Optimisation and Evaluation of Software Support Systems

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

Vorgelegt von
Helmut Lang
aus Neumarkt i.d.OPf.

2015
Amtierender Dekan: Prof. Dr. Tina Seufert
Gutachter: Prof. Dr. Dr.-Ing. Wolfgang Minköp
Prof. Dr. Hans Peter Großmann
Tag der Promotion: 26.01.2016
Abstract

This thesis investigates the current status of help systems and user assistance available in the context of computer software usage. The focus of this thesis is on help available to users without explicitly referring to another person like a friend, a coworker, a helpline, or an e-mail support system. After introducing important concepts from the area of cognitive load theory and the usage of anthropomorphic agents in computer systems, existing help systems are reviewed with respect to their impact on users' cognitive load, and their usefulness is evaluated. Based on this evaluation we present an advanced help system implemented for a collaborative grid computing web portal. This help system is capable of assisting users at multiple levels in the interaction process and fosters efficient software usage. Following a worked examples based approach to introduce users to the User Interface (UI) of the portal, it points out essential elements of the UI required for task completion. Processing these worked examples, users quickly attain first results. Initial results are refined by users employing less important features of the UI. The functionality of these less prominent features is learned by users following an exploratory based approach. The knowledge acquired, following the worked examples, forms the foundation to guide users during this exploratory learning process. To assist users at this stage the help system is capable of providing additional information for certain elements of the UI. Additionally, interested users can be provided with information on related concepts not directly relevant for task completion by the help system.

Different aspects of the help system are evaluated with respect to users' objective performance and subjective acceptance of the help system. Overall six experiments were performed in the context of this thesis. Comparing our advanced help system to Hypertext Markup Language (HTML) based help pages showed the superiority of our help system with respect to objective performance. Another Comparison between a help system using an anthropomorphic agent and a system showing a generic question mark symbol to represent the help system, revealed that the kind of representation used for the system had virtually no influence on the subjective ratings. Two experiments comparing systems with different levels of proactivity revealed that in general users subjectively prefer reactive help systems over help systems featuring proactive behaviour. An experiment comparing a help system employing text and mouse based interaction to a system using spoken in- and output showed that users avoid using spoken input to interact with computers in public. Increasing social pressure on users by explicitly asking them to use the speech input feature could mitigate this effect and users made use of the feature in this case. The final experiment conducted evaluated social aspects of interaction with anthropomorphic agents. By using different modalities to present the questionnaires evaluating the system, we hoped to be able to replicate the results of Nass et al. [1994]. However, our results were not conforming to the results reported in Nass et al. [1994]. Accordingly,
we assume that users’ perspective on computers might have changed recently.

Our work contributes to the area of help systems and is a first integrated step in applying the findings of cognitive load theory in the context of computer help systems. It also identifies aspects of help systems that might have detrimental effects on user acceptance.
Danksagung

An erster Stelle möchte ich hier Professor Wolfgang Minker danken, für seinen Einsatz, seine Unterstützung, das sehr angenehme Arbeitsklima mit ihm und in seiner Abteilung sowie nicht zuletzt für seine Geduld bei der Entstehung dieser Arbeit.


Außerdem möchte ich zu guter Letzt noch meinen Eltern, Maria und Helmut Lang, sowie meiner Partnerin Ellen Kalchschmidt ganz herzlich für ihre Geduld und für ihre moralische Unterstützung vor allem während des Abschlusses dieser Arbeit danken.
Contents

I. Motivation, Theoretical Background and Related Work 1

1. Introduction 3
   1.1. Motivation .................................................. 6
      1.1.1. Frustration with Using Computers .................... 7
      1.1.2. How do Users Solve Software Problems? ............. 8
   1.2. Problem Setting ........................................... 11
   1.3. Proposed Solution – Integrated Advanced Help System ... 11
   1.4. Thesis Contributions ....................................... 12
   1.5. Document Structure ....................................... 13

2. Theoretical Background 15
   2.1. Learning Theory ......................................... 15
      2.1.1. Memory Models ........................................ 15
      2.1.2. Cognitive Load Theory ................................ 17
      2.1.3. Modality Principle ................................... 18
      2.1.4. Worked-Examples Effect ................................ 19
      2.1.5. Extraneous Processing in Multimedia Learning ........ 20
      2.1.6. Exploratory Learning and the Efficient Software Usage Process 20
   2.2. Persona Effect ............................................. 23
   2.3. Uncanny Valley ............................................. 24

3. Contemporary Help Systems and Resources for User Assistance 27
   3.1. Traditional Help Pages ................................. 27
      3.1.1. Help Pages Available Offline ....................... 28
      3.1.2. Hybrid Offline/Online Help Pages ..................... 31
      3.1.3. Help Pages Available Online .......................... 34
      3.1.4. General Drawbacks .................................... 34
   3.2. Tooltips ..................................................... 36
   3.3. ‘What’s this?’ .............................................. 37
   3.4. Extended Tooltips .......................................... 38
   3.5. Ubuntu HUD, macOS Sierra Help Menu Search, Office 2013 Offline Help 40
   3.6. Tips and Tours at Installation, Startup, or First Start 42
   3.7. Clippy the Paperclip ...................................... 46
   3.8. Virtual Agents on E-Commerce and Retail Web Sites ........ 49
   3.9. Wizards ..................................................... 51
   3.10. Crowdsourced Online Assistance ....................... 54
   3.11. StackExchange ............................................. 55
   3.12. Conclusion .................................................. 58
## II. Advanced Help System

### 4. Requirements of an Advanced Help System

4.1. From a User's Perspective ........................................... 63
4.2. From a Software Vendor's Perspective ......................... 65
4.3. Concept .............................................................. 66
  4.3.1. Formal Definition of Help Systems ......................... 67
  4.3.2. Help Systems .................................................. 71

### 5. The Prototype System ............................................. 73

5.1. Project Context .................................................... 73
  5.1.1. Grid Computing ................................................. 74
  5.1.2. The bwGRID Portal Project ................................. 76
  5.1.3. bwGRID Portal – Architectural Overview ................. 77
  5.1.4. Servlets .......................................................... 77
  5.1.5. JavaServer Pages .............................................. 79
  5.1.6. Tag-libraries .................................................... 80
  5.1.7. Portlets .......................................................... 80

5.2. Features of the Help System ..................................... 81
  5.2.1. General UI ....................................................... 82
  5.2.2. Step by Step Guidance for Novice Users ................ 85
  5.2.3. Assistance to Refine Initial Results ..................... 94
  5.2.4. Providing Further Information to Interested Users .... 97
  5.2.5. Spoken In- and Output ...................................... 100
  5.2.6. Proactivity ...................................................... 101
  5.2.7. Input Validation .............................................. 101
  5.2.8. Conclusion ...................................................... 101

5.3. Technical Realization ............................................. 102
  5.3.1. Architectural Overview ...................................... 102
  5.3.2. Server-Side Implementation .................................. 104
  5.3.3. Client-Side Implementation .................................. 115
  5.3.4. Spoken In- and Output ...................................... 116
  5.3.5. Documentation .................................................. 118
  5.3.6. Conclusion ...................................................... 121

### III. Evaluation and Experiments .................................. 123

6. General Methodology ............................................... 125
  6.1. Scenario .......................................................... 125
    6.1.1. Nuclear Power Plant Simulation ......................... 125
    6.1.2. Tasks .......................................................... 127
    6.1.3. Assistance .................................................... 131
  6.2. Participants ...................................................... 132
  6.3. Procedure .......................................................... 133
  6.4. Questionnaires ................................................... 134
    6.4.1. Before Interaction ......................................... 134
    6.4.2. After Interaction .......................................... 137
    6.4.3. Questionnaire Length ..................................... 141
  6.5. Objective Data .................................................... 142
### Contents

6.6. Statistical Testing .............................................. 145  
6.7. Influence of Personality Traits ................................. 145

