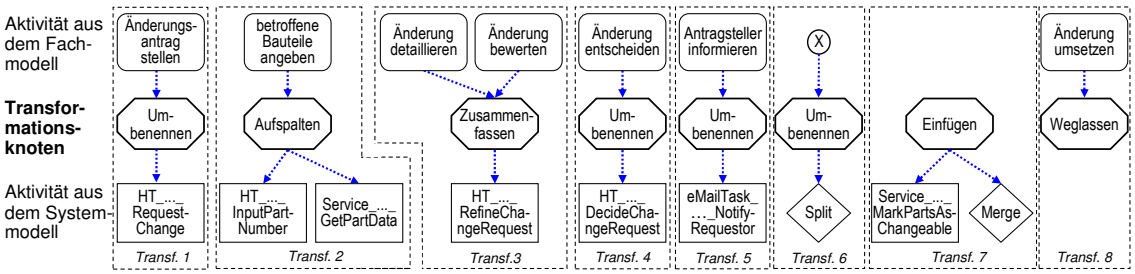


Abbildung 4: Abbildungsmodell für das in Abbildung 1 vorgestellte Beispielszenario

Damit ein Abbildungsmodell, die Beziehung zwischen den Aktivitäten des Fachmodells und des Systemmodells dokumentieren kann, müssen die betreffenden Aktivitäten referenziert werden. Dies ist für den Systemprozess einfach möglich, wenn die Referenzierung innerhalb desselben Werkzeugs erfolgt, d.h. durch dieselben Objektinstanzen verwendet werden (z.B. Ausprägungskopien in ARIS). Häufig wird das Fachmodell allerdings mit einem anderen Werkzeug erstellt. Diese Werkzeuggrenze muss bei der Referenzierung überwunden werden. Dazu gibt es prinzipiell folgende Möglichkeiten:

Ansatz 1: Die Aktivitäten des fachlichen Modells werden über eine standardisierte Schnittstelle (z.B. XML-Format) exportiert und in das Systemmodell importiert. Anschließend sind Kopien der Aktivitäten aus dem Fachmodell im Systemmodell verfügbar. Diese Kopien enthalten einen Identifier (ID) aus dem Fachprozess, so dass eine eindeutige und werkzeugübergreifende Zuordnung zu Objekten im Fachmodell möglich ist. Zusätzlich sollten auch beschreibende Daten (z.B. der Name des fachlichen Verantwortlichen) der Aktivitäten aus dem fachlichen Modell importiert werden.

Sollte zwischen den verwendeten Werkzeugen kein geeignetes Austauschformat oder keine Export- und Import-Funktionalität existieren, genügt es auch, die Aktivitäten-IDs und wegen der Übersichtlichkeit möglichst auch noch den Namen, aus dem fachlichen Modell auszulesen und entsprechende Objekte im Systemmodell anzulegen.

Ansatz 2: Eine elegantere Lösung bietet die Einbindung eines zentralen Repositories [BBP09]. Hier kann auf bilaterale Schnittstellen verzichtet werden. Dies ist nützlich, wenn in verschiedenen Projekten unterschiedliche Werkzeuge für die Erstellung von Fachmodellen und Systemmodellen verwendet werden. Die Aktivitäten des Fachprozesses werden, sobald ein gewisser Freigabestand erreicht ist, in das Repository geschrieben. Wird mit der Erstellung des Systemprozesses begonnen, werden die Aktivitäten aus dem Repository in das Systemmodell importiert. Anschließend werden die IDs aus dem Fachprozess in einem Attribut gespeichert und als Identifikationsmerkmal im Systemprozess verwendet.

Das SOA-Repository kann sogar eine noch weitergehende Rolle spielen, indem das gesamte Abbildungsmodell dort abgelegt wird (durch einen Export der Objekte und Beziehungen). Dadurch wird es langfristig archiviert, d.h. über die Laufzeit des aktuellen Umsetzungsprojektes hinaus (selbst wenn die Modellierungswerkzeuge deinstalliert werden). Durch die Werkzeug-unabhängigen Schnittstellen eines SOA-Repository ist es außerdem für einen größeren Personenkreis nutzbar als ausschließlich die Systemmodell-Ersteller und kann auch in anderen Werkzeugen wieder verwendet werden. All dies ist wichtig, wenn bei der Erstellung einer späteren Version der Prozessimplementierung andere Prozess-Modellierungswerkzeuge zum Einsatz kommen.

Das Abbildungsmodell kann manuell erstellt und gepflegt werden. Jedoch wäre eine Werkzeug-Unterstützung sinnvoll. Hierzu wird eine Werkzeugfunktionalität benötigt (vgl. Abbildung 5), die für

Aktivitäten aus dem Fachmodell (1), nach Angabe einiger Parameter (2), automatisch das entsprechende Abbildungsmodell (3a) generiert und Aktivitäten im Systemmodell (3b) bereitstellt.

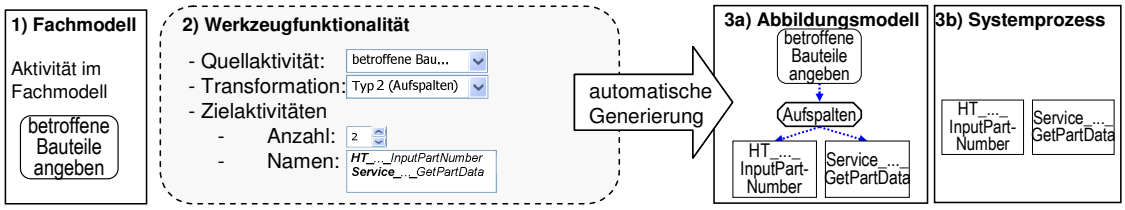


Abbildung 5 Werkzeugfunktionalität am Beispiel des Transformationstyp 2 (Aufspalten)

4.2. Verwendung des Abbildungsmodells

Wie erwähnt, ermöglicht das Abbildungsmodell, die Nachvollziehbarkeit durchgeführter Transformationen und eine schnelle Zuordnung (und damit Umsetzung) von geänderten fachlichen Anforderungen zu Aktivitäten des ausführbaren Modells. Ebenso können Veränderungen der Umgebung des ausführbaren Modells, z.B. die Abschaltung eines Services (vgl. Abbildung 1, `Service_GetPartData`), in diesem Ansatz leichter den fachlichen Aktivitäten (`betroffene Bauteile angeben`) zugeordnet werden. Dies ermöglicht eine vom Fachbereich getriebene Anpassung des Prozesses. Schließlich kann die Information des Abbildungsmodells auch zur Ausführungszeit (Run-Time) der Workflows genutzt werden, um z.B. nach einem Monitoring von Aktivitäten (Business Activity Monitoring, BAM) abzuprüfen, ob die fachlich definierten Anforderungen erfüllt werden (z.B. Bearbeitungsdauer oder Abbruchquote).

4.2.1. Sicherstellung der Konsistenz

Das Abbildungsmodell eröffnet außerdem die Möglichkeit, die Konsistenz und Aktualität zwischen den beiden Modellebenen zu erhöhen. Dadurch werden Fehler vermieden, und es lässt sich sicherstellen, dass der implementierte Prozess tatsächlich die fachlichen Anforderungen umsetzt.

Konsistenzanforderung 1: Business-IT-Alignment bedingt, dass Änderungen im Regelfall im fachlichen Modell initiiert und danach in die weiteren Modellebenen übernommen werden. Das vorgestellte Abbildungsmodell bietet eine geeignete Basis, um dem gerecht zu werden, da auf seiner Grundlage Konsistenzanalysen automatisiert durchgeführt werden können. Hierzu werden nach einer Änderung des Fachprozesses dessen Aktivitäten in das Abbildungsmodell importiert (Schritt 1 in Abbildung 6). Die Konsistenzanalyse (Schritt 2) vergleicht anschließend die im Abbildungsmodell aktualisierte Menge der fachlichen Aktivitäten mit der bereits vorhandenen. Sind bestimmte Aktivitäten nicht mehr enthalten, wurden diese aus dem Fachprozess gelöscht (2a). Diese Information wird z.B. in Form eines Berichts (Report) an Modellierer kommuniziert (3). Daraufhin passt er den Prozess des Systemmodells und das Abbildungsmodell geeignet an, indem er diese Aktivitäten entfernt.

Wird das Fachmodell um neue Aktivitäten erweitert, erkennt die Konsistenzanalyse, dass fachliche Aktivitäten im Abbildungsmodell fehlen (2b). Hierauf muss der Modellierer reagieren, indem er das Abbildungsmodell und ggf. das Systemmodell geeignet erweitert, sofern die Aktivität technisch relevant ist.

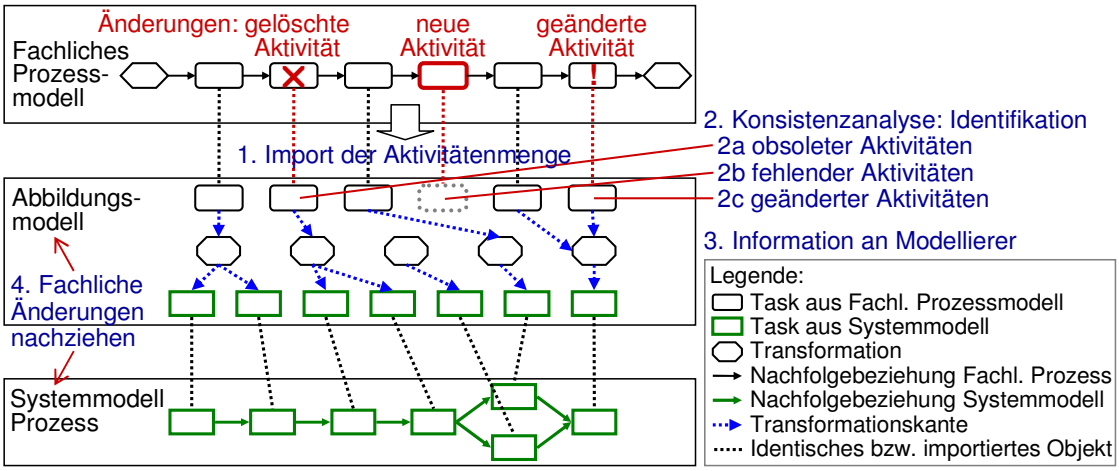


Abbildung 6: Vorgehensweise bei Konsistenzanalysen und dem Einbringen fachlicher Änderungen

Konsistenzanforderung 2: Verwendet das Werkzeug zur Fachprozessmodellierung zugreifbare Zeitstempel für Aktivitäten-Objekte (z.B. Attribut „letzte Änderung“ in ARIS), sind Veränderungen fachlicher Aktivitäten erkennbar. Die Konsistenzanalyse vergleicht nicht nur die aktualisierte mit der bereits vorhandenen Aktivitätsmenge, sondern zusätzlich die Zeitstempel einzelner Aktivitäts-Objekte. Hierzu werden diese Zeitstempel exportiert und in den entsprechenden Objekten des Abbildungsmodells gespeichert. Nach erneutem Import der Objekte ins Abbildungsmodell, sind Aktivitäten mit verändertem Zeitstempel einfach erkennbar (2c). Natürlich gibt es bei Änderungen an Einzelaktivitäten auch Fälle, die vom Modellierer des Systemprozesses ignoriert werden können, weil sie für die IT-Implementierung irrelevant sind (z.B. veränderte Kosten für eine Aktivitätenausführung, die lediglich in Verbindung mit Prozess-Simulation und -Analyse relevant sind). Eine Analyse durchzuführender Änderungen liefert somit eine Obermenge der tatsächlich notwendigen Anpassungen: allerdings ist stets sichergestellt, dass keine Veränderungen unentdeckt bleiben.

Konsistenzanforderung 3: In der Praxis sind nicht alle Änderungen fachlich getrieben. Oftmals ist es notwendig, Änderungen direkt im Systemmodell (oder sogar im ausführbaren Modell) vorzunehmen, ohne zuvor das fachliche Prozessmodell angepasst zu haben. Dies kann erforderlich werden, wenn kurzfristig auf Änderungen im laufenden Betrieb reagiert werden muss, etwa wenn ein verwendeter Service plötzlich nicht mehr zur Verfügung steht. Auch solche Änderungen können mittels Abbildungsmodell und Konsistenzanalysen zuverlässig erkannt werden. Wird eine Aktivität aus dem Prozess des Systemmodells entfernt, wird erkannt, dass sich die betroffene Aktivität noch im Abbildungsmodell befindet. Gemeinsam mit dem Fachbereich wird dann entschieden, ob nur das Abbildungsmodell angepasst werden soll (so dass die Inkonsistenz aufgelöst wird), oder ob sich die durchgeführte Änderung auch auf das fachliche Prozessmodell auswirkt.

Für neu in das Systemmodell aufgenommene Aktivitäten erkennt unser Analysealgorithmus, dass diese im Abbildungsmodell (noch) nicht vorhanden sind. Ein häufig auftretender Fall ist, dass bereits existierende Aktivitäten des Systemmodells verändert werden, etwa wenn fort an ein anderer Service bzw. eine andere Serviceversion gerufen oder eine Bearbeiterzuordnungsregel angepasst werden soll. Solche Änderungen werden ebenfalls erkannt und können geeignet umgesetzt werden. Durch Vergleich der Zeitstempel erkennt die Konsistenzanalyse auch hier wieder, dass eine Änderung erfolgt ist. Dann werden der Zeitstempel der entsprechenden Aktivität des Abbildungsmodells aktualisiert und die Veränderung im Fachprozess dokumentiert, sofern sie fachlich relevant war.

Damit Änderungen schnell und komfortabel durch Modellierer bearbeitet werden können, sollten sie aktiv über durchzuführende Anpassungen benachrichtigt werden. Hierfür bietet sich an, die Information über Änderungen direkt durch das Systemmodell zu visualisieren (Schritt 3 in Abbildung 6): So kann in das entsprechende Werkzeug eine Aufgabenliste integriert werden, welche alle noch durchzuführenden Änderungen enthält. Diese werden nach Erledigung (4) vom Modellierer entsprechend markiert. Alternativ oder zusätzlich können betroffene Aktivitäten im Systemmodell-Prozess hervorgehoben werden, solange bis der Modellierer die Behebung der Inkonsistenz bestätigt. Beide Varianten verhindern, dass durchzuführende Änderungen einfach vergessen werden. Für die Realisierung der Varianten ist Vorraussetzung, dass das Modellierungswerkzeug eine (erweiterbare) Funktionalität für Aufgabenlisten bzw. Aktivitäten-

markierungen anbietet.

4.2.2. Potentielle Einsatzszenarien und Erweiterungen für das Abbildungsmodell

Eine weitergehende Nutzung des Abbildungsmodells wird möglich, wenn sein Informationsgehalt erweitert wird.

Erweiterung 1: Der Kontrollfluss (Sequenzen, Schleifen, etc.) zwischen Aktivitäten des Fachprozesses könnte in das Abbildungsmodell aufgenommen werden, um dadurch nach einem Re-Import Veränderungen automatisch erkennen zu können (z.B. vertauschte Reihenfolge von Aktivitäten). Wie erwähnt, ist aber eine Überführung von Prozessmodellen in eine andere Modellierungssprache schwierig. Entsprechende Veränderungen können einfacher durch einen Vergleich der beiden Modellversionen des Fachprozesses erkannt werden (was von üblichen Modellierungswerkzeugen direkt unterstützt wird). Deshalb wird auf eine Verwaltung des Kontrollflusses im Abbildungsmodell verzichtet und somit der resultierende Aufwand vermieden.

Erweiterung 2: Analog könnte auch im Abbildungsmodell vorhandene Information über den Kontrollfluss des Systemprozesses genutzt werden: Ist z.B. bekannt, in welcher Reihenfolge die Aktivitäten des Systemmodells ausgeführt werden sollen, in die eine fachliche Aktivität bspw. aufgespalten wird, so kann automatisch ein Teil des im Systemmodell abgebildeten Prozesses generiert werden. Zusammen mit dem Kontrollfluss des veränderten Fachprozesses kann so theoretisch der gesamte Prozess auf Systemmodell-Ebene neu generiert werden. Die bisherige Version des Systemmodells wird hierbei nicht verwendet. Allerdings ist eine automatische Generierung des gesamten Systemmodell-Prozesses kaum realisierbar und praktikabel, da es wegen der vagen und informalen Beschreibung der Fachprozesse extrem schwierig ist, die resultierenden (komplexen) Transformationsregeln formal zu spezifizieren. So können auch Fälle auftreten, in denen im Fachprozess sequentiell modellierte Aktivitäten in Systemprozess parallel angeordnet sein sollen (z.B. um die Ausführungszeit zu reduzieren), oder aus einem Aufspalten einer fachlichen Aktivität resultierende Aktivitäten im Systemmodell nicht unmittelbar aufeinander folgen, was im Abbildungsmodell nicht durch ein Kontrollfluss-Fragment beschrieben werden kann.

Deshalb verfolgen wir einen grundsätzlich anderen Weg als in Erweiterung 1 und 2 beschrieben: Die letzte existierende (und aufwendig erstellte) Version des Systemprozesses wird weiter verwendet (d.h. nicht verworfen), und die erforderlichen Änderungen aus dem Fachprozess werden dort nachvollzogen. Diese Vorgehensweise führt unserer Ansicht bei fachlichen Prozessmodellen nicht nur zu einer besseren Qualität des (nun manuell erstellten) Systemmodells, sondern auch zu einem geringerem Pflegeaufwand der Modelle: Das Abbildungsmodell muss dann nur die Beziehungen zwischen Aktivitäten beider Modelle widerspiegeln und es müssen weder komplexe Ablauffragmente gepflegt noch Regeln für deren Anwendung definiert werden.

4.3. Notationen und Werkzeuge zur Erstellung des System- und Abbildungsmodells

Wie erwähnt, müssen zur Erstellung des Fachprozesses eine im Fachbereich gut verständliche und etablierte Notation und ein entsprechendes Werkzeug verwendet werden, wie z.B. eEPK (z.B. IDS-Scheer ARIS) oder BPMN (z.B. IBM WebSphere Business Modeler (WBM) [IBM09] oder ebenfalls ARIS). Bei einer Geschäftsprozess-Modellierung mit anderen Werkzeugen wie z.B. Adonis [BOC08] oder Bonapart [BTC09] ergeben sich ähnliche Auswirkungen für das Abbildungsmodell, da die Notationen dieser Werkzeuge auf denselben Benutzerkreis ausgerichtet sind und deshalb ähnliche Eigenschaften haben. Deshalb wird in diesem Beitrag auf eine Einzelbetrachtung für weitere Werkzeuge verzichtet.

Das ausführbare Modell wird z.B. mittels BPEL realisiert. Da es viele Hersteller von BPEL-Engines (mit jeweils ihrem eigenen BPEL-Derivat) gibt, werden wir uns auf die exemplarische Betrachtung für nur eines dieser Produkte beschränken, nämlich den IBM WebSphere Process Server (WPS) [IBM09].

4.3.1. Anforderungen an das Systemmodell

Ziel dieses Abschnitts ist es, die verschiedenen für das Systemmodell in Frage kommenden Notationen zu vergleichen und zu bewerten. Sofern relevant, werden auch produktspezifische Erweiterungen konkreter Notationen betrachtet. Um eine Bewertung vornehmen zu können, werden zuerst die Anforderungen an das Systemmodel (d.h. Vergleichskriterien) vorgestellt.

Kriterium 1 (Formalitätsgrad): Der Prozess des Systemmodells muss ausreichend formal beschrieben sein, so dass er in einem Pflichtenheft für eine IT-Umsetzung verwendet werden kann. Hierzu muss die Bedeutung aller modellierbaren Objekte im Systemmodell eindeutig definiert sein. Dies fordert, dass

gewisse Kontrollflussstrukturen verboten werden, da z.B. bei einem Rücksprung aus einem einzelnen Zweig einer Parallelität heraus unklar ist, was dies für die anderen Zweige bedeuten soll (Abbruch, Beendigung, mit oder ohne späterer erneuten Ausführung dieser Aktivitäten).

Kriterium 2 (Vollständigkeit): Das Systemmodell muss vollständig sein in dem Sinne, dass alle typischerweise benötigten Kontrollfluss-Konstrukte in dem Metamodell enthalten sind. So wird außer AND-/OR-/XOR-Aufspaltungen und -Zusammenführungen z.B. auch ein Konstrukt für eine zur Modellierungszeit unbekannte Anzahl paralleler Zweige benötigt (BPEL Parallel-ForEach [OAS07], Multiple Instances with a priori Run-Time Knowledge [AHK03]).

Kriterium 3 (Workflow-Aspekte): Das Systemmodell muss alle relevante Aspekte [Jab07] umfassen können und nicht auf den Kontrollfluss beschränkt sein. So müssen auch Geschäftsobjekte (Datentypen), Organisationseinheiten und der Beziehungen (Rollen, Abteilungen, Kompetenzen), Geschäftsregeln, Services, etc. beschreibbar sein. Diese Objekttypen müssen dann bei der Definition des Prozessgraphen verwendet (d.h. referenziert) werden können.

Kriterium 4 (Verständlichkeit): Da der Prozess auf Ebene des Systemmodells mit dem Fachbereich abgestimmt werden muss, ist es wichtig, dass die Notation für diesen Anwenderkreis gut verständlich ist. Eine sehr technische Darstellung ist ungeeignet.

Kriterium 5 (Abbildungsmodell-Notation): Um zu vermeiden, das die Ersteller des Systemmodells mehrere Notationen beherrschen müssen, sollte die Prozessbeschreibung und das Abbildungsmodell in derselben Notation realisierbar sein. Es ist aber auch akzeptabel, dass eine verwandte Notation verwendet wird, z.B. verschiedene UML-Diagrammtypen.

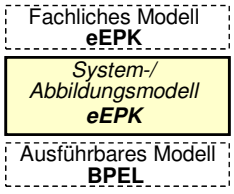
Kriterium 6 (Fachmodell-Transformation): Der Aufwand für die Transformation des existierenden Fachprozesses in das Systemmodell soll möglichst gering sein. Dies ist insbesondere dann gegeben, wenn die Prozess-Metamodelle für diese beiden Ebenen identisch, ähnlich oder zumindest „kompatibel“ sind. Dann kann der fachliche Prozess übernommen und angepasst werden, anstatt dass der Prozess auf Systemmodell-Ebene vollständig neu erstellt werden muss.

Kriterium 7 (Generierung des ausführbaren Modells): Die Art der Prozessdokumentation im Systemmodell soll für eine direkte 1:1 Ableitung des ausführbaren Prozesses etwa als BPEL oder BPMN geeignet sein. Diese Modelltransformation soll mit nur geringem intellektuellen Aufwand manuell erfolgen können oder aber automatisch durch entsprechende Generatoren (vgl. Abschnitt 6).

Anhand der Kriterien 1 bis 4 wird nachfolgend verglichen, wie gut sich der Geschäftsprozess auf Ebene des Systemmodells mit der entsprechenden Notation dokumentieren lässt. Die Kriterien 5 bis 7 beschreiben, wie gut der Übergang zwischen unterschiedlichen Modellebenen möglich ist. Bei dem Kriterium 5 (geeigneten Notation zur Realisierung des Abbildungsmodells) muss aber klar sein, dass die Verwendung (d.h. Zweckentfremdung) einer heute von Modellierungswerkzeugen unterstützten Notation nur eine Übergangslösung darstellen kann, die solange verwendet wird, bis die Werkzeuge einen speziellen Modelltyp für Abbildungsmodelle bereitstellen. Bei gewissen Werkzeugen kann ein solcher Modelltyp auch durch den Anwender erstellt werden. In UML-Werkzeugen [MID08] kann dies durch die Definition eines entsprechenden Stereotypen erfolgen, dessen Basismodell wiederum das hier diskutierte UML Activity Diagram sein wird. Adonis [BOC08] ermöglicht durch die Definition eigener Metamodelle eine beliebige Gestaltung des Modelltyps für Abbildungsmodelle wofür aber ein Customizing notwendig ist. Bei anderen Werkzeugen wie IDS-Scheer ARIS oder IBM WBM können die existierenden Modelltypen in nur eingeschränktem Umfang angepasst werden und keine neuen Modelltypen definiert werden.

Im Folgenden werden potentiell für das Systemmodell geeignete Notationen aus der Praxis (eEPK, UML Activity Diagram (UAD) und BPMN) anhand dieser Kriterien verglichen. Hierbei werden teilweise zusätzliche Varianten betrachtet, die sich durch Verwendung unterschiedlicher Notationen und Werkzeuge in den anderen beiden Modellebenen (Fachmodell und ausführbares Modell) ergeben.