7. Advanced Help System vs. the Traditional Approach 147  
7.1. Method .......................................................... 147  
7.1.1. Conditions ...................................................... 147  
7.1.2. Questionnaires and Objective Data ........................ 149  
7.1.3. Participants .................................................... 151  
7.2. Hypotheses .......................................................... 151  
7.3. Results .............................................................. 152  
7.3.1. Strategies ...................................................... 152  
7.3.2. Objective Scales .............................................. 152  
7.3.3. Subjective Scales ............................................. 155  
7.3.4. Changes in Self-efficacy .................................... 159  
7.4. Discussion .......................................................... 159  
7.4.1. Strategies ...................................................... 159  
7.4.2. Objective Scales .............................................. 159  
7.4.3. Subjective Scales ............................................. 162  
7.4.4. Changes in Self-efficacy .................................... 163  
7.5. Conclusion .......................................................... 163

8. Animated Agent vs. Depiction of a Question Mark 165  
8.1. Method .............................................................. 165  
8.1.1. Conditions ...................................................... 165  
8.1.2. Participants .................................................... 165  
8.1.3. Questionnaires ................................................ 166  
8.2. Hypotheses .......................................................... 167  
8.3. Results .............................................................. 167  
8.3.1. Objective Scales .............................................. 167  
8.3.2. Subjective Scales ............................................. 170  
8.4. Discussion .......................................................... 173  
8.4.1. Objective Scales .............................................. 173  
8.4.2. Subjective Scales ............................................. 174  
8.5. Conclusion .......................................................... 175

9. Reactive vs. Proactive Agent 177  
9.1. Method .............................................................. 178  
9.1.1. Conditions ...................................................... 178  
9.1.2. Participants .................................................... 178  
9.1.3. Questionnaires ................................................ 178  
9.2. Hypotheses .......................................................... 179  
9.3. Results .............................................................. 179  
9.3.1. Objective Scales .............................................. 179  
9.3.2. Subjective Scales ............................................. 182  
9.4. Discussion .......................................................... 184  
9.4.1. Objective Scales .............................................. 184  
9.4.2. Subjective Scales ............................................. 185  
9.5. Conclusion .......................................................... 187
# Contents

10. Reactive Agent vs. Deactivateable Proactivity  189  
10.1. Method ............................................. 189  
  10.1.1. Conditions ..................................... 189  
  10.1.2. Participants ................................... 190  
  10.1.3. Questionnaires and Objective Data ............. 190  
10.2. Hypotheses ....................................... 191  
10.3. Results .......................................... 191  
  10.3.1. Proactivity Strategies ......................... 191  
  10.3.2. Objective Scales ................................ 192  
  10.3.3. Subjective Scales .............................. 195  
10.4. Discussion ....................................... 197  
  10.4.1. Proactivity Strategies ......................... 197  
  10.4.2. Objective Scales ................................ 198  
  10.4.3. Subjective Scales .............................. 198  
10.5. Conclusion ...................................... 200  
11. Spoken Out- and Input vs. Text-based Interaction  201  
11.1. Method .......................................... 202  
  11.1.1. Conditions ..................................... 202  
  11.1.2. Participants ................................... 202  
  11.1.3. Procedure .................................... 202  
  11.1.4. Questionnaires and Objective Data .......... 203  
11.2. Hypotheses ....................................... 203  
11.3. Results .......................................... 203  
  11.3.1. Cognitive Load .................................. 203  
  11.3.2. Objective Scales ................................ 203  
  11.3.3. Speech Input Strategies ....................... 207  
  11.3.4. Subjective Scales .............................. 207  
11.4. Discussion ....................................... 212  
  11.4.1. Cognitive Load .................................. 212  
  11.4.2. Objective Scales ................................ 213  
  11.4.3. Speech Input Strategies ....................... 213  
  11.4.4. Subjective Scales .............................. 214  
11.5. Conclusion ...................................... 215  
12. Animated Agents as Social Actors  217  
12.1. Method .......................................... 218  
  12.1.1. Conditions ..................................... 218  
  12.1.2. Participants ................................... 220  
12.2. Hypotheses ....................................... 221  
12.3. Results .......................................... 221  
  12.3.1. Own Questionnaire ............................. 221  
  12.3.2. AttrakDiff .................................... 222  
12.4. Discussion ....................................... 224  
  12.4.1. Conclusion .................................... 225
## Contents

IV. Conclusion and Future Directions 227

A. Questionnaires 235
   A.1. Before Interaction 235
       A.1.1. Personal Data 235
   A.2. After Interaction 236
       A.2.1. Self Developed Subjective Evaluation Questionnaire 236

Bibliography 241

List of Figures 253

List of Tables 259

List of Own Publications 261

Curriculum Vitae 263
Acronyms and Initialisms

API  Application Programming Interface
CASA  Computers As Social Actors
CSUQ  Computer Usability Satisfaction Questionnaire
FAQ  Frequently Asked Questions
GUI  Graphical User Interface
HCI  Human Computer Interaction
HPC  High Performance Computing
HUD  Head-Up Display
HTML  Hypertext Markup Language
HTTP  Hypertext Transfer Protocol
LTM  Long Term Memory
MVC  Model–View–Controller
NEO-FFI  NEO-Five Factor Inventory
SASSI  Subjective Assessment of Speech System Interfaces
UI  User Interface
URL  Uniform Resource Locator
JSP  JavaServer Pages
JSR  Java Specification Request
OS  Operating System
TTS  Text-To-Speech
WIMP  Windows, Icons, Menus, Pointer
WM  Working Memory
WYSIWYG  What You See Is What You Get
XML  Extensible Markup Language
Part I.

Motivation, Theoretical Background and Related Work
1. Introduction

“A computer is a general purpose device” [Lewis, 1968]

This quotation summarises the dilemma computer users and software developers have been facing since the emergence of today’s computers and even before computers came into widespread use by the public. The possibilities offered by computers are virtually endless and to enable users to easily access this plethora of opportunities different layers of abstraction have been introduced. From using Assembler to give names to instructions, to modern programming languages, from command and control based to today’s Windows, Icons, Menus, Pointer (WIMP) interfaces the idea always was to make using computers easier and more beneficial for an ever increasing audience.

The approach chosen today to make computers tractable is to provide different pieces of software for different kinds of tasks. There is a specialized software to browse the internet, another program to manage e-mails, a program to create spreadsheets and so on. While this paradigm is not without its critics (see [Raskin, 2000] for an example), using different programs to accomplish different tasks is the predominant way to provide users with the capabilities they need to solve their task. This development was even accelerated recently in the context of smart phones. There, a whole new industry evolved providing users with extremely specialized programs or Apps offering very limited functionality to solve certain tasks. The advantage of this approach is that due to the limited range of functions it is usually very easy to learn how to use this software. However, due to the fact that writing such tiny pieces of software – compared for example to a full blown office suite – is rather easy, there tends to be a huge diversity of different programs from various vendors that cover a certain aspect. Currently there are for example more then 1.300.000 Apps available at the Google Play Store¹ and searching for example for a “Jogging” app brings up about 250 results making the selection of the best one for a specific purpose quite hard. Having found a suitable application users start using it and typically miss features they would like to have for this application. The running app is for example able to log the distance covered and to show the route taken on a map. So why shouldn’t it be used for cycling also? Of course using the App for cycling messes up the summary of your training progress for the running part. Thus, the software has to add a second mode for cycling, a third mode for Nordic walking, another mode for hiking, and so on. The next thing is, if there are maps anyway, why does the App not show running trials other users have been running at the given location, and show new routes that can be tried? And if the App shows new routes it should also be able to give directions, when running or cycling to guide users along these routes. Users also might want to share their progress

with their friends. So there should be an integrated functionality to upload training data to a social network.

From a software developer’s perspective it is not hard to integrate all this doubtless desirable features. However, from a user’s perspective adding this functionalities also adds complexity to the given program. Adding complexity also means to make its usage harder to learn and is thus contrary to the trend of specialization in software. Of course the software vendor could split the App into different parts, like one App for Nordic walking and another App for cycling, but especially if the Apps are not for free, users might feel deceived and reviews like “Bought the Nordic walking app, basically the same as the Jogging app. Total rip-off!” should definitely be avoided, since those reviews have significant influence on the success of such an App [Felt et al., 2012; Ha and Wagner, 2013].