4.3.2. Systemmodell als eEPK



Dieser Ansatz ist insbesondere dann interessant, wenn der fachliche Geschäftsprozess als eEPK vorliegt. Im Systemmodell wird er dann ebenfalls als eEPK modelliert. Auch für die Realisierung des Abbildungsmodells kann eine eEPK genutzt werden: Hierzu wird vom Objekttyp Funktion (oder Ereignis) ein Subtyp „Transformation“ abgeleitet, der Attribute wie Transformationstyp, Beschreibung, etc. enthält. Diesem Subtyp wird ein spezielles Symbol (z.B. Achteck) zugeordnet, um ein gute Unterscheidbarkeit von regulären Funktionen (Aktivitäten aus dem Fachmodell) zu erreichen. Diese Aktivitäten werden mit dem Transformationsknoten mittels Kanten

verbunden, die eigentlich Nachfolgebeziehungen beschreiben. Auch diese erhalten einen speziellen Subtyp, der sie als Transformationskante kennzeichnet und ggf. eine spezielle Darstellung.

Der Ansatz hat zwei entscheidende Schwachpunkte:

i) Eine Prozessbeschreibung als eEPK ist nicht ausreichend formal (Kriterium 1). So ist bei eEPKs die Semantik von Split- und Join-Knoten sowie von Rücksprüngen nicht in allen Fällen eindeutig, da z.B. die Modellierung des bereits erwähnten Sprungs aus einer Parallelität heraus möglich ist. Deshalb muss die Modellierungsmächtigkeit zusätzlich mittels Konventionen eingeschränkt werden, was dann aber nicht mehr unmittelbar durch Werkzeuge unterstützt wird, sondern allenfalls mittels Berichten (vgl. Abschnitt 4.2) analysiert werden kann.

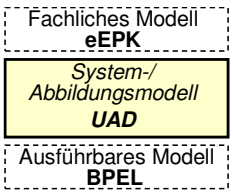
ii) Zudem werden nicht alle erforderlichen Kontrollfluss-Konstrukte (2) angeboten. So gibt es keine Multi-Instanz-Aktivität, so dass man sich hier ebenfalls mit einer selbst eingeführten Konvention als Workaround behelfen muss (z.B. die Kennzeichnung eines AND-Split mit einem bestimmten Kommentar). Dies ist aber nicht nur sehr unsauber, sondern solche Informationen sind dann auch von existierenden BPEL-Generatoren nicht nutzbar (7).

In eEPKs selbst lassen sich diverse Workflow-Aspekte referenzieren (3), die Definition der verwendeten Objekte erfolgt in anderen Diagrammtypen. Insbesondere ARIS bietet hierfür eine Vielzahl von Diagramm- und Objekttypen an. Schwierig ist allerdings die Definition der potentiellen Bearbeiter einer Aktivität, da boolesche Verknüpfungen zwischen Objekten (z.B. Rolle und Abteilung) ebenso wenig möglich sind, wie die Referenzierung von Vorgängeraktivitäten, z.B. zwecks Ausschluss des Bearbeiter einer bestimmten Aktivität (4-Augen-Prinzip). Da dies aber bei allen Metamodellen schwer möglich ist, sollte es nicht zur Abwertung von eEPKs führen.

Da im Systemmodell dieselbe Notation verwendet wird, wie für den Fachprozess, ist sowohl die Verständlichkeit für Fachanwender (4) wie auch die entsprechende Transformationsfähigkeit (6) besonders gut. Wie erwähnt, ist eine Realisierung des Abbildungsmodells als eEPK (5) möglich. Zudem existieren (kommerzielle) BPEL-Generatoren (7) die Systemprozesse in ausführbare Prozesse konvertieren, wobei deren Leistungsfähigkeit prinzipiell durch die mangelnde Formalisierung von eEPKs und die erforderliche Einführung zusätzlicher Konventionen beschränkt ist.

ARIS stellt eine Plattform zur Modellierung von eEPKs und der anderen benötigten Diagrammtypen dar. Auch ein BPEL-Export wird unterstützt, der besonders gut funktioniert, wenn SAP die Zielplattform ist. In diesem Sonderfall ist dieser Ansatz erwägenswert, wohingegen er ansonsten aufgrund der Schwachpunkte (Kriterien 1, 2 und 7) zu verwerfen ist. Dazu kommt, dass weder eEPKs noch ARIS in IT-nahen Bereichen stark verbreitet ist, so dass durch deren Verwendung evtl. noch Schulungs- und Lizenzkosten entstehen.

4.3.3. Systemmodell als UML Activity Diagram



Ein Fachprozess wird in den seltensten Fällen als UML Activity Diagram (UAD) vorliegen, da UML-Werkzeuge wie MID Innovator oder IBM Rational Software Architekt eher als Case-Tools positioniert sind. Dadurch werden UAD primär von IT-Implementierern verwendet und nicht von Fachanwendern. Wir gehen deshalb von einem Fachprozess z.B. in Form einer eEPK aus. Im Systemmodell lässt sich der Prozess als UAD modellieren, da IT-Bereiche mit CASE- und UML-Werkzeugen wie IBM Rational Software Architect [Mit05] oder MID Innovator [MID08] vertraut sind. Das Abbildungsmodell ist ebenfalls als UAD realisierbar, da es möglich ist, einen eigenen Stereotyp als Transformationsknoten zu definieren und ihn mit den Aktivitäten mittels gerichteter Kanten zu verbinden (ähnlich wie bei eEPKs).

Die Konstrukte eines UAD sind ausreichend formal definiert (1), d.h. es existiert eine definierte Menge an Konstrukten für relevante Sachverhalte (etwa Verzweigungs-, Parallelisierungs-, Synchronisations- sowie Schleifenknoten). Wobei ebenso wie bei eEPKs Einschränkung bzgl. der Prozessstruktur durchgeführt werden müssen, um stets eine definierte Ausführungssemantik zu erhalten. Allerdings stehen für ein UAD alle benötigten Konstrukte (2) zur Verfügung. Durch Verwendung von weiteren UML-Diagrammtypen lassen sich auch alle anderen Workflow-Aspekte (3) modellieren. So ist z.B. eine Datenmodellierung mittels UML-Klassendiagrammen sogar eine besonders geeignete und verbreitete Methode. Für die Festlegung von Bearbeitern einer Aktivität existieren jedoch auch bei UML die üblichen Einschränkungen. Die Verständlichkeit für Fachbereich (4) ist nicht besonders gut, aber mit einer begleitenden Erklärung durch den IT-Bereich für eine Abstimmung ausreichend. Problematisch ist hierbei die Vielzahl der dargestellten Symbole (z.B. Pins einer Activity) und die eingeschränkte Symbolik und Sym-

bolvielfalt. So können in einem UAD z.B. für unterschiedliche Aktivitätentypen nicht einfach unterschiedliche Symbole verwendet werden, um eine gute optische Unterscheidbarkeit zu erreichen.

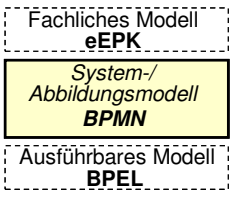
Das Abbildungsmodell kann ebenfalls als UAD (5) realisiert werden (UML würde sogar noch die Verwendung anderer Diagrammtypen ermöglichen). Auch die Transformation des Fachprozesses in ein UAD ist unproblematisch (6), da das Zielmodell ähnlich viele Freiheitsgrade bietet wie eine eEPK. Sollte der Fachprozess in einem Ausnahmefall sogar bereits als UAD vorliegen, so ist seine Übernahme natürlich noch einfacher möglich. Kritischer sind hingegen die Möglichkeiten zur Ableitung ausführbarer Prozesse (7) – BPEL und BPMN sind nicht Teil der UML-Familie. Prozessmodell-Generatoren können eine UAD nur dann transformieren, wenn sie eine eindeutige Ausführungssemantik besitzt. Um eine solche zu erreichen, müssen Modellierungskonventionen eingehalten werden, die vom UML-Werkzeug nicht sichergestellt werden. Es ist aber zu beachten, dass die Situation ungleich günstiger ist als bei eEPKs, da ein UAD über formal definierte Konstrukte für alle relevanten Sachverhalte verfügt (wie bereits erwähnt z.B. für Multi-Instanz-Aktivitäten).

4.3.4. Systemmodell in BPMN

Prozesse auf Ebene des Systemmodells können ebenso mittels BPMN realisiert werden wie das Abbildungsmodell, wobei dasselbe Vorgehen wie bei den bisher betrachteten Notationen verwendet wird. Die Besonderheit an BPMN ist, dass die Notation (zumindest ab der Version 2.0 [OMG09]) auch für die anderen Modellebenen eingesetzt werden kann, da bspw. eine formale Ausführungssemantik im Standard definiert ist. Deshalb werden im Folgenden zusätzliche Varianten betrachtet, bei denen BPMN nicht ausschließlich auf Ebene des Systemmodells verwendet wird.

Mit Gültigkeit für alle Varianten lässt sich feststellen, dass BPMN ausreichend formal ist (1), um einen Prozess des Systemmodells zu beschreiben. Dies gilt insbesondere ab BPMN 2.0, da dann sogar eine formale Ausführungssemantik definiert ist. Auch ist bereits BPMN 1.1 [OMG09] derart mächtig (2), dass sich viele relevante Abläufe darstellen lassen. So ist z.B. für die bereits erwähnte Multi-Instanz-Aktivität der Objekttyp „Multiple Instance Task“ verfügbar. Den Prozessablauf umgebende Aspekte (3) wie die Datenmodellierung im Systemmodell, lassen sich nicht unmittelbar in BPMN realisieren. Hierzu muss auf andere Sprachen wie UML zurückgegriffen werden. Somit ist ein Modellierungswerkzeug erforderlich, das beide Sprachen unterstützt (z.B. ARIS). Die Verständlichkeit von BPMN für den Fachbereich (4) ist gut, wie erwähnt wird BPMN teilweise sogar für die fachliche Modellierung verwendet. Ein Nachteil von BPMN 1.1, nämlich dass alle Rollen- und System-Zuordnung stets per Swimlane erfolgen müssen, entfällt mit der Version 2.0. Diese Einschränkung hat insbesondere bei mehrdimensionalen Zuordnungen zu unübersichtlichen Diagrammen geführt. Wie erwähnt ist auch das Abbildungsmodell in BPMN mittels Prozessfragmenten realisierbar, wobei jedes Fragment aus Tasks und Sequenzkanten besteht. Um die Tasks des Fachmodells optisch von den Transformationsknoten und den Tasks des Systemmodells abzugrenzen, können BPMN-Lanes verwendet werden (vgl. Abbildung 8).

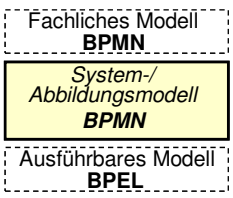
Standardfall: Fachprozess als eEPK und ausführbares Modell als BPEL



Der Aufwand für die Transformation vom Fach- zum Systemmodell (6) ist akzeptabel, aber der Vorgang selbst kaum automatisierbar: Damit eine formale Ausführungssemantik erreicht wird, wurde BPMN recht restriktiv gestaltet. Deshalb muss aus die eEPK zuerst eine BPMN-kompatible („wohl-definierte“) Struktur umgeformt werden, was intellektuellen Aufwand verursacht und somit nicht automatisierbar ist.

Die Ableitung eines BPEL-Prozesses (7) ist wegen dem formalen Charakter von BPMN und auch der nahen Verwandtschaft der beiden Sprachen (verglichen mit eEPKs und UADs) leicht möglich. So befindet sich im BPMN-Standard ein vordefiniertes Mapping auf BPEL-Konstrukte. Auch wirken Hersteller von BPEL-Engines bei der Standardisierung von BPMN 2.0 mit. Die Mächtigkeit von BPMN erlaubt sogar eine Transformation von Prozessen in Hersteller-spezifisch erweiterte BPEL-Derivate.

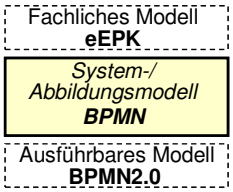
Variante 1: Fachprozess in BPMN



Wird bereits für den Fachprozess BPMN verwendet, so ist die Verständlichkeit des Systemmodells für den Fachbereich (4) besonders gut, da die BPMN-Notation bekannt ist. Der Aufwand für die Transformation des Fachprozesses in das Systemmodell (6) reduziert sich erheblich, da der Prozess direkt (ohne Konvertierung des Diagramtyps) übernommen werden kann. Dies kann etwa durch ein Kopieren des Fachprozesses geschehen, so dass eine zweite Prozessversion entsteht, die

dann mittels der erwähnten Transformationen geeignet angepasst werden kann.

Variante 2: Ausführbares Modell in BPMN

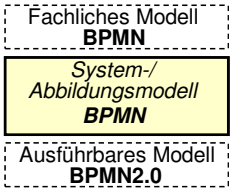


Wird für Prozesse des Systemmodells und des ausführbaren Modells BPMN verwendet, so verbessert sich die Generierbarkeit von Workflows (7) sogar noch gegenüber dem Standardfall, da Modelltransformationen völlig entfallen. Es ist nur noch eine weitere Detaillierung technischer Attribute erforderlich.

Dies ist ein durchaus relevantes Szenario, das zukünftig weiter an Bedeutung gewinnen wird. So verwendet die Modellierungsumgebung (WebSphere Integration Developer 6.2) des IBM WPS 6.2 eine an BPMN angelehnte Notation und entsprechende Konstrukte. Beispielsweise wird ein sog. Generalized Flow angeboten, innerhalb dessen BPMN-Gateways mit Split- und Join-Logik verwendet werden können. Dadurch ist beim WPS 6.2 bereits heute die Verwendung von weitaus mehr Konstrukten als in BPEL möglich.

Wird der WPS als Ausführungsumgebung gewählt, so bietet sich eine Erstellung des Systemmodells mittels BPMN sehr stark an. Hierfür muss nicht die Workflow-Implementierungsumgebung des WPS verwendet werden. Stattdessen bieten sich stärker auf fachliche Benutzer ausgerichtete BPMN-Werkzeuge an (z.B. ARIS). Auch IBM bietet mit dem WebSphere Business Modeler (WBM) [IBM08] ein entsprechendes Werkzeug, das zur Erstellung der Prozesse des Systemmodells und des Abbildungsmodells geeignet ist. Außerdem ermöglicht es einen direkten Modell-Export für den WebSphere Integration Developer und realisiert dieselben Hersteller-spezifischen Erweiterungen wie z.B. Human-Tasks und Staff-Verbs zur Festlegung der Bearbeitermenge. Außerdem wird Funktionalität zum Modellvergleich bereitgestellt, so dass im Systemmodell durchgeführte Änderungen leicht erkannt und dadurch im ausführbaren Modell zuverlässig nachgezogen werden können.

Variante 3: Alle 3 Modellebenen in BPMN



Dieses Szenario ist prinzipiell ideal. Wenn man allerdings die Modellierungswerkzeuge betrachtet, so stellt man fest, dass es heute am Markt keine Umgebung gibt, die für alle Ebenen geeignet ist: Jedes einzelne Werkzeug ist entweder zu technisch für die Fachanwender oder es ermöglicht keine ausreichend detaillierte Spezifikation von ausführbaren Prozessen. Beispielsweise bietet der IBM WBM die Möglichkeit zur Modellierung technischer Details. Das geht so weit, dass er sogar ein direktes Deployment auf die Prozess-Engine WPS ermöglicht. Damit eignet sich dieses Werkzeug gut für die Ebenen Systemmodell und ausführbares Modell. Allerdings sind die Modellierungsmöglichkeiten des WBM zu eingeschränkt, um ihn für die Erstellung eines fachlichen Modells verwenden zu können. Ein unstrukturiertes und vages Modellieren in einer frühen Analysephase ist schwer möglich und die Bedienung und Darstellungsform ist für Fachbereiche zu kompliziert, als dass sie selbst damit arbeiten könnten.

Andererseits ermöglicht ARIS die fachliche Modellierung in BPMN. Das Werkzeug kann für die Erstellung des Systemmodells verwendet werden, wenngleich es in IT-Bereichen eher weniger verbreitet ist. Die Definition eines direkt ausführbaren Prozessmodells ist hingegen ausgeschlossen, da weder die notwendige technische Tiefe erreicht wird, noch entsprechende Besonderheiten von Prozess-Engines berücksichtigt sind. ARIS verfügt deshalb folgerichtig über keine Funktion zum Deployment von Prozessen.

Es bleibt also festzuhalten, dass zwar BPMN als einheitliche Notation für alle Modellebenen verwendet werden kann, aber dennoch Werkzeugwechsel mit entsprechendem Export und Import der Prozesse stattfinden müssen. Dies ist aber recht unkritisch (verglichen mit dem Wechsel des Prozess-Metamodells), insbesondere wenn man bedenkt, dass ausführbare Prozesse meist von externen Dienstleistern implementiert werden, die häufig ohnehin ihre eigenen Werkzeug-Landschaften verwenden.

4.3.5. Fazit

Die Diskussion der Notationen hat ergeben, dass BPMN generell am besten zur Realisierung des Systemmodells geeignet ist. Dies gilt insbesondere dann, wenn BPMN auch für den Fachprozess und das ausführbare Prozessmodell verwendet wird.

Abbildung 7 zeigt das Ergebnis des Notationsvergleichs und betont, dass im Allgemeinen BPMN am besten zur Realisierung des Systemmodells geeignet ist. Deshalb wird in Kapitel 5 BPMN als Notation für eine exemplarische Umsetzung des Beispielszenarios (vgl. Kapitel 1) ausgewählt.

Systemmodell-Notation:	eEPK	UML 2.0 Activity Diagram	BPMN	Var. 1: BPMN und Fachprozess in BPMN	Var. 2: BPMN und ausführbares Modell in BPMN	Var. 3: BPMN auf allen 3 Modellebenen
1. Formalitäts-grad	– – Konventionen für Einschränkung Split/Join notwendig	– Semantik ist definiert, aber auch Konventionen notwendig	++ BPMN 2.0 sogar mit formaler Ausführungs-semantik			
2. Vollständig-keit	– – z.B. keine Multi-Instanz-Aktivitäten modellierbar	+ alle benötigten Konstrukte verfügbar	+ alle benötigten Konstrukte verfügbar			
3. Workflow-Aspekte	+ per Hinterlegung: viele verfügbaren Diagrammtypen (ARIS-spezifisch)	+ UML umfasst viele Aspekte, ggf. zusätzliche Stereotypen def.	– beschreibt nur Prozess, Kombina-tion mit UML (auch OMG) notwendig			
4. Verständlich-keit	++ Notation wie Fachprozess	o durch UML eingeschränkte Symbolik	+ BPMN teilweise auch für GPM verwendet	++ weiter verbessert, da selbe Notation		++ weiter verbessert, da selbe Notation
5. Abbildungs-modell-Notation	++ beides als eEPK	++ auch als UML Activity Diagram	++ beides in BPMN			
6. Fachmodell-Trans-formation	++ selbe Notation für beide Modell-ebenen	+ eEPK und UML Activity Diagram sind eher unformal	o aufwendiger wg. Formalitätsgrad von BPMN	++ selbe Notation für beide Modell-ebenen		++ selbe Notation für beide Modell-ebenen
7. Generierung des ausführbaren Modells	– große Unter-schiede bzgl. Struktur und Grad an Formalität	– große Unter-schiede bzgl. Struktur und Grad an Formalität	++ Mapping ist im Standard definiert		++ Modelltrans-formation entfällt (nur Ergänzung techn. Attribute)	++ Modelltrans-formation entfällt (nur Ergänzung techn. Attribute

Abbildung 7 Vergleich unterschiedlicher Notationen für das Systemmodell

5. Prototypische Umsetzung des Abbildungsmodells

Im Folgenden wird anhand des in Kapitel 1 eingeführten Beispiel demonstriert, wie ein Abbildungsmodell mit der Business Process Modeling Notation (kurz: BPMN) realisiert werden kann. Außerdem wird auf identifizierte Schwierigkeiten bei der Modellierung eingegangen, um einen Eindruck zu vermitteln, inwieweit das Konzept des Abbildungsmodells mit heutigen Prozessmodellierungs- und Prozessausführungswerkzeugen umgesetzt werden kann. Hierfür wird zuerst das Werkzeug IBM WebSphere Business Modeler 6.2 verwendet. Es bietet im Vergleich zu eher fachlichen Modellierungswerkzeugen eine höhere Durchgängigkeit zum ausführbaren Modell, erfordert deswegen aber auch die Einhaltung diverser Vorgaben bei der Modellierung. Zum Vergleich wird anschließend eine Umsetzung in ARIS vorgestellt, wobei ebenfalls BPMN als Notation für das Abbildungsmodell verwendet wird.

5.1. IBM WebSphere Business Modeler (WBM)

Wie Abbildung 8 zeigt, können mit WBM Abbildungsmodelle erstellt werden[2]. So separiert die BPMN Swimlane-Darstellung die Objekte (Aktivitäten) aus dem Fach- und Systemprozess sowie die dazwischen liegenden Transformationsknoten. Eine entsprechende Kategorisierung der Objekttypen erlaubt zudem eine farbliche Markierung. Außerdem können die Namen der Transformationsknoten so gewählt werden, dass der Transformationstyp leicht erkennbar ist. Die Eindeutigkeit der Knoten, wird durch deren Erweiterung um eine fortlaufende Nummer erreicht. Es können hier auch Namen gewählt werden, die den Transformationsgrund näher beschreiben (z.B. „Einfügen: Service für Zustandsänderung im PDM“).

[2] Die zugehörigen Geschäftsprozess-Diagramme des fachlichen Modells und des Systemmodells sind im Anhang dargestellt.

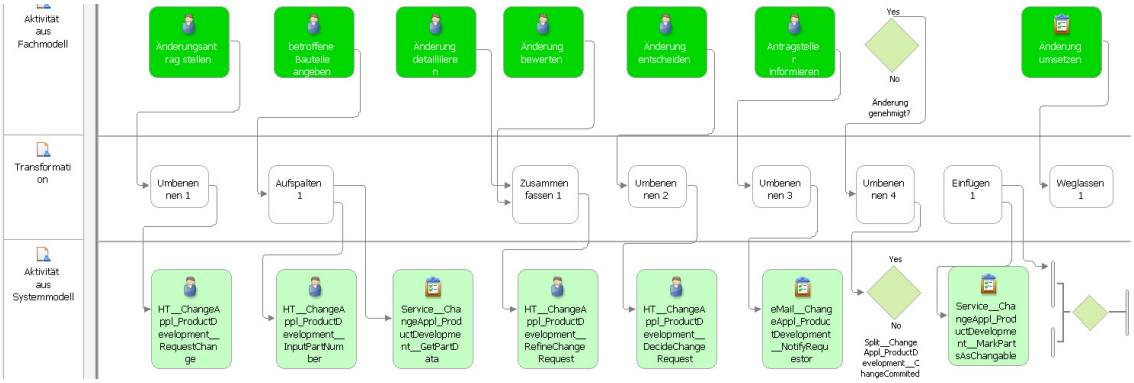


Abbildung 8 Abbildungsmodell in BPMN erstellt mit IBM WebSphere Business Modeler

Die Erstellung des Abbildungsmodells wird durch das Werkzeug WBM teilweise etwas erschwert, weil benötigte Funktionalität fehlt. So gibt es z.B. keine komfortable Möglichkeit im Fachprozess bereits existierende Aktivitäten ins Abbildungsmodell zu kopieren und dieselbe Objektinstanz zu referenzieren (vgl. ARIS Ausprägungskopien). Sollen keine unabhängigen Kopien erzeugt werden (z.B. damit spätere Namensänderungen in beiden Objektinstanzen durchgeführt werden), muss das Objekt mittels des Navigationsbaums erzeugt werden, bevor es in das erste Prozessdiagramm eingefügt wird (was umständlich ist und leicht vergessen wird).

Weitere Schwierigkeiten entstehen durch die eher technische Ausrichtung des WBM: Da die Prozessmodelle eine Ausführungssemantik haben, sind für Aktivitäten die Ein- und Ausgabedaten (sogenannte Ports) ebenso zu definieren, wie die entlang der Kanten fließenden Daten. Somit sind auch die Transformationskanten des Abbildungsmodells mit einem (eigentlich nicht existierenden) Datenfluss belegt. Die zugehörigen Datenobjekte können beliebig gewählt werden. Der WBM ermöglicht es diese ebenso auszublenden, wie die Ports der Aktivitäten. Deren Darstellung wäre insbesondere bei Aktivitäten mit mehreren Ports irritierend, da im Abbildungsmodell stets nur einer von diesen verwendet wird. Wird für das Abbildungsmodell der Modeling Mode „Basic“ verwendet, zeigt der WBM keine Fehlermeldungen und Warnungen z.B. für nicht verbundene Task-Ausgabedaten oder unbefüllte Attribute von Transformationsknoten an. Man sollte sich aber bewusst sein, dass diese Fehler existieren, auch wenn sie nicht relevant sind, weil kein Deployment des Abbildungsmodells auf eine Ausführungsumgebung vorgesehen ist.