Another aspect of this is that, when a user searches for a Jogging App and can choose between an App that he can merely use for running and another one that will also be usable for cycling and both Apps are available at the same price, the user will quite surely choose the more versatile App, even if he might not need the extra features provided by the App right now.

Norman [2002] calls this tendency to add more and more features to a product on request of users “creeping featurism” and suggests two paths to avoid it.

One of it is modularization which he describes as the creation of “separate functional modules, each with a limited set of controls, each specialized for some different aspect of the task” [Norman, 2002]. This approach perfectly fits the App approach, there are small pieces of software designed to serve certain purposes. However, as the example of the Jogging app shows, even the App approach has its limitations.

The second path is avoidance, what means limiting features to a bare minimum. However, at least to us, this seems not to be a feasible approach. Since, what is more annoying than using a software and realizing this software lacks a desperately needed feature? Take for example the plots in this thesis. Those were all generated using gnuplot [Williams et al., 2013]. When the author decided to add error bars to the histograms, he realized that histograms do not support to define a different color for borders and error bars. Thus it is not only not possible to have error bars without borders, but both also have to share the same color. The process of finding out that this limitation actually seems to be inherent to the histogram feature of gnuplot and implementing a more or less appropriate workaround was very time consuming. While this feature seems not to be essential for others, the author was rather frustrated about it. At such a point users have two options, they can either switch to using another program that offers the required feature and redo all their work or have to live with the given limitation. Either way, stumbling across a missing feature is always an annoyance and can have very time consuming consequences. Thus, we firmly believe that abandoning features for the sake of simplicity should not be the path taken to make software easier to use. Taking the initial quote “A computer is a general purpose device” into consideration, it is the responsibility of software developers to make as many of those purposes computers can serve, accessible to users in an easy to learn and efficient way.

How to make the usage of computers easier to learn is one of the major ambitions in
the field of Human Computer Interaction (HCI) research. Starting from punching cards over command and control based interfaces to today’s WIMP interfaces there has been a lot of progress in this field. Interfaces are often even described to be intuitive or natural to use. However, as Raskin [2000] states: “When examined closely, this concept turns out to vanish like the pea in a shell game and be replaced with the more ordinary but more accurate term ‘familiar’”. This emphasizes the fact that for an interface to really be intuitive users would not have to learn how to use the interface, but be able to use it without further knowledge or instruction. This is virtually never true for any kind of UI. Take for example the usage of a mouse. Motion of the mouse is translated to a corresponding motion of the mouse pointer on screen. The shape of the mouse is an excellent affordance [Norman, 2002] indicating that the mouse should be manipulated by hand. The existence of a button indicates that there is some kind of action that can be performed with this specific tool. Hence, from a usability perspective computer mice bring everything to be considered as natural and intuitive input device and using a mouse seems to be natural and intuitive to us. However, even in the case of a computer mouse users are required to have attained some knowledge concerning its proper use. Take for example a scene from the movie “Star Trek IV: The Voyage Home”. In this movie the Enterprise travels back in time to the year 1986 and in one scene Scotty, the chief engineer, is interacting with a computer system of that time. Scotty, used to the speech enabled interfaces of his time, starts talking to the computer. When this has no effect and he sees a small device connected to the computer and placed right next to it, he picks it up, presses the button, and starts talking into the mouse (see Figure 1.1). While this is definitely an odd and even comical behaviour to us, it is perfectly understandable from the perspective of a person not having learned the proper use of computer mice. Having a closer look at it, all affordances of the mouse, the shape – perfectly fitting into your hand – and the existence of a button – to activate and deactivate speech recognition, like
1. Introduction

with a walkie-talkie - even fit Scotty’s use better than its actual usage pattern. While this might be a fictitious example, it nevertheless demonstrates that even for the best UIs that follow all rules of good design, users have to gain additional knowledge in order to properly use them. For the example of the computer mouse this means, users had to eventually learn how to use it. However, this learning process is usually quite easy due to its excellent usability. Increasing the usability and making computers more intuitive and natural to use has been the focus of HCI research and focusing on this aspect, the second approach of assisting users in learning how to use these computers has been rather neglected due to various reasons. The most important of these reasons is surely that it is better to avoid a problem users are facing using a program than to teach them how to deal with problems and limitations induced by the status quo of current software development.

Users are having trouble using command based interfaces? Invent a different, easier to use interface that allows direct manipulation employing the WIMP metaphor.

Users are having trouble using complex multipurpose programs like word processors? Change the interface of the word processor to increase usability or split the program into smaller programs targeting a more specific purpose.

While it is of course important to ease access to computers introducing new technologies and interface paradigms, we argue that the aspect of teaching users how to use existing interfaces should be stressed more in the area of HCI. Especially since, as the example above shows, even the best UI can not be used without attaining prior knowledge and the problem of complexity is inherent to the computers and thus can and should not be avoided completely.

We believe that it is possible for users to efficiently work even with complex multipurpose software, as soon as this software offers a help system assisting users to tackle this complexity.

A help system, we consider to be adequate to solve this issue, will be introduced and evaluated within this thesis.

1.1. Motivation

Supporting users with software usage is a very complex field in the area of HCI. While it certainly involves knowledge gained in the area of learning theory it is important to realize that the objectives of a user learning how to use a new computer program might be inherently different to the objectives of a learner in other fields. Even though the mantra of schools around the world is “non scholæ sed vitæ discimus” (We do not learn for school, but for life) quite often the opposite is true and the original version of the quote as stated by Seneca [1969] “non vitæ sed scholæ discimus” ("We learn not for life but for classtime") better describes the motivation of learners. This means that at school and even at university students are not aware of the benefits they will gain in “real life” remembering the material taught. Up to now for example I have not come across the necessity to calculate the Householder transformation [Householder, 1958] of a matrix using a calculator and a piece of paper. However I can vaguely remember that this was
very important for a test in linear algebra during my bachelor studies.

This is usually totally different for users learning how to use new software. Of course there are also students taught how to use a certain software at school or university. However, those learners are surely not the majority and most users start using a particular software having a certain objective in mind. This could be as simple as “I want to write an e-mail to aunt Mary inviting her to my next birthday party” or as complex as “I need to train a Hidden Markov Model able to recognize spoken numbers from zero to nine”. Given that users typically have a well defined objective using computers, is a gift as well as a burden. It is a gift, since those users are inherently motivated and want to complete the task at hand and thus they are actually willing to learn. It is a burden since users having a certain task that needs to be solved, might get impatient if they are not able to finish their tasks. Every minute you are wasting sorting out marginalities with the software used, you do not have to complete useful stuff. Every minute you need to configure your mail client to be able to send mails, is a minute you can not use to write the mail you wanted. Thus, it is understandable that users get annoyed by software not supporting them at completing their task quickly and will get even more frustrated if they are not able to get their tasks done at all.

1.1.1. Frustration with Using Computers

Causes and severity of user frustration in the context of students have been investigated by Ceaparu et al. [2004] and for workplace related computer usage by Lazar et al. [2006]. Both studies present evidence that user frustration is a major problem in the context of computer usage. The time lost due to frustrating experiences was between 47 % and 53% of the overall time spent working with the computer for the first study [Ceaparu et al., 2004], and about 43 % in the workplace related second study [Lazar et al., 2006]. Both studies categorized frustrating experiences into five classes:

- Internet
- Applications
- Operating System
- Hardware
- Other

Table 1.1 shows a summary of specific causes for user frustration during computer usage as reported by Ceaparu et al. [2004]; Lazar et al. [2006].