5.2. ARIS

Wie Abbildung 9 zeigt, kann mit dem ARIS Business Architect 7.1 ein graphisch ansprechendes Abbildungsmodell als BPMN-Diagramm realisiert werden. Das Abbildungsmodell enthält fachliche Aktivitäten einer entsprechenden erweiterten ereignisgesteuerten Prozesskette (kurz: eEPK) und bildet diese auf Systemmodell-Aktivitäten eines BPMN-Diagramms ab. Untergliedert wird das Abbildungsmodell durch BPMN-Lanes.

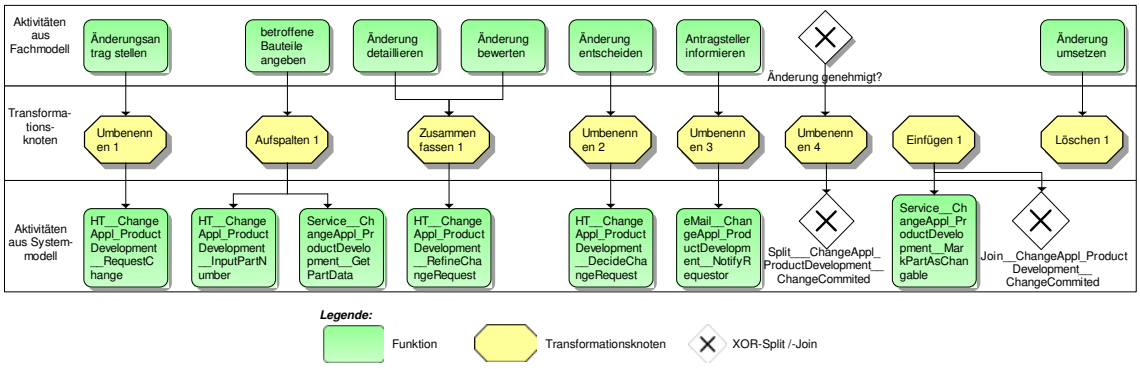


Abbildung 9 Abbildungsmodell in BPMN-Notation erstellt mit IDS-Scheer ARIS

Die Übersichtlichkeit der Darstellung resultiert u.a. daraus, dass von vorhandenen Objekttypen eigene Sub-Typen mit selbstdefiniertem Symbol abgeleitet wurden. Diese Funktionalität wird für die Transformationsknoten verwendet, um die Symbolik aus Abbildung 4 zu realisieren. Zudem lässt sich ARIS so konfigurieren, dass beliebige Attribute dargestellt werden. Dadurch wird es möglich, auch für Strukturknoten (z.B. für Verzweigungen) eindeutige Namen einzublenden. Nicht möglich ist es, eine spezielle Transformationskante zwischen fachlichen Aktivitäten und Transformationsknoten sowie Transformationsknoten und Aktivitäten aus dem Systemmodell zu definieren. Deshalb wird der „normale“ Kantentyp „folgt auf“ verwendet. Werden in ARIS Berichte erzeugt, etwa über einzelne Funktionen, so kann nicht zwischen Transformationen im Abbildungsmodell und Nachfolgebeziehungen von Aktivitäten im Prozessablauf unterscheiden werden. Dieses Problem tritt aber ebenso beim WebSphere Business Modeler auf und könnt dadurch gelöst werden, dass vom Modellierungswerkzeug ein eigener Modelltyp für Abbildungsmodelle mit speziellen Knoten- und Kantentypen angeboten wird.

Sehr einfach möglich ist in ARIS die Referenzierung von Aktivitäten des Fach- und Systemprozesses im Abbildungsmodell: Durch das ARIS-Objektkonzept existiert jedes in einem Diagramm modellierte Objekt einmalig in der ARIS-Objektdatenbank. Dadurch können Aktivitäten, die aus dem fachlichen Modell oder dem Systemmodell-Prozess stammen, einfach als Ausprägungskopien in das Abbildungsmodell kopiert werden. Da es sich bei diesen Ausprägungskopien um dasselbe Datenbank-Objekt handelt, wirken sich Änderungen jeweils auf alle Ausprägungen des Objektes aus. Dadurch bleiben Aktivitätennamen und andere Attribute stets aktuell.

Ein weiterer Vorteil von ARIS für die Realisierung des Abbildungsmodells ist, dass in verschiedenen Diagrammen erstellte Kanten sich nicht beeinträchtigen und auch keinen Datenfluss beschreiben. Dadurch tritt das beim WBM skizzierte Problem nicht auf, dass Transformationskanten mit einem bestimmten Ausgabeparameter der fachlichen Aktivität verbunden werden müssen. Ebenso ist im Abbildungsmodell nicht sichtbar, ob für die Aktivität weitere Ausgabeparameter definiert wurden, da die entsprechenden Satellitenobjekte ausschließlich im Fachprozess existieren. Allerdings muss beachtet werden, dass die zuletzt genannten Schwierigkeiten beim WBM nur deshalb auftreten, weil wir einen existierenden Diagrammtyp zweckentfremdet haben. Dies ist bei ARIS leichter möglich, weil die Modellierung hier weniger formal und mit keiner Ausführungssemantik belegt ist. Der Grund hierfür ist, dass das Werkzeug nicht für ein IT-Design, sondern für die fachliche Modellierung, vorgesehen ist.

Dies stellt aber wiederum ein Problem für die Verwendung von ARIS dar, weil das ARIS ist als Werkzeug bei Anwendern aus IT-Bereichen kaum verbreitet und für die Erstellung eines Systemmodells wenig geeignet. So ist kaum zu erwarten, dass ein IT-Architekt das Design von Klassen für Datenobjekte in ARIS durchführen wird, um so die IT-Spezifikation zu entwickeln.

5.3. Fazit

Beide untersuchten Modellierungswerkzeuge haben gezeigt, dass ein Abbildungsmodell als BPMN-Diagramm realisiert werden kann. Das Ergebnis war auch jeweils recht übersichtlich und mit geringem Aufwand erstellbar: Die Modellierung eines Transformationsknotens mit den zugehörigen Kanten und der Verwendung existierender Aktivitäten aus Fach- bzw. Systemmodell-Prozess ist in Sekunden bis maximal wenigen Minuten möglich. Deshalb eignen sich beide Werkzeuge für die Erstellung von Abbildungsmodellen und man wird jenes auswählen, welches für die restlichen Diagrammtypen des Systemmodells besser geeignet ist.

Trotz der prinzipiellen Verwendbarkeit können BPMN-Diagramme eigentlich nur eine Übergangslösung darstellen. Sie sollten solange verwendet werden, bis in Prozessmodellierungswerkzeugen eigene Diagrammtypen für Abbildungsmodelle zur Verfügung stehen. Damit lösen sich dann auch alle Probleme, die aus nicht realisierbaren Ausprägungskopien oder zweckentfremdeten Kontroll- und Datenflusskanten resultieren, da dem Modellierungswerkzeug dann die Bedeutung der Inhalte eines Abbildungsmodells bekannt ist.

6. Verwandte Arbeiten

In diesem Kapitel werden verwandte Ansätze diskutiert, die eine Transformation von abstrakt formulierten Geschäftsprozessen in ausführbare Modelle realisieren. Zunächst betrachten wir Ansätze, die Transformationen zwischen unterschiedlichen Meta-Modellen realisieren. Anschließend werden Ansätze wie MDA (Model-Driven Architecture) und MDSD (Model-Driven Software Development) sowie Modellierungsmethoden diverser Hersteller diskutiert.

Transformation von Meta-Modellen: In der Literatur werden zwei grundsätzlich verschiedene Arten von (Meta-) Modelltransformationen diskutiert: Zum einen die direkte Transformation von fachlichen Modellen (bspw. eEPK oder BPMN) in ausführbare Modelle (bspw. BPEL oder BPMN) und zum anderen die Transformation über eine zusätzliche Modellebene (bspw. eEPK oder BPMN).

Im ersten Fall wird meist durch Restriktionen (bspw. Zyklenfreiheit) das fachliche Modell eingeschränkt. [ZM05] nutzt etwa das XML-basierte Austauschformat für eEPK-Modelle EPML [MN06] als Basis, um diese in BPEL-Modelle zu transformieren. Das eEPK-Modell wird dabei soweit eingeschränkt, dass eine direkte Abbildung auf ein BPEL-Modell realisiert werden kann. Werden Änderungen am fachlichen Modell durchgeführt, bspw. durch das Einfügen eines neuen eEPK-Element, muss der Ansatz erneut angewendet werden. Dadurch gehen Änderungen die auf der technischen Ebene (BPEL) bereits durchgeführt wurden verloren. Dennoch liefert dieser Transformationsansatz einen Basismechanismus zur Überführung von azyklischen eEPK-Modellen nach BPEL.

Es existieren weitere ähnliche Ansätze: So wird in [Kop05] ebenfalls eine Transformation von eingeschränkten eEPK-Modellen nach BPEL beschrieben. In [AL05] wird dargestellt, wie Workflow-Netze nach BPEL transformiert werden können. [Ga03] beschreibt eine automatische Abbildung von UML nach BPEL. Basistransformationen von BPMN nach BPEL werden bereits im BPMN-Standard [OMG09] und in [Whi05] sowie in [OADH06] beschrieben. Letzterer Ansatz detailliert dabei eine Transformation die ein nicht eingeschränktes BPMN-Modell nach BPEL überführt. Mittels des im Ansatz beschriebenen Algorithmus werden die Modelle automatisch in BPEL überführt. Der Algorithmus durchsucht dabei das BPMN-Modell nach bestimmten Mustern und ersetzt diese durch speziell definierte Komponenten, die eine direkte Abbildung auf BPEL ermöglichen.

In der anderen bereits erwähnten Klasse von Ansätzen, wird ein Zwischenmodell zur Überführung fachlicher in ausführbare Modelle eingeführt: [TLD+07] beschreibt die Überführung eines eEPK-Modells in BPEL und nutzt dabei ein zusätzliches BPMN-Modell als Zwischenmodell. Die Idee hierbei ist, analog zu unserem Ansatz, die mangelnde Durchgängigkeit zwischen Fachmodell und ausführbaren Modell in den Griff zu bekommen und somit das Business-IT-Alignment zu erhöhen. Es ist jedoch nicht das Ziel, die Beziehung zwischen Aktivitäten unterschiedlicher Ebenen transparent zu machen. Den Ausgangspunkt für die Transformation liefert dabei die Ablauflogik aus dem fachlichen Modell (eEPK), welche mittels zuvor zu definierenden Spachkonstrukten in ein BPMN-Modell überführt wird. Anschließend wird das BPMN-Modell durch Anreicherung technischer Details zur Prozessausführung erweitert und in BPEL überführt. Ähnlich zu diesem Ansatz geht [KK05] vor, verwendet jedoch als Zwischenmodell eine formalisierte Version einer eEPK mit gewissen Einschränkungen.

MDA-basierte Ansätze: Zahlreiche verwandte Arbeiten basieren auf standardisierten Ansätzen wie MDA und MDSD.

[All07] beschreibt eine notationsunabhängige Vorgehensweise, die mittels Modelltransformationen aus grob granularen, fachlichen Geschäftsprozessen detaillierte und ggf. ausführbare Modelle generiert. Um eine solche Transformation zu realisieren, werden für das Ausgangsmodell (eEPK) verschiedene Muster (Patterns) definiert, die fachliche Objekte detaillieren und die beschreiben, wie diese in das Zielmodell (BPMN) überführt werden können. Anschließend werden Transformationsregeln definiert, die unter Verwendung von Patterns, die Transformation durchführen.

[Bau08] beschreibt, wie fachliche und technische Patterns definiert werden sollten, damit sie auf verschiedene Prozesse möglichst gut anwendbar sind. Dazu werden in einem ersten Schritt fachliche Patterns auf grob granulare Prozessmodelle (BPMN) angewendet. Auf Basis des resultierenden Modells (erweitertes BPMN), wird durch eine automatische Transformation ein ausführbares Grundmodell (sog. Pseudo-BPEL) erzeugt. Dieses wird in einem weiteren Schritt, unterstützt durch zusätzlich zu definierende technische Patterns, in ein ausführbares Modell (BPEL) überführt.

Das Projekt OrVia [FKT08] verfolgt einen ähnlichen Ansatz, indem eine Methoden- und Werkzeuggestützte Abbildung von eEPK-Prozessmodellen auf ausführbare BPEL-Modelle automatisiert realisiert wird. Konkret werden solche Prozessstrukturen in fachlichen Prozessmodellen identifiziert für die vordefinierte Patterns existieren. Anschließend wird eine Transformation der Prozessstruktur auf Basis der Patterns durchgeführt.

MDA-basierte Ansätze wenden Patterns auf fachliche Modelle an. Allerdings sind solche Ansätze zur Generierung ausführbarer Modelle nicht immer realisierbar. Dies trifft auch auf unser Szenario zu, bei dem eine *freie* Modellierbarkeit des Systemmodells gefordert wird: So ist für Geschäftsprozesse, die fachlich meist grob, sehr abstrakt und vage beschrieben sind, eine strukturelle Adaption dieser fachlichen Prozesse auf Systemmodell-Ebene notwendig. Eine solche Überarbeitung mittels automatisch anwendba-

ren Patterns zu realisieren, wäre zu aufwendig, da umfangreiche Transformationen fachlicher Objekte (z.B. Aktivitäten) auf Objekte im Systemmodell festgelegt werden müssten. Zudem sind Transformationen, wie das Einfügen von zusätzlichen Aktivitäten (vgl. Abschnitt 2), schwer realisierbar, da für diese keine entsprechende Aktivität im fachlichen Modell existiert. In dem von uns betrachteten Anwendungsfall sind Geschäftsprozesse unterschiedlich, so dass eine Wiederverwendung vordefinierter Patterns in unterschiedlichen Geschäftsprozessen nicht realistisch ist. Deshalb muss für jeden Geschäftsprozess individuell entschieden werden, wie eine entsprechende technische Repräsentation im Systemmodell aussieht.

Neben Pattern-basierten MDA-Ansätzen, existieren Service-orientierte Ansätze, die eine Entwicklung von Informationssystemen modellgetrieben unterstützen [CMW09]. Darüber hinaus existieren Ansätze die beschreiben, wie Modelle in eine andere Notation transformiert werden. [ODA+09] stellt eine Technik dar, mir der sich BPMN-Modelle in lesbare (Block-strukturierte) BPEL-Modelle transformieren lassen. Diese Transformation ist nur in eine Richtung bestimmt, wodurch es zu Inkonsistenzen kommt, sobald Änderungen am BPEL-Modell vorgenommen werden. Weitere Ansätze, wie etwa [WWM09], beschreiben wie Änderungen an BPMN-Modellen auf unterschiedlichem Abstraktionsniveau nachvollzogen werden können. Arbeiten aus dem Requirements Engineering beschäftigen sich mit der bidirektionalen Propagierung von Änderungen an Anforderungen und zugehörigen UML-Modellen [Mar05, Rup07].

Herstellermethoden: Um eine freie Modellierbarkeit des Systemmodells zu ermöglichen, empfehlen Hersteller von Modellierungswerkzeugen in der Regel eine andere Vorgehensweise. Ebenso wie in unserem Ansatz vorgeschlagen, führen sie ein zusätzliches Modell zwischen fachlicher und technischer Ebene ein.

Die Modellierungsmethodik M3 der Firma MID basiert auf der MDA, wobei drei Varianten unterschieden werden. Zum Vergleich mit unserem Ansatz ist insbesondere die Modellierungsmethodik M3 für SOA interessant [PR05]. Diese bietet für jeden Modelltyp, von der Geschäftsprozessmodellierung bis hin zur technischen Implementierung, Profilerweiterungen für Innovator [MID08] in Form von UML „Stereotypen“. Der Grundgedanke dabei ist, die Modellierung aller Ebenen und Aspekte von fachlichen Geschäftsprozessen bis zur entsprechenden IT-Implementierung abzudecken [PR05]. Prinzipiell wird dabei in drei unterschiedlichen Ebenen modelliert. Die erste Ebene beschreibt das fachliche Modell, in welchem Geschäftsprozesse sowie andere Objekttypen (z.B. Datenobjekte) ohne Einschränkungen frei modellierbar sind. Anschließend werden aus der Geschäftsprozessbeschreibung Anwendungsfälle abgeleitet. Letztere beschreiben die Anforderungen an das zu entwickelnde Informationssystem. Auf Basis dieser Anwendungsfälle und zusätzlicher Information aus den Geschäftsprozessen entsteht ein plattformunabhängiges „Analysemodell“ als Ebene zwei, vergleichbar mit unserem Systemmodell. Dies beschreibt bspw. Klassen- und Datenmodelle sowie Ablaufbeschreibungen die zur Implementierung der Geschäftsprozesse notwendig sind. In der dritten Modellebene wird ein plattformspezifisches Modell beschrieben, welches Zielplattform und -sprache festlegt, sowie dies um technische Informationen ergänzt.

Andere Hersteller wie bspw. IBM [AGA+08, Ars04], IDS Scheer [Dee07, BEH+08] oder Quasar Enterprise [EV08] beschreiben ähnliche Vorgehensmethoden, um fachliche Prozessmodelle in die IT-Implementierung zu überführen. Bei diesen werden ebenfalls mehrere Modellebenen verwendet, um eine solche Überführung schrittweise durchführen zu können. Allerdings bemüht sich keine dieser Methodiken darum, die Abhängigkeiten zwischen den Aktivitäten unterschiedlicher Modellebenen nachvollziehbar zu dokumentieren.

Fazit: Wir setzen in unserem Projekt ENPROSO auf diesen Ansätzen auf, um bei dem Modellwechsel einzelner Ebenen, bspw. von fachlicher Ebene zur Systemmodell Ebene, diese Ansätze als grundlegende Transformationen zu verwenden. In einigen Ansätzen wurde die Notwendigkeit eines Zwischenmodells erkannt und eine entsprechende Notations- bzw. Werkzeug-abhängige, Transformation über die einzelnen Ebenen realisiert. Ein Abbildungsmodell zur Sicherstellung der Nachvollziehbarkeit zwischen den einzelnen Ebenen wird in keinem Ansatz diskutiert.

7. Zusammenfassung und Ausblick

Von Fachbereichen erstellte Prozessmodelle müssen strukturell adaptiert werden, bevor sie mittels Prozess-Management-System implementiert werden können. Hierbei ist es das Ziel, möglichst schnell und nachvollziehbar von fachlichen Anforderungen zur IT-Implementierung zu gelangen (Business-IT-Alignment). Um dies zu erreichen, haben wir zwischen dem fachlichen und dem ausführbaren Prozessmodell die Ebene des Systemmodells eingeführt, das in Verantwortung des IT-Bereichs liegt und als Spezifikation für die IT-Implementierung dient. Um bei der Modellierung des Systemmodells die durchgeführten Prozesstransformationen nachvollziehbar zu gestalten, haben wir das Konzept des Abbildungs-

modells entwickelt. Es erlaubt, die bei der Erstellung des Systemmodells benötige „freie Geschäftsprozess-Modellierung“ weiterhin zu ermöglichen, aber dennoch einen hohen Grad an Nachvollziehbarkeit zu erzielen. Diese Nachvollziehbarkeit wird genutzt, um Prozessimplementierungen schneller erstellen bzw. anpassen zu können. Sie ermöglicht zudem die Konsistenz zwischen den verschiedenen Modellebenen sicher zu stellen.

Die ausgewählte Variante zur Realisierung eines Abbildungsmodells wird in diesem Beitrag detailliert sowie prototypisch beschrieben. Für einen Beispielprozess wird das Prozessmodell auf fachlicher Ebene, Systemmodell-Ebene und das zugehörige Abbildungsmodell im WebSphere Business Modeler umgesetzt (zum Vergleich ebenfalls im ARIS, siehe Kapitel 5.2), mit der Erkenntnis, dass Abbildungsmodelle leicht als BPMN-Diagramme realisiert werden können: Der Aufwand für die Modellierung einer einzelnen Transformationsbeziehung ist vernachlässigbar, da die hierbei notwendigen Abstimmungen mit dem Fachbereich den größeren Aufwand darstellen. Als nächster Schritt soll die Methodik nun in einem realen Pilotprojekt erprobt werden.

Wir haben gezeigt, wie die Flexibilität von Prozessapplikationen erhöht werden kann, indem man schneller und besser von fachlichen Anforderungen zur IT-Implementierung gelangt. Der Ansatz stellt die Basis dar, um die Flexibilität der resultierenden IT-Implementierung selbst zu erhöhen: So können im Sinne eines *Frontloading* Stellen im Fachprozess markiert werden, an denen ein hoher Flexibilitätsbedarf besteht, etwa häufig veränderte Verzweigungsbedingungen. Das Abbildungsmodell erlaubt es nun, diejenigen Aktivität(en) des Systemmodells zu identifizieren, die diese Flexibilität realisieren müssen. Die benötigte Art von Flexibilität kann realisiert werden, indem z.B. an der Prozessverzweigung eine Geschäftsregel verwendet wird, die zur Laufzeit des Workflows noch anpassbar ist. Solche Möglichkeiten zur Erhöhung der Flexibilität werden wir im Projekt ENPROSO zukünftig detailliert betrachten.

Literatur

[Agr07] A. Agrawl et al.: WS-BPEL Extension for People Specification. Technical Report, Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG, 2007

[AGA+08] A. Arsanjani; S. Ghosh; A. Allam; T. Abdollah; S. Ganapathy; K. Holley: SOMA - A method for developing service-oriented solutions; In: IBM Systems Journal 47, 2008

[AHK03] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, und A.P. Barros: Workflow Patterns. Distributed and Parallel Databases, 14(3), pages 5-51; 2003.

[AL05] W. M. P. van der Aalst und K. B. Lassen: Translating Workflow Nets to BPEL4WS. Bericht BPM-05-16, BPM Center, Eindhoven, Netherlands, 2005.

[All07] T. Allweyer: Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen; Proc.: 6. Workshop Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, St. Augustin, Germany, S. 23-38, 2007

[Ars04] A. Arsanjani: Service-oriented modeling and architecture; IBM developer works, 2004

[CMW09] V. de Castro, E. Marcos, R. Wieringa: Towards a Service-Oriented MDA-Based Approach to the Alignment of Business Processes with IT Systems: from the Business Model to a Web Service Composition Model, Int. J. Cooperative Inf. Syst., 2009

[Bau08] P. Bauler et.al.: Usage of Model Driven Engineering in the context of Business Process Management; In: Proc. 4th GI Workshop XML Integration and Transformation for Business Process Management, S. 1963 – 1974, 2008

[BBP09] S. Buchwald, T. Bauer, R. Pryss: IT-Infrastrukturen für flexible, service-orientierte Anwendungen - ein Rahmenwerk zur Bewertung; In: Proc. 13. GI-Fachtagung Datenbanksysteme in Business, Technologie und Web, S. 524–543, Münster, 2009

[BEH+08] A. Bösl, J. Ebell, S. Herold, C. Linsmeier, D. Peters, A. Rausch: Modellbasierte Software-Entwicklung von Informationssystemen: Vom Geschäftsprozess zum servicebasierten Entwurf, White Paper; 2007

[BOC08] BOC-Gruppe. Das Geschäftsprozessmanagement-Werkzeug ADONIS, White Paper; 2008.

[BTC09] BTC Business Technology Consulting: BONAPART Professional, White Paper; 2009.

[Che08] H.-M. Chen: Towards Service Engineering, Service Orientation and Business-IT Alignment; In: Proc. 41st Hawaii Int. Conf. on System Sciences, 2008

[Dee07] M. Deeg: SOA Fängt weit vor BPEL an - Serviceorientierte Geschäftsprozessmodellierung als Basis für eine SOA. In: Objektspektrum, 2007

[End09] R. Enderle: Frühe fachliche Modellierung ausführungsrelevanter Prozess-Aspekte, Prozessmodellierung in Zeiten von SOA. Diplomarbeit, Fakultät für Ingenieurwissenschaften und Informatik, Institut für Daten-

banken und Informationssysteme, 2009

[Erl05] T. Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall Verlag, 2005

[EV08] G. Engels, M. Voss: Quasar Enterprise. In: Informatik Spektrum 31, S. 548 – 555, 2008

[FKT08] K.P. Fähnrich; S. Kühne; M. Thränert: Model-Driven Integration Engineering - Modellierung, Validierung und Transformation zur Integration betrieblicher Anwendungssysteme; Eigenverlag Leipziger Informatik-Verbund (LIV), 2008

[Ga03] T. Gardner: UML Modelling of Automated Business Processes with a Mapping to BPEL4WS; In: Proc. of the First European, Workshop on Object Orientation and Web Services at ECOOP; 2003

[IBM05] IBM, WebSphere MQ Workflow Workflow: Getting Started with Buildtime. Version 3.6, Produktdokumentation, Document Number SH12-6286-10, 2005

[IBM08] IBM, WebSphere Business Modeler, Version 6.2, White Paper, 2008

[IBM09] IBM, WebSphere Process Server, Version 6.2, White Paper, 2008

[Jab97] S. Jablonski: Architektur von Workflow-Mangement-Systemen. Informatik Forschung und Entwicklung, Themenheft Workflow-Management, 12(2):72–81, 1997

[Jos07] N.M. Josuttis: SOA in Practice - The Art of Distributed System Design. O'Reilly; 2007

[Mar05] F. Marschall: Modelltransformationen als Mittel der modellbasierten Entwicklung von Software-Systemen. Dissertation, Institut für Informatik, TU München, 2005

[KK05] T. Kahl, F. Kupsch; Transformation und Mapping von Prozessmodellen in verteilten Umgebungen mit der ereignisgesteuerten Prozesskette; In: Nüttgens, Markus; Rump, Frank J. (Hrsg.): 4. Workshop EPK 2005 : Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten; Hamburg; 2005

[Kop05] O. Kopp; Abbildung von EPKs nach BPEL anhand des Prozessmodellierungswerkzeugs Nautilus; Diplomarbeit Nr. 2341, Universität Stuttgart, Institut für Architektur von Anwendungssystemen; 2005

[MID08] MID: Innovator Object – Objektorientiertes Software Engineering mit der UML, White Paper; 2008

[Mit05] K. Mittal: Introducing IBM Rational Software Architect. Technical Report, IBM; 2005.