Of course the results reported in Table 1.1 seem a bit dated and for example the huge values for crashes, freezes in both Applications as well as Operating System categories might not be what would be expected if the study was performed today. There also would quite surely not be a report of a “multitasking failure” incident. However, a study performed recently by Janneck and Guzalka [2013] with only slightly different methodology (not directly assessing “frustrating experiences” but “situations of failure”)
1. Introduction

<table>
<thead>
<tr>
<th>Internet related problems</th>
<th>Applications missing, hard to find, annoying, unusable features</th>
<th>Operating System crashes, freezes</th>
<th>Printing problems</th>
<th>Other user kicked from system</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>54</td>
<td>44</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>email related problems</td>
<td>crashes, freezes</td>
<td>48</td>
<td>unexpected response or behaviour</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>web page not found</td>
<td>unexpected response or behaviour</td>
<td>43</td>
<td>low resources</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pop-up ads</td>
<td>error messages</td>
<td>40</td>
<td>missing software</td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>login failure</td>
<td>installation issues</td>
<td>17</td>
<td>virus problems</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>browser, plug-in failure</td>
<td>unrecognized file type</td>
<td>5</td>
<td>file browser operations</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sum</td>
<td>125</td>
<td>287</td>
<td>79</td>
<td>34</td>
</tr>
</tbody>
</table>

Table 1.1.: Recompiled list of specific causes for frustration as reported by Ceaparu et al. [2004]; Lazar et al. [2006]

yielded remarkably similar results concerning the distribution of problem causes among categories, indicating that while certain causes for frustration might have changed, the overall distribution is still valid today.

Having a closer look at single items shows that a lot of issues are caused by what Baecker et al. [2000] calls a gap between what users know and what users need to know. Especially the items concerning software features, unexpected responses and behaviour of the program or Operating System (OS), and error messages fall into this category. Thus, by assisting users to reduce the gap between what they know and what they need to know by offering help using an appropriate help system, could have reduced if not completely prevented user frustration for at least 1/3 of cases of Table 1.1. This stresses the importance of user assisting technologies like help systems and the benefits that can be attained by improvements in this area.

1.1.2. How do Users Solve Software Problems?

While the causes for user frustration have already been investigated it is hard to find evaluations determining users’ actions when they are experiencing frustrating problems with computers. The studies from the last section [Ceaparu et al., 2004; Lazar et al., 2006] asked users, how they were ultimately able to solve specific problems by offering ten options (see Table 1.2) and participants were allowed to exclusively check one option. The results reported by Ceaparu et al. [2004] are listed in Table 1.2. This results reveal that “online help” and “consult a manual or book” were apparently the least commonly adopted solutions to problems. Ceaparu et al. [2004] argue accordingly that “providing
1.1. Motivation

<table>
<thead>
<tr>
<th>solution taken</th>
<th>#</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>I knew how to solve it because it happened before</td>
<td>137</td>
<td>26.3</td>
</tr>
<tr>
<td>I was unable to solve it</td>
<td>76</td>
<td>14.6</td>
</tr>
<tr>
<td>I figured out a way to fix it myself without help</td>
<td>62</td>
<td>11.9</td>
</tr>
<tr>
<td>I ignored the problem or found an alternative</td>
<td>62</td>
<td>11.9</td>
</tr>
<tr>
<td>I asked someone for help</td>
<td>56</td>
<td>10.7</td>
</tr>
<tr>
<td>I rebooted</td>
<td>51</td>
<td>9.8</td>
</tr>
<tr>
<td>I tried again</td>
<td>29</td>
<td>5.6</td>
</tr>
<tr>
<td>I restarted the program</td>
<td>29</td>
<td>5.6</td>
</tr>
<tr>
<td>I consulted online help or the system/application tutorial</td>
<td>15</td>
<td>2.9</td>
</tr>
<tr>
<td>I consulted a manual or a book</td>
<td>4</td>
<td>.8</td>
</tr>
<tr>
<td>sum</td>
<td>521</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2.: Combined list of solutions to frustrating experiences from Ceaparu et al. [2004]; Lazar et al. [2006]

Post-hoc assistance by way of electronic or paper manuals is not a sufficient solution to the problem of user frustration.” As a side note an interesting aspect of this statement is that the authors are equating online help and manuals. This shows that in 2004 the general comprehension was that online help systems are more or less only electronic versions of manuals. Sadly this is usually still true today. While we agree with this statement in general, the given study has some shortcomings with respect to evaluating the actual actions performed by users in case of computer problems. First of all it covers frustration with computers in general. This makes it hard to find relations between problems and solutions. For example the causes for frustration quite often seem to have been technical, like for example crashes or freezes. For this kind of problem rebooting and restarting the program are more or less the only solutions applicable and consulting a manual or online help are usually not an option. The second set of options (manual and online help) is more applicable to other, mostly application specific problems. However, since the focus of the studies was not on this aspect, relations between problem causes and solutions were not reported. Additionally, by far the most commonly adopted solution was “I knew how to solve it because it happened before” and consulting a manual or online help is also not an option if the user already knows the solution to a problem. Another aspect that might be important is that users were only allowed to tick the alternative that finally solved their problem. This means users might have consulted a manual or online help at first and after not finding a solution there have asked somebody or decided to ignore the problem. By far the most important issue when trying to apply the cited findings to today’s situation is the complete negligence of the Internet. This is absolutely fascinating and shows how quickly things can change in the HCI area. The studies were reported ten and eight years ago and there was no “I found a solution on the Internet” option among the list of solutions. This seems weird today, since using an Internet search engine to find solutions for computer related problems seems like a second nature to us now and only eight years ago this option was not even listed among the alternatives of a questionnaire.
1. **Introduction**

concerning computer related problems.

Hence, even though the results of Ceaparu et al. [2004]; Lazar et al. [2006] indicate that the help systems of software are hardly used when it comes to computer problems, we decided to perform a preliminary survey [Lang et al., 2013a] to further investigate users strategies to solve problems especially in an application specific context. Since the focus of this thesis is on help systems, we were especially interested in the current status of internal help systems of software. This was of interest to us, since if – in contrast to the findings of Ceaparu et al. [2004]; Lazar et al. [2006] – users perceive the internal help systems of software nowadays to be highly useful and report to use it a lot in case of problems, there would be no need for improvement. However, if users reported the opposite (hardly useful, hardly use it) this would proof that there actually is a strong need for improvement. Especially, since virtually every program comes with an internal help system and creating content for this help system is time consuming and software corporations are dedicating money and man hours for this purpose. If users do not access and benefit from the help content created, that money and effort would be spent in vain. Taking this into account we consider evaluating the usefulness of internal help systems to be a very important aspect and thus we were rather surprised by the lack of research dealing with this aspect. It seems like the software industry is acting on the assumption that having some kind of internal help is compulsory, but since nobody will use it anyway, the content is not created very carefully and does not have the focus when creating software. It definitely does not get the attention it might deserve and is treated rather neglectfully. However, this negligence and the ensuing cumulation of flaws might be the reason why users stop using the help system. That way leading into a vicious circle were internal help systems become more and more obsolete.

In our preliminary study [Lang et al., 2013a] we were able to show that younger and expert users currently prefer using the Internet to find solutions to software problems and older and less tech-savvy users tend to ask a third person for help. Especially these less tech-savvy, non expert users hardly use the internal help systems of software. Help systems in their current form seem to be rather helpful for expert users, a user group that is also able to find assistance using the Internet. Since from our perspective help systems should be especially designed to assist non-expert users, there is definitely a need for improvement in this area.

In addition to the results reported in [Lang et al., 2013a] we also asked participants how the perfect form of assistance for a software program would look like using an open question. We got some interesting answers and insights by this question, even though it was not formulated unambiguously. It seems like some participants thought that “perfect help for software programs” exclusively referred to the internal help system of software and gave answers like “clearly structured and logical”, while the question actually was supposed to target a wider scope and participants were supposed to give answers like “I want a printed manual with the software” or “There should be a fast responding hotline allowing me to talk to a real person”. The second example was actually the answer given by eight of the 45 participants. These participants stated that they wanted to talk to someone using a helpline. Given the finding that asking a third person for help is the typical strategy to solve software problems for so many participants, this is not a surprise.
1.2. Problem Setting

Ten participants stated that they wanted easy to follow step by step guides, tutorials, or examples including images or videos. Indicating that they wanted the help provided to be based on use cases [Adolph et al., 2002] and not just listing the functionalities of the program. Six participants explicitly mentioned that they wanted images. This shows that there might be a lack of demonstrativeness in tutorials without images and users might have trouble following tutorials not using images, but describing controls of the interface using textual descriptions. In this context four participants also answered that they wanted the help to be formulated clear and understandable for novice, non-expert users.

Five participants answered that they wanted some kind of integrated help system, with one participant stating for example, he wanted “something to click on, showing me what I am supposed to enter here”.

One participant even wrote that he would not want anything with “links” referring to “links” describing a typical “lost in hyperspace” [Theng et al., 1996] scenario.

The majority of answers shows that for internal help systems users want to be provided with a functionality showing them how to concretely solve given problems in an easy to follow way.

1.2. Problem Setting

Users are using computers to achieve well defined objectives. To do so they use programs that are, due to the nature of computers as general purpose machines, not perfectly fit for the task at hand and offer more features than users actually need for task completion. Thus, users are confronted with an overhead of complexity that make the software hard to learn and require an extended training period before the concrete objective can be achieved. Being confronted with this overly complex programs users need assistance to solve arising problems and learn how to efficiently use the software program. Not being able to solve problems and quickly use software to tackle the task at hand causes frustration in users.

1.3. Proposed Solution – Integrated Advanced Help System

The aspect of how to provide users with appropriate help to achieve certain goals has been quite neglected in favor of making software easier to use and to avoid usability problems in general. Approaches to efficiently teach users how to use complex software to quickly attain results and solve problems has not received the attention it deserves in research. Some of the problems arising in the context of computer usage can be solved by providing users with an improved help system. Our proposed solution is a modular help system that is integrated into the software and that assists users in using the software at multiple levels:

**Beginner** Beginners are assisted in identifying the basic relevant controls of the UI and are guided on their way to achieving first results quickly and without attaining in-depth knowledge of the concepts behind these controls and the program in general.
1. Introduction

**Intermediate** Users at an intermediate level are assisted in refining the basic results attained after initial guidance.

**Interested Users** Interested users are provided with additional information not directly needed for task completion, but allowing them to get an in-depth knowledge of the concepts behind the program.

1.4. Thesis Contributions

The contribution of this thesis can be divided into a theoretical, a practical, and an experimental aspect:

**Theoretical**

The theoretical foundation of this thesis is provided by the identification of a typical usage pattern of software. To efficiently use software users have to be introduced to important concepts and elements of a UI, allowing them to quickly attain results. This initial results are later refined using less essential elements of the UI following an exploratory learning based approach. The knowledge to efficiently guide users at this exploratory learning stage was attained processing the worked example from the beginning. Information describing additional concepts involved in software usage, not directly related to task completion, might be of interest and hence should be provided. However, essential and less important information should be well distinguishable. In this context various findings from the area of cognitive load theory have implications on the design of efficient help systems for software. These findings also allow to identify possible limitations and drawbacks of existing help systems. Accordingly, existing help systems are evaluated with respect to their impact on users’ cognitive load and their capabilities to provide assistance in the context of the identified usage pattern in the context of this thesis.

**Practical**

Based on our theoretical analysis of the software usage process we designed a help system for a grid computing web portal. In this context we made a number of practical contributions. The help system implemented is applicable for a wide range of problem domains. The implemented components are platform independent and highly reusable. During development a focus was on portability and seamless integration into the existing portal framework. The system consists of a modular framework that allows different parties to contribute content. This distributed content is seamlessly integrated into the overall system and a unified point of reference for assistance is offered to users.

The system is able to automatically generate worked examples in form of step by step tutorials that inherently reflect the current application state. Different to other help systems the system is capable of interacting with the content of the UI by pointing at elements and providing example input for input fields. Due to the tight coupling of help system and UI, developers and site administrators could save time creating help content compared to a help page based system.
Experimental

Various experiments conducted in the context of this thesis investigated aspects of performance, user acceptance and social behaviour towards computers. Overall six experiments have been performed and 203 participants were involved during these experiments. The results of our experiments have various implications on the design of help systems and showed the superiority of our system compared to a system following a help page based approach to user assistance concerning objective performance.

1.5. Document Structure

This thesis consists of four parts and twelve chapters. The first part is dedicated to providing an introduction and motivation for our work, to establish some theoretical background needed to evaluate current help systems for software and to evaluate existing help systems based on these theoretical findings. Accordingly, this first part comprises three chapters, the present one with the introduction and motivation, one chapter for the theoretical background, and a last chapter evaluating existing help systems. The second part of this thesis discusses some requirements we consider to be essential to an advanced help systems and afterwards introduces a prototype system implementing these requirements and that was realised in the context of this thesis. Hence, this part consists of two chapters. The third part gives an overview of the experiments performed to evaluate our help system and to identify factors contributing to the subjective acceptance of help systems. It starts with a chapter introducing the general methodology used throughout the experiments and afterwards all six experiments performed will be discussed. Thus, this part consists seven chapters overall. One for the general methodology and one for each experiment. Part four concludes this thesis and discusses future research directions.
2. Theoretical Background

“Tell me and I forget. Show me and I remember. Let me do and I understand” [attributed to Confucius (551–479 BC)]

This chapter discusses theoretical background and previous work in the context of help systems. The primary objective of help systems – as the name suggests – is not in teaching users, but to assist them to achieve their goals. Hence, findings in learning theory might not be directly applicable to help systems. Nevertheless, a lot of these findings, especially from the field of cognitive load theory, are very relevant for help systems, and a solid understanding of the current theory behind learning is indispensable when it comes to designing and evaluating help systems.

Using virtual agents in order to introduce a virtual peer, collaboratively assisting users in learning and problem solving, have a long standing tradition in the context of learning and help systems. Various results reported in literature suggest that using virtual agents might have beneficial effects on users’ subjective acceptance of help systems. Hence, some of this results and aspects that have to be considered when using anthropomorphic agents in computer systems will also be introduced within the scope of the following chapter.

2.1. Learning Theory

The initial quote from the beginning of this chapter illustrates that learning not only depends on the learner alone, but also on a number of external factors like for example the way learning material is presented to the learner. In order to know how to adapt the presentation of learning material to foster successful learning, the learning process has to be investigated. By understanding the cognition of learning, learning material can be adapted to better match learner’s cognitional capabilities. Thus, trying to maximize users’ performance, when it comes to assisting them at learning how to solve problems, involves being aware of limitations and characteristics of the human learning and problem solving apparatus. Some parts of this apparatus will be introduced within the next sections.

2.1.1. Memory Models

Atkinson and Shiffrin [1968] postulated a model of human memory consisting of three parts. These components are depicted in Figure 2.1.

External sensory stimuli enter the sensory register, sensory memory, or sensory buffer. This register consists of multiple parts, one part for each sense. The information carried
by the stimulus is not processed in this registers, but buffered to be retrieved by the short-term store, also called short-term memory. Information is only transferred from sensory registers to the short-term store when it receives attention. Otherwise this information decays rapidly and is forgotten.

In the short-term store the information received from the sensory registers is selected, organised in a meaningful way, and integrated into existing knowledge. Therefore, the short-term store retrieves information from the long-term store and transfers information back to it. The capacity of short term memory is limited and it is possible to keep seven plus minus two elements called chunks in short term memory simultaneously [Miller, 1956]. Similar to the sensory store, information in the short-term store decays and is forgotten within 20 to 30 seconds [Baddeley, 1986; Van Merriënboer and Sweller, 2005]. In the memory model of Atkinson and Shiffrin [1968] this decay can be prevented by actively rehearsing the information.

The model postulates that transfer from short- to long-term storage happens automatically as long as the information is attended to in short-term storage. Hence, the model suggests that rehearsal is the key process responsible for transferring information from short- to long-term storage and thus facilitating learning. This view of the learning process was later questioned for example by Craik and Lockhart [1972], suggesting that the nature and depth of processing is crucial to learning, with deeper, more elaborate processing leading to improved learning. Thus, while the model by Atkinson and Shiffrin [1968] postulates that the short term memory plays an integral part in learning and problem solving processes, it does not sufficiently describe the characteristics of this component, its interaction with other parts of the memory model [Baddeley and Hitch, 1974] and additionally was not consistent with neuropsychological evidence [Baddeley, 2012].

To improve on this situation Baddeley and Hitch [1974] introduced a new model of working memory, that was later refined by Baddeley [2000]. Figure 2.2 shows an overview of Baddeley’s model of working memory as described by Baddeley [2012].

The model consists of a central master system called the central executive controlling the flow of information to and from three domain-specific slave systems. This central executive is responsible to control and regulated different aspects of the system [Baddeley, 1996], like focusing and dividing attention to different targets or stimulus streams, task switching and interfacing with Long Term Memory (LTM) [Baddeley, 2012]. In this way the central executive resembles the unitary model of working memory introduced by Atkinson and Shiffrin [1968] [Sweller et al., 1998].

The initial Working Memory (WM) model from Baddeley and Hitch [1974] only defined
two slave systems, the *visuo-spatial sketch-pad* holding information about what we see and a *phonological loop* dealing with sound and phonological information. The idea that there must be two separate subsystem for visual and auditory information stemmed from the finding that there are tasks – like for example riding a bike and talking – that hardly interfere with each other if performed simultaneously, but for tasks from the same perceptual domain – like talking while reading a text – the performance is lower than for both tasks performed separately. The *episodic buffer* was added to the system later, when Baddeley and Wilson [2002] found that some amnesic patients, while not able to transfer information to long-term memory are nevertheless able to recall much more information of stories on a short-term basis than could be recalled relying exclusively on the phonological loop.

The more information available in WM at the same time, the more connections and relations can be identified and schemas can be constructed. Caused by this deeper processing, during information organisation and the construction of a mental model, understanding is supported [Baddeley, 1986]. According to Low and Sweller [2005] the final consequence of understanding the elements or information held in WM leads to changes in LTM, information is permanently stored and thus learned.

Hence, the presentation of instructional material should consider the limitations and characteristics inherent to WM, in order to not overwhelm a learner’s internal cognitive processes. Related to successful learning, this implies that instructional material has to be designed in a way best suited to the learner’s cognition and the characteristics of the WM accordingly. *Cognitive load theory* [Sweller, 1988] focuses on the adaptation of learning conditions to avoid cognitive overloading.

### 2.1.2. Cognitive Load Theory

The beginnings of cognitive load theory are based on the construct of a *mental load*, becoming imbalanced as soon as there is a situative difference between the demands of a task and personal faculties [Moray, 1979]. In case a task requires more capabilities than a learner is able to provide, there will be a mental overload. The point at which such
2. Theoretical Background

A mental overload occurs differs between people, is dependent on subjective perception and influenced by factors like fatigue, stress, training, and motivation [Kantowitz, 1987]. Does a learner for example have to use too much cognitive resources during problem solving tasks this can inhibit learning [Chandler and Sweller, 1991]. Instead of trying to change internal cognitive processes and the limited capacity of WM of learners, instructional material should be designed in a way to not additionally strain learners cognitive resources. In this context cognitive load theory establishes a general framework having a broad range of implications on instructional design [Sweller et al., 1998]. The theory introduces three different kinds of cognitive load:

**Intrinsic cognitive load** results from the level of complexity of a specific instructional topic. Every instructional topic inherently has an associated difficulty that is intrinsic to the material being dealt with. For an average person for example the intrinsic cognitive load in the context of simple arithmetic would usually be quite low and for a more complex topic involving probability density functions rather high.

**Extraneous cognitive load** describes the unnecessary cognitive load added to the inevitable intrinsic cognitive load due to the presentation of instructional material. Different to intrinsic cognitive load, extraneous cognitive load can be reduced by good instructional design.

**Germane cognitive load** is different to the other two types of cognitive load positive for learning. It is the result of devoting free WM resources to a deeper understanding of instructional material. Instructional design can specifically increase motivation and thus germane cognitive load.

Those three types of cognitive load are additive and together can not exceed the limited available WM resources [Moreno and Park, 2010]. Hence, if for example the extraneous cognitive load implied by poorly designed instructional material is too high, there might not be enough cognitive resources left especially for topics with high intrinsic or germane cognitive load and learning might be inhibited. In the typical computer software related context we assume germane cognitive load to be rather high, since users usually start using a software with a particular objective in mind. Thus users are willing to devote cognitive resources to learning how to use the software in order to achieve their given goals. For typical software with an extensive set of features, complexity and thus intrinsic cognitive load is also quite high. Hence, in order to foster the learning of how to use software, the extrinsic cognitive load for instructional material provided with the software has to be kept as low as possible to avoid cognitive load from exceeding users’ cognitive capacity. Different possibilities to reduce extraneous cognitive load will be introduced within the next sections.

2.1.3. Modality Principle

A way to decrease cognitive load or enhance WM capacity is described by the *modality principle*. Using multiple WM processors, WM capacity might be increased compared
2.1. Learning Theory

to relying exclusively on a single WM processor [Sweller et al., 1998]. To make use of both phonological loop and visuo-spatial sketch-pad in parallel presenting information in a mixed visual and auditory mode might be beneficial. Conforming to this there is evidence that information presented in a mixed mode – partly visual and partly auditory – is more effective in learning than information presented exclusively in one mode – visual or auditory alone [Low and Sweller, 2005].

Accordingly, it might be beneficial for help systems to use spoken output to provide information about the visual UI of computer programs.

2.1.4. Worked-Examples Effect

One approach to decrease extraneous cognitive load is using worked examples. In comparison to traditional problem solving approaches using worked examples has the advantage of drawing learners’ attention towards problem states and solution steps [Moreno and Park, 2010]. An experiment conducted by Cooper and Sweller [1987] could verify that using worked examples has positive effects on learning and can foster faster learning success. The reason for this effect can be attributed to the cognitive load induced on learners during problem solving. During problem solving the focus of attention is drawn towards aspects that are hardly related to the problem at hand. Compared to this worked examples allow to focus attention more adequately facilitating beneficial cognitive processing. WM capacity is less strained as during problem solving and learners are more involved in the learning process following worked examples. This circumstance can induce a reduction of extraneous and increase of germane cognitive load [Chandler and Sweller, 1991].

Worked examples are not applicable to every kind of learning material and only make sense in areas where step by step explanations can be used. Accordingly, research up to now focused on material related to mathematics [Tarmizi and Sweller, 1988] and physics [Ward and Sweller, 1990]. However, due to the nature of HCI that usually requires users to process a sequence of interaction steps to complete certain tasks, we consider the worked example approach to learning highly applicable for computer software and help systems. In the context of computer help systems worked examples consist of explanations and references to elements of the UI. Learners have to establish a mental link between these references and elements of the UI. This aspect of mental integration can – similar to the problem solving approach – have negative consequences on learning success. Instead of adequate guidance, worked examples might split a users’ attention, increase memorisation and mental reconstruction efforts [Runner et al., 2008] and this way unnecessarily strain users’ WM capacity [Chandler and Sweller, 1991]. Accordingly the positive effects of worked examples could not be universally replicated. In case worked examples required a split of attention between different information sources, the effect was mitigated and there was no longer a positive effect compared to the problem solving approach [Tarmizi and Sweller, 1988]. This shows that additional aspects play an important role in the context of worked example learning and cognitive load. To efficiently use worked examples in learning, extraneous processing and thus extraneous cognitive load have to be minimized.
2. Theoretical Background

2.1.5. Extraneous Processing in Multimedia Learning

Mayer [2005] proposes five multimedia design methods to achieve this minimization of extraneous processing, helping to avoid a potential extraneous overload:

**Coherence** By excluding extraneous material it is possible to improve learning, since it reduces the processing of extraneous material preserving WM capacity for the relevant material. This means for example that irrelevant but interesting statements and graphics should be excluded from instructional material, thus keeping the instructional material coherent and relevant.

**Signaling** Different to the coherence principle, which reduces extraneous processing by excluding irrelevant material, the signaling principle directs learners attention to essential material by providing cues for how to process a lesson. For typical help texts signaling can involve adding an outline and headings and using underlined and bold text.

**Redundancy** When redundant information is provided, like identical streams of spoken and on-screen text, extraneous processing occurs, as users have to reconcile the auditory and printed stream of words. Thus, it is preferable to avoid this kind of redundancy and provide either spoken or written text exclusively.

**Spatial contiguity** Placing printed words close to corresponding parts of an illustration or animation reduces the effort of having to scan back and forth between text and graphic. That way, reducing extraneous processing by reducing the amount of visual scanning required.

**Temporal contiguity** In order to minimize the need to hold representations in WM an animation and corresponding narration should be presented simultaneously rather than successively.

Those principles can be applied to help systems and have various implications on their design. They also help to identify limitations in existing user assistance systems and allow to evaluate possible problems.

2.1.6. Exploratory Learning and the Efficient Software Usage Process

Usually, when users start using a new software they have a certain goal in mind. Such an abstract objective could be something like “I need to write a letter to . . .”, “I have to file a tax declaration”, or “I want to create a spreadsheet summarizing my working hours”. Acting on this objective users plan a sequence of actions needed to be performed in order to achieve this goal. After planning the required actions, these actions are being executed, the produced result is interpreted and it is evaluated if actions were successful and the result matches the initial goal. Abstract tasks can be split into various hierarchical subtasks. For the letter writing task this means for example that there are subtasks like “I need to add a recipient to the letter” or “I want the subject header to have bold formatting”.

20
2.1. Learning Theory

The process of users using computers to achieve this kind of goals can be described in terms of the action cycle introduced by Norman [2002]. An adaptation of this cycle to computer usage is depicted in Figure 2.3.

Usually, the first step in the planning sequence of using a computer to solve given problems is to identify an appropriate program for the task at hand. Already this step is not trivial. There are for example people using spreadsheet programs like *Microsoft Excel* to write letters and people using What You See Is What You Get (WYSIWYG) word processors like *Microsoft Word* to write HTML markup for web pages. While this might not be the most appropriate choice for each of the tasks, it is nevertheless possible to use the program to accomplish the desired objective. This of course adds complexity to solving the task that could be avoided choosing a program that better fits the requirements imposed by the nature of the task. This shows, while programs are created with a special purpose in mind, due to the extended features they offer, they can be used for purposes that were not planned when designing the program. This makes the selection of
2. Theoretical Background

the appropriate program quite hard especially for users unfamiliar with the capabilities of a given software product.

After users have managed to select a program appropriate to solve their problem, they start using the software. At this stage the action sequence is completed for the first time and users have a chance to evaluate if their action had the desired result. A word processor displaying a blank page and an input caret is for example an indicator to users that they started the correct program to write text on a page.

Usually the started software – especially in case of software with a wide range of application – offers a lot more features than the user actually needs to accomplish his primary goal. In this context the pareto principle [Newman, 2005] or 80-20 rule often comes up. In its original form this law states that roughly 80% of effects are caused by only 20% of factors. Applied to software this would imply that 80% of users only use 20% of the features of a program to accomplish their objectives. While it is hard to find empirical evidence supporting this hypothesis, given the vast amount of features of, for example a general purpose word processor like Microsoft Word, it seems quite possible that this assumption holds. Undoubtedly, there are features of programs more relevant for task completion than other features and the existence of additional unneeded features reduces users' efficiency. To put this into perspective, a blog post by Harris [2005] from the Microsoft Developer Network states that the five top features of Microsoft Word 2003, namely paste, save, copy, undo, and bold account for about 32% of command use and paste alone accounts for more than 11%. Additionally, the article also states that while the curve flattens out considerably beyond the top ten of commands “people really do use a lot of the breadth of Office and beyond the top 10 commands there are a lot of different ways of using the product” [Harris, 2005]. This shows that while there is a clear hierarchy in the importance of features, none of the features are actually dispensable. Nevertheless, this abundance of features makes it hard for users to plan the next steps required to achieve their particular objective. Especially, since the UI of software usually does not provide any support to plan the next steps in the action sequence apart from the arrangement of UI elements and dialogues. As this arrangement tends to adhere to semantic principles it does not serve such a purpose at all. A typical example for the semantic ordering of elements are menu bars. Items are ordered into groups and subgroups according to their semantic meaning and thus do not allow users to infer an advantageous sequence of steps from the order of elements. This usually means users, especially those users using a software for the first time, have to familiarize with the UI using an exploratory approach. During this exploratory learning process the action sequence is continuously repeated and users permanently have to evaluate if their actions caused the desired outcome. Similar to the problem solving approach in learning, this permanent reevaluation of results might focus users’ attention on aspects not relevant to their initial problem and thus increase cognitive load and inhibit learning. Conforming to this, an experiment conducted by Tuovinen and Sweller [1999] showed that users not familiar with a certain domain – in case of the experiment the domain was databases – substantially profited from worked examples in comparison to exploration based learning. If users were familiar with the database domain of the experiment, using worked examples or an exploratory approach revealed no significant difference with respect to learning.
Tuovinen and Sweller [1999] attribute the lack of a difference in this case to users familiar with a domain being able to draw on existing knowledge guiding the exploration process.

Taking this into account we formulated the “efficient software usage process”. This process involves several steps:

1. A user starts using a software product with a predetermined objective in mind.

2. The user is assisted in attaining relevant results quickly by introducing him to essential concepts and features of the software providing him with a worked example. Following this approach maximizes learning and fosters “pareto efficient” software usage.

3. Initial results are refined employing less essential features of the software. The knowledge attained processing the worked example, guides the user through the exploratory learning process concerning these less prominent features.

This view on software usage has various consequences on the design and implementation of help systems. The first consequence is that the developers of the help system have to identify possible use cases reflecting users objectives, in order to provide users with an appropriate set of worked examples. The number of steps involved and features introduced during these worked examples should be as minimal and task relevant as possible to increase coherence and foster “pareto efficient” software usage. Additionally the help system should be capable of assisting users during exploratory learning of less relevant features.

2.2. Persona Effect

The influence of using animated agents in UI design has been investigated in a variety of different studies. Results reported by Lester et al. [1997] suggested the existence of a so called persona effect. This means that the presence of a lifelike character in an interactive learning environment had a strong positive effect on the perceived learning experience of 100 middle school students. This positive influence on subjective rating of interaction is supported by Van Mulk et al. [1998] for an animated agent used in a learning environment covering technical information. However, for a system providing non-technical information (introducing people) this positive effect could not be reproduced. Additionally, the presence of an animated agent had neither a positive nor a negative effect on objective measures like comprehension and recall for both kinds of information provided. Similar results were also reported by Moundridou and Virvou [2002]. While the presence of an animated agent enhanced student’s learning experience, it did not improve students’ short term learning.

A study by Baylor and Rosenberg-Kima [2006] suggests that introducing an animated agent can mitigate user frustration in case of problems. In this study users were answering an online survey and were interrupted by an error message popping up and blocking them from answering questions. In one condition there was an animated agent present during the survey and this agent delivered either an apologetic or empathic message for the error.
2. Theoretical Background

message at the end of the survey. In the second condition this message was provided using speech only without the presence of an animated agent. The presence of either message had a positive influence and students were attributing the reason for their frustration to the program instead of themselves. Especially in case of the empathic message the system with a visual representation of the agent was more effective than the speech only version, indicating that the presence of an anthropomorphic agent can help to reduce user frustration with software.

Different to the previous studies reporting positive consequences of using an anthropomorphic agent Rickenberg and Reeves [2000] report a rather detrimental effect of users feeling anxious when they were monitored by an anthropomorphic agent during their interaction with a web site. They also report decreased task performance for the system displaying a monitoring character. Nevertheless, they also report that using the character increased trust in the content of the web site.

This result of increased trust in web site content is similar to the results attained by a study performed by Chattaraman et al. [2012] evaluating the effects of using a virtual agent on a retail web site on elderly subjects with a mean age of 69 years.

2.3. Uncanny Valley

When designing help systems, especially if these employ animated agents or spoken output, the theory of the so called “uncanny valley” has to be considered. The idea of the existence of an uncanny valley was first expressed by Mori [1970] in the context of robot design. The original idea was that theoretically there should be a straight line characterising familiarity to humans between an industrial robot, that bears no resemblances to a human at all and a actual human or a perfect humanoid robot. The industrial robot would not exhibit any familiarity to a human being and thus would be at the point of origin of the familiarity scale. A healthy human by definition would be at the upper bound of the scale. Between a human being and the industrial robot would be a straight line of familiarity. This means for example a robot having two legs and two arms, and thus would be more human like than a plain industrial robot, would lie somewhere in-between industrial robot and human. Accordingly, the familiarity for this kind of robot would be perceived higher than for the industrial robot but not as high as for a human. However, this line does not appear to be straight. There seems to be a dip in it at a point where the robot is perceived to be quite human like but having a closer look at it reveals its unhuman nature. Mori [1970] uses the example of a prosthetic hand to illustrate this. Those hands can be manufactured to perfectly look like a human hand. However, as soon as the prosthetic hand is touched its texture reveals that it is actually not a real human hand causing a feeling of uncanniness. Thus, while the hand is very similar to a real human hand the perceived familiarity is rather low and even lower as for prosthesis not looking like a human hand at all. This relation between human likeness and familiarity is depicted in Figure 2.4.

Consider a help system using an anthropomorphic agent and representing the agent with a photograph of a real person. The user might expect to refer to a real person for
help in this case. Since there are quite a lot live chat system offering user support this seems not to be an unreasonable expectation. Realising that interaction is actually with a computer system might cause the help system to be perceived as uncanny.

An equivalent scenario would be if the voice of the system was too human like. Then, users might also believe that they are talking to a real human and realising that they are actually speaking with a computer system might cause the same uncanny feeling. Hence, it might be beneficial for spoken output to be clearly understandable but not too human like in the context of virtual agents.

Figure 2.4.: The uncanny valley according to Mori [1970]
3. Contemporary Help Systems and Resources for User Assistance

This chapter will present and discuss historical and recent efforts undertaken to improve user assistance in the context of HCI. Accordingly, the following sections will give a short overview of help systems integrated into software, that were especially designed to assist users with the bare WIMP interface of current operating systems. Of course this thesis does not cover a complete list of help systems for all software products available, but it tries to showcase some common strategies implemented to overcome the shortcomings of WIMP interfaces when it comes to users needing assistance.

In addition to these offline measures of user assistance the Internet is also a valuable resource for help. In case of certain programs it is even hard to draw a border between help available offline and online. In this context the concept of crowdsourcing user assistance has gained some importance recently. Hence, the last part of this section is dedicated to recent developments in this area.

At the present two types of interfaces are prevalent to computing. On the one hand there are touch based interfaces, like those provided by Android, iOS, and Windows Phone that are primarily used on mobile devices like smart phones and tablets. On the other hand there are more “traditional” WIMP interfaces as used in Windows 7, macOS, and various Linux distributions like Ubuntu openSUSE, or Debian and are usually used in conjunction with bigger desktop or laptop computers having larger screens and pointing devices like mice or touchpads. However, currently both approaches are converging and it is very likely that sooner or later there will be only one OS remaining for both kinds of devices. A first attempt to combine touch and mouse/keyboard based interaction is Windows 8 that can be used for both large desktop computers and small tablets. Ubuntu phone and macOS providing a user experience that becomes more and more similar to iOS are yet two other indicators for this trend. Thus, it is seems legitimate to evaluate the state of current help systems without making a distinction between both UI concepts.

3.1. Traditional Help Pages

The predominant way to provide users with help to use software is by means of dedicated help pages. Those usually consist of an indexed and hyperlinked [Bush, 1945] collection of text files that are arranged in chapters, sections, and subsections and provide some kind of search capability [Lang et al., 2013b; Nauman et al., 2010]. Help pages are usually accessible via the “Help menu” of programs, by an icon showing a question mark or by hitting the F1 key of the keyboard.
In this context we asked the participants of the survey in [Lang et al., 2013a] if they were aware of the function that is usually bound to the F1 key. Of the 45 participants 19 stated that they were aware of the function of the key and 18 subjects were actually able to name the correct function associated by mentioning something related to “Help”. Only one subject stated that the key would open the “start menu”. This is definitely not what we would have expected. We expected the number of users familiar with the concept behind the F1 key to be much lower. Especially, since internal help systems are hardly used by users. That is why we also did not ask participants how often they actually use the key to get help.

When it comes to help pages there are currently three different approaches implemented by software vendors. Those approaches differ in the extend, content available on the Internet is incorporated into the content available through the UI of the help system. There are traditional help pages that are available completely without an Internet connection, help systems that follow a hybrid approach adding content from the Internet to content available offline, and a third group of applications that only display content available on the Internet. Examples for each group as well as the drawbacks and advantages that come with each approach are introduced in the following sections. The last section covers drawbacks all approaches have in common.

### 3.1.1. Help Pages Available Offline

This kind of help pages is usually bundled with the software and available without an Internet connection. Some examples for this kind of help pages are shown in Figures 3.1–3.4.

Usually these help pages have similarities to a printed manual and can – at least in theory – be read from start to end. Compared to printed manuals these help pages have the advantage that they can be searched for keywords and that they allow to add hyperlinks between pages.

The UI integrating this help pages usually resembles the UI of web browsers. There are forward and back buttons, a home button and in some cases a function to manage bookmarks. Due to the nature of the content provided being actually just HTML pages sometimes even the web browser installed on the user’s computer is used to display the content. This is for example the case for the help pages of IBM® SPSS® Statistics Software (SPSS) (see Figure 3.2). Since users are familiar with the features and controls of their standard browsers, this seems to be an advantage over using a dedicated small browser integrated into the program. However, even in this case the UI of the help system as displayed in the browser often adds additional back and forward buttons instead of relying on the buttons built into the browser (see Figure 3.2). This way adding complexity to the UI of the help system and increasing extraneous cognitive load.

Another negative aspect of this kind of help system is that as it is shipped along with the software product making it hard to change and adapt the content provided. Additional content as well as changes correcting errors can only be distributed by means of an update. If for example the developers responsible for the help content realize that there is a certain feature of the software causing trouble to users, they do not have a
3.1. Traditional Help Pages

Figure 3.1.: UI of the help pages of version 3.8.1 of Eclipse.

Figure 3.2.: UI of the help pages of version 21 of SPSS.
3. Contemporary Help Systems and Resources for User Assistance

Figure 3.3.: UI of the help pages of version 2.8 of GIMP.

Figure 3.4.: UI of the help pages of version 4.2.4.2 of LibreOffice.
possibility to adapt the help content accordingly in order to assist users with the given problem. To avoid this situation many software vendors have started to split the content of their help pages into an online and an offline part.

3.1.2. Hybrid Offline/Online Help Pages

This approach is for example used in Microsoft Office for Windows 2010 (see Figure 3.5) or the help pages available for the Synology Disk Station Manager (see Figure 3.6).

Certain, rather basic content, of the help pages is kept locally and other parts are retrieved from the web. For both cases of the examples above the functionality of the help system is severely restricted in offline mode and most tutorials and guides are only available in online mode. Accordingly, a working Internet connection seems to be a prerequisite for using the system. In the case of the help system implemented for Microsoft Office 2010 the content is no longer organized in a book like manner. Users are presented with an arbitrarily sorted lists of keywords referring to subpages that link to guides and tutorials called “articles”. Those articles usually show users how to use certain features of the UI. In offline mode the search area at the top of the UI of the help system allows users to search locally stored help content