[MN06] J. Mendling, M. Nüttgens: EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC), Inf. Syst. E-Business Management, 4, 245-263; 2006

[OADH06] C. Ouyang, W. M. P. van der Aalst, M. Dumas, A.H.M. Hofstede; Translating BPMN to BPEL, BPM Center Report, BPM-06-02; 2006

[OAS07] OASIS: Web Services Business Process Execution Language Version 2.0. OASIS Standard, 2007

[ODA+09] C. Ouyang, M. Dumas, W.M.P. Aalst, A.H.M. ter Hofstede, J. Mendling: From Business Proccess Model to Process-Oriented Software Systems, ACM Trans. Softw. Eng. Methodol, 2009

[OMG09] OMG (Hrsg): Business Process Model and Notation (BPMN) Specification 2.0, V0.9.14, revised submission draft, 2009

[PR05] O. Pera, B. Rintelmann; Von betrieblichen Geschäftsprozessen zu einer SOA; 18. Deutsche ORACLE-Anwenderkonferenz, 2005

[RD00] M. Reichert, P. Dadam: Geschäftsprozessmodellierung und Workflow-Management - Konzepte, Systeme und deren Anwendung, GITO-Verlag, Industrie Management, 2000

[Rei00] M. Reichert: Dynamische Ablaufänderungen in Workflow-Management-Systemen. Dissertation, Universität Ulm, Fakultät für Informatik; 2000

[RRD09] M. Reichert, S. Rinderle-Ma, P. Dadam: Flexibility in Process-aware Information Systems. Springer, LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC), Special Issue on Concurrency in Process-aware Information Systems, 2009

[Rup07] C. Rupp: Requirements-Engineering und Management. Hanser, 2007

[Sch98] A.-W. Scheer: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. Springer-Verlag, 1998

[TLD+07] O. Thomas, K. Leyking, F. Dreifus, M. Fellmann, P. Loos: Serviceorientierte Architekturen: Gestaltung, Konfiguration und Ausführung von Geschäftsprozessen. Veröffentlichungen des Instituts für Wirtschaftsinformatik; 2007

[Wes07] M.Weske: Business Process Management - Concepts, Languages, Architectures. Springer; 2007

[WSR09] B. Weber, S. Sadiq, M. Reichert: Beyond Rigidity - Dynamic Process Lifecycle Support: A Survey on Dynamic Changes in Process-aware Information Systems. Springer, Computer Science - Research and Development, Vol. 23, 2009

[WWM09] M. Weidlich, M. Weske, J. Mendling: Change Propagation in Process Models using Behavioural Profiles, IEEE International Conference on Services Computing, 2009

[Whi05] S. White: Using BPMN to Model a BPEL Process, BPTrends; 2005

[ZM05] J. Ziemann und J. Mendling: EPC-Based Modelling of BPEL Processes: A pragmatic Transformation Approach. In MITIP 2005, Italy, 2005.s, Architectures. Springer; 2007

Anhang

In Kapitel 5 wurde eine prototypische Umsetzung des Abbildungsmodells mit IBM WebSphere Business Modeler und ARIS realisiert. Im Folgenden werden die zugehörigen Prozessmodelle der fachlichen und der Systemmodell-Ebene dargestellt. Diese entsprechen unmittelbar dem in Kapitel 1 eingeführten Szenario und den Prozessdarstellungen aus Abbildung 1.

Abbildung 10 und Abbildung 11 zeigen die BPMN-Realisierung für diese Prozessebenen im IBM WebSphere Business Modeler (WBM). Sie definieren die Aktivitäten, die im Abbildungsmodell (vgl. Abbildung 8) referenziert werden.

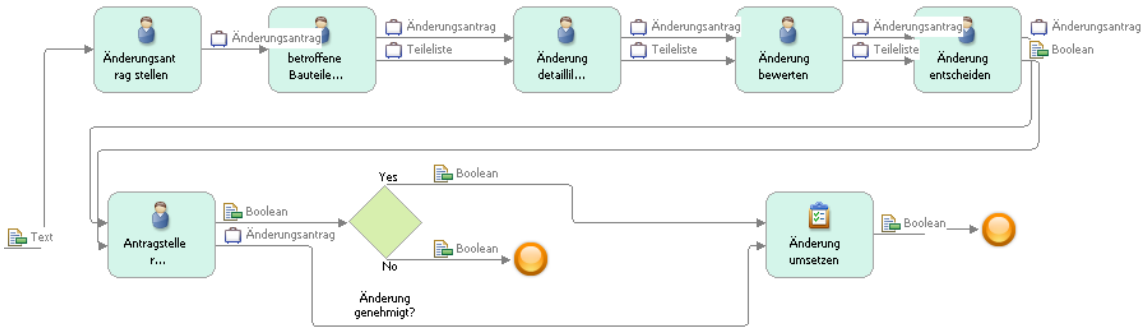


Abbildung 10 Prozess auf fachlicher Ebene im IBM WebSphere Business Modeler

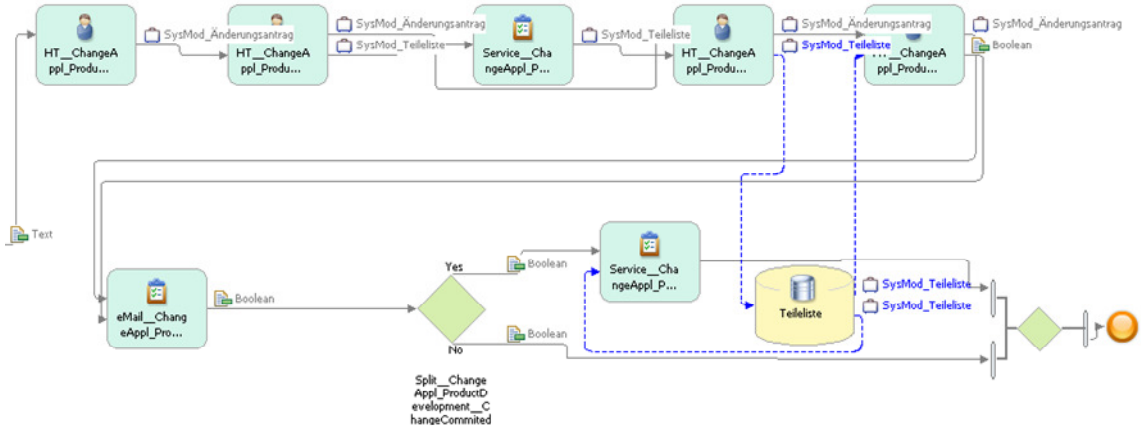


Abbildung 11 Prozess auf Systemmodell-Ebene im IBM WebSphere Business Modeler

Abbildung 12 zeigt den Fachprozess als im Modellierungswerkzeug ARIS erstellte eEPK. Die Verwendung der BPMN-Notation hätte keine Veränderungen für das Systemmodell (vgl. Abbildung 9) bedeutet, da in ARIS der identische Objekttyp „Funktion“ sowohl in eEPK- wie auch in BPMN-Diagrammen für Aktivitäten verwendet wird. Für das Systemmodell wurde in Abbildung 13 die BPMN-Notation verwendet.

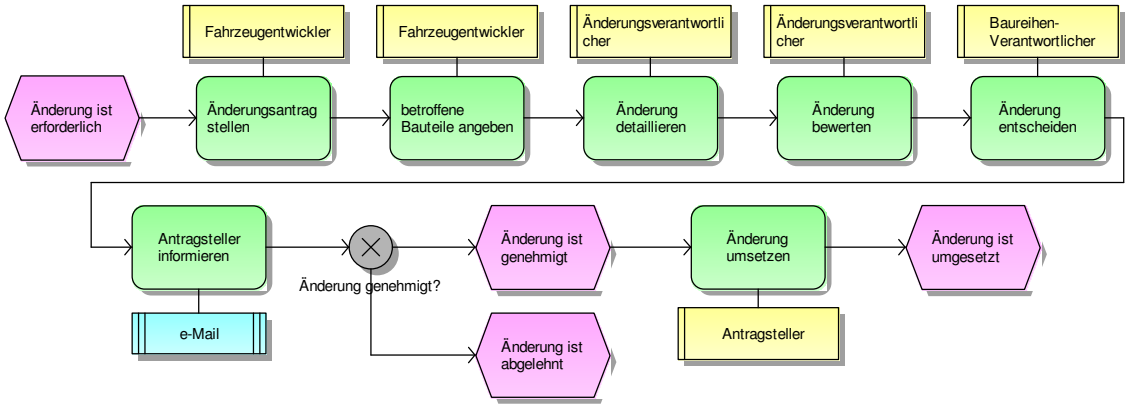


Abbildung 12 Prozess auf fachlicher Ebene in ARIS

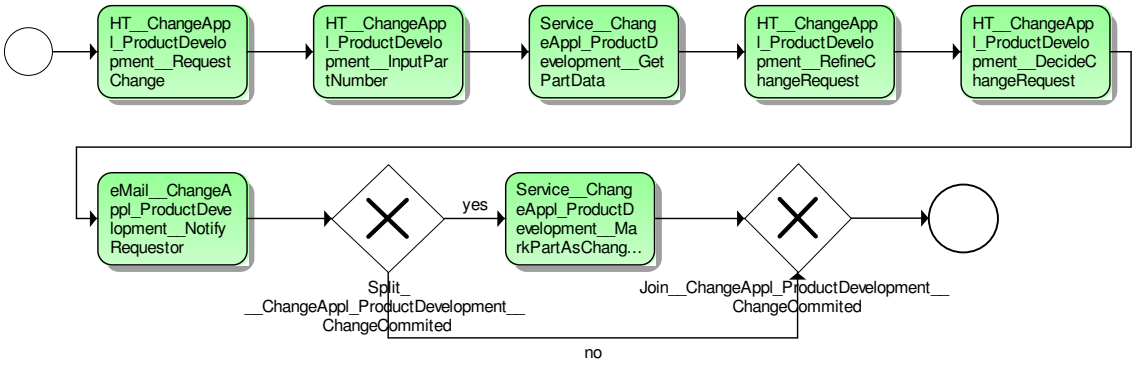


Abbildung 13 Prozess auf Systemmodell-Ebene in ARIS

Liste der bisher erschienenen Ulmer Informatik-Berichte

Einige davon sind per FTP von `ftp.informatik.uni-ulm.de` erhältlich
Die mit * markierten Berichte sind vergriffen

List of technical reports published by the University of Ulm

Some of them are available by FTP from `ftp.informatik.uni-ulm.de`
Reports marked with * are out of print

91-01 *Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe*
Instance Complexity

91-02* *K. Gladitz, H. Fassbender, H. Vogler*
Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03* *Alfons Geser*
Relative Termination

91-04* *J. Köbler, U. Schöning, J. Toran*
Graph Isomorphism is low for PP

91-05 *Johannes Köbler, Thomas Thierauf*
Complexity Restricted Advice Functions

91-06* *Uwe Schöning*
Recent Highlights in Structural Complexity Theory

91-07* *F. Green, J. Köbler, J. Toran*
The Power of Middle Bit

91-08* *V.Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf*
Reductions for Sets of Low Information Content

92-01* *Vikraman Arvind, Johannes Köbler, Martin Mundhenk*
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02* *Thomas Noll, Heiko Vogler*
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 *Fakultät für Informatik*
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04* *V. Arvind, J. Köbler, M. Mundhenk*
Lowness and the Complexity of Sparse and Tally Descriptions

92-05* *Johannes Köbler*
Locating P/poly Optimally in the Extended Low Hierarchy

92-06* *Armin Kühnemann, Heiko Vogler*
Synthesized and inherited functions -a new computational model for syntax-directed semantics

92-07* *Heinz Fassbender, Heiko Vogler*
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing

92-08* *Uwe Schöning*
On Random Reductions from Sparse Sets to Tally Sets

92-09* *Hermann von Hasseln, Laura Martignon*
Consistency in Stochastic Network

92-10 *Michael Schmitt*
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 *Johannes Köbler, Seinosuke Toda*
On the Power of Generalized MOD-Classes

92-12 *V. Arvind, J. Köbler, M. Mundhenk*
Reliable Reductions, High Sets and Low Sets

92-13 *Alfons Geser*
On a monotonic semantic path ordering

92-14* *Joost Engelfriet, Heiko Vogler*
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 *Alfred Lupper, Konrad Froitzheim*
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

93-02 *M.H. Scholl, C. Laasch, C. Rich, H.-J. Schek, M. Tresch*
The COCOON Object Model

93-03 *Thomas Thierauf, Seinosuke Toda, Osamu Watanabe*
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 *Jin-Yi Cai, Frederic Green, Thomas Thierauf*
On the Correlation of Symmetric Functions

93-05 *K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam*
A Conceptual Approach to an Open Hospital Information System

93-06 *Klaus Gaßner*
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 *Ullrich Keßler, Peter Dadam*
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 *Michael Schmitt*
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 *Armin Kühnemann, Heiko Vogler*
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 *Harry Buhrman, Jim Kadin, Thomas Thierauf*
On Functions Computable with Nonadaptive Queries to NP

94-04 *Heinz Faßbender, Heiko Vogler, Andrea Wedel*
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers

94-05 *V. Arvind, J. Köbler, R. Schuler*
On Helping and Interactive Proof Systems

94-06 *Christian Kalus, Peter Dadam*
Incorporating record subtyping into a relational data model

94-07 *Markus Tresch, Marc H. Scholl*
A Classification of Multi-Database Languages

94-08 *Friedrich von Henke, Harald Rueß*
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

94-09 *F.W. von Henke, A. Dold, H. Rueß, D. Schwier, M. Strecker*
Construction and Deduction Methods for the Formal Development of Software

94-10 *Axel Dold*
Formalisierung schematischer Algorithmen

94-11 *Johannes Köbler, Osamu Watanabe*
New Collapse Consequences of NP Having Small Circuits

94-12 *Rainer Schuler*
On Average Polynomial Time

94-13 *Rainer Schuler, Osamu Watanabe*
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14 *Wolfram Schulte, Ton Vullinghs*
Linking Reactive Software to the X-Window System

94-15 *Alfred Lupper*
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16 *Robert Regn*
Verteilte Unix-Betriebssysteme

94-17 *Helmuth Partsch*
Again on Recognition and Parsing of Context-Free Grammars:
Two Exercises in Transformational Programming

94-18 *Helmuth Partsch*
Transformational Development of Data-Parallel Algorithms: an Example

95-01 *Oleg Verbitsky*
On the Largest Common Subgraph Problem

95-02 *Uwe Schöning*
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03 *Harry Buhrman,Thomas Thierauf*
The Complexity of Generating and Checking Proofs of Membership

95-04 *Rainer Schuler, Tomoyuki Yamakami*
Structural Average Case Complexity

95-05 *Klaus Achatz, Wolfram Schulte*
Architecture Indepentent Massive Parallelization of Divide-And-Conquer Algorithms

95-06 *Christoph Karg, Rainer Schuler*
Structure in Average Case Complexity

95-07 *P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe*
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen

95-08 *Jürgen Kehrer, Peter Schulthess*
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09 *Hans-Jörg Burtschick, Wolfgang Lindner*
On Sets Turing Reducible to P-Selective Sets

95-10 *Boris Hartmann*
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper

95-12 *Klaus Achatz, Wolfram Schulte*
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13 *Andrea Mößle, Heiko Vogler*
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
A Generic Specification for Verifying Peephole Optimizations

96-01 *Ercüment Canver, Jan-Tecker Gayen, Adam Moik*
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE

96-02 *Bernhard Nebel*
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

96-03 *Ton Vullinghs, Wolfram Schulte, Thilo Schwinn*
An Introduction to TkGofer

96-04 *Thomas Beuter, Peter Dadam*
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering

96-05 *Gerhard Schellhorn, Wolfgang Ahrendt*
Verification of a Prolog Compiler - First Steps with KIV

96-06 *Manindra Agrawal, Thomas Thierauf*
Satisfiability Problems

96-07 *Vikraman Arvind, Jacobo Torán*
A nonadaptive NC Checker for Permutation Group Intersection

96-08 *David Cyrluk, Oliver Möller, Harald Rueß*
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction

96-09 *Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte*
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–Ansätzen

96-10 *Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Formalizing Fixed-Point Theory in PVS

96-11 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Mechanized Semantics of Simple Imperative Programming Constructs

96-12 *Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß*
Generic Compilation Schemes for Simple Programming Constructs

96-13 *Klaus Achatz, Helmuth Partsch*
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01 *Jochen Messner*
Pattern Matching in Trace Monoids

97-02 *Wolfgang Lindner, Rainer Schuler*
A Small Span Theorem within P

97-03 *Thomas Bauer, Peter Dadam*
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04 *Christian Heinlein, Peter Dadam*
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05 *Vikraman Arvind, Johannes Köbler*
On Pseudorandomness and Resource-Bounded Measure

97-06 *Gerhard Partsch*
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07 *Manfred Reichert, Peter Dadam*
$ADEPT_{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08 *Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler*
The Project NoName - A functional programming language with its development environment

97-09 *Christian Heinlein*
Grundlagen von Interaktionsausdrücken

97-10 *Christian Heinlein*
Graphische Repräsentation von Interaktionsausdrücken

97-11 *Christian Heinlein*
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12 *Gerhard Schellhorn, Wolfgang Reif*
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers

97-13 *Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn*
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 *Wolfgang Reif, Gerhard Schellhorn*
Theorem Proving in Large Theories

97-15 *Thomas Wennekers*
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 *Peter Dadam, Klaus Kuhn, Manfred Reichert*
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 *Mohammad Ali Livani, Jörg Kaiser*
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

97-18 *Johannes Köbler,Rainer Schuler*
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

98-01 *Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf*
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 *Thomas Bauer, Peter Dadam*
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 *Marko Luther, Martin Strecker*
A guided tour through *Typelab*

98-04 *Heiko Neumann, Luiz Pessoa*
Visual Filling-in and Surface Property Reconstruction

98-05 *Ercüment Canver*
Formal Verification of a Coordinated Atomic Action Based Design

98-06 *Andreas Küchler*
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 *Heiko Neumann, Thorsten Hansen, Luiz Pessoa*
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 *Thomas Wennekers*
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 *Thomas Bauer, Peter Dadam*
Variable Migration von Workflows in *ADEPT*

98-10 *Heiko Neumann, Wolfgang Sepp*
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 *Frank Houdek, Dietmar Ernst, Thilo Schwinn*
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment

98-12 *Gerhard Schellhorn*
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 *Gerhard Schellhorn, Wolfgang Reif*
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 *Mohammad Ali Livani*
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 *Mohammad Ali Livani, Jörg Kaiser*
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01 *Susanne Boll, Wolfgang Klas, Utz Westermann*
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02 *Thomas Bauer, Peter Dadam*
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03 *Uwe Schöning*
On the Complexity of Constraint Satisfaction

99-04 *Ercument Canver*
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05 *Johannes Köbler, Wolfgang Lindner, Rainer Schuler*
Derandomizing RP if Boolean Circuits are not Learnable

99-06 *Utz Westermann, Wolfgang Klas*
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07 *Peter Dadam, Manfred Reichert*
Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications. Paderborn, Germany, October 6, 1999, GI–Workshop Proceedings, Informatik ’99

99-08 *Vikraman Arvind, Johannes Köbler*
Graph Isomorphism is Low for ZPP^{NP} and other Lowness results

99-09 *Thomas Bauer, Peter Dadam*
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02 *Thomas Bauer, Peter Dadam*
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03 *Gregory Baratoff, Christian Toepfer, Heiko Neumann*
Combined space-variant maps for optical flow based navigation

2000-04 *Wolfgang Gehring*
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen

2000-05 *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06 *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07 *Gerhard Schellhorn, Wolfgang Reif (eds.)*
FM-Tools 2000: The 4th Workshop on Tools for System Design and Verification

2000-08 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09 *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10 *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11 *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01 *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02 *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03 *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04 *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02 *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03 *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01 *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values

2003-04 *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
On Dealing With Semantically Conflicting Business Process Changes.

2003-05 *Christian Heinlein*
Dynamic Class Methods in Java

2003-06 *Christian Heinlein*
Vertical, Horizontal, and Behavioural Extensibility of Software Systems

2003-07 *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values
(Corrected Version)

2003-08 *Changling Liu, Jörg Kaiser*
Survey of Mobile Ad Hoc Network Routing Protocols)

2004-01 *Thom Frühwirth, Marc Meister (eds.)*
First Workshop on Constraint Handling Rules

2004-02 *Christian Heinlein*
Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures

2004-03 *Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)*
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence

2005-01 *Armin Wolf, Thom Frühwirth, Marc Meister (eds.)*
19th Workshop on (Constraint) Logic Programming

2005-02 *Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven*
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm

2005-03 *Walter Guttmann, Markus Maucher*
Constrained Ordering

2006-01 *Stefan Sarstedt*
Model-Driven Development with ACTIVECHARTS, Tutorial

2006-02 *Alexander Raschke, Ramin Tavakoli Kolagari*
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen

2006-03 *Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari*
Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg

2006-04 *Thorsten Liebig*
Reasoning with OWL - System Support and Insights –

2008-01 *H.A. Kestler, J. Messner, A. Müller, R. Schuler*
On the complexity of intersecting multiple circles for graphical display

2008-02 *Manfred Reichert, Peter Dadam, Martin Jurisch,l Ulrich Kreher, Kevin Göser, Markus Lauer*
Architectural Design of Flexible Process Management Technology

2008-03 *Frank Raiser*
Semi-Automatic Generation of CHR Solvers from Global Constraint Automata

2008-04 *Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander*
Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse

2008-05 *Markus Kalb, Claudia Dittrich, Peter Dadam*
Support of Relationships Among Moving Objects on Networks

2008-06 *Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)*
WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke

2008-07 *M. Maucher, U. Schöning, H.A. Kestler*
An empirical assessment of local and population based search methods with different degrees of pseudorandomness

2008-08 *Henning Wunderlich*
Covers have structure

2008-09 *Karl-Heinz Niggl, Henning Wunderlich*
Implicit characterization of FPTIME and NC revisited

2008-10 *Henning Wunderlich*
On span-P^{cc} and related classes in structural communication complexity

2008-11 *M. Maucher, U. Schöning, H.A. Kestler*
On the different notions of pseudorandomness

2008-12 *Henning Wunderlich*
On Toda's Theorem in structural communication complexity

2008-13 *Manfred Reichert, Peter Dadam*
Realizing Adaptive Process-aware Information Systems with ADEPT2

2009-01 *Peter Dadam, Manfred Reichert*
The ADEPT Project: A Decade of Research and Development for Robust and Fexible Process Support
Challenges and Achievements

2009-02 *Peter Dadam, Manfred Reichert, Stefanie Rinderle-Ma, Kevin Göser, Ulrich Kreher, Martin Jurisch*
Von ADEPT zur AristaFlow® BPM Suite – Eine Vision wird Realität "Correctness by Construction" und flexible, robuste Ausführung von Unternehmensprozessen

2009-03 *Alena Hallerbach, Thomas Bauer, Manfred Reichert*
Correct Configuration of Process Variants in Provop

2009-04 *Martin Bader*
On Reversal and Transposition Medians

2009-05 *Barbara Weber, Andreas Lanz, Manfred Reichert*
Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

2009-06 *Stefanie Rinderle-Ma, Manfred Reichert*
Adjustment Strategies for Non-Compliant Process Instances

2009-07 *H.A. Kestler, B. Lausen, H. Binder H.-P. Klenk. F. Leisch, M. Schmid*
Statistical Computing 2009 – Abstracts der 41. Arbeitstagung

2009-08 *Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam*
Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen

2009-09 *Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch*
Progressive Point-Light-Based Global Illumination

2009-10 *Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler*
Boolean networks for modeling and analysis of gene regulation

2009-11 *J. Hanika, H.P.A. Lensch, A. Keller*
Two-Level Ray Tracing with Recordering for Highly Complex Scenes

2009-12 *Stephan Buchwald, Thomas Bauer, Manfred Reichert*
Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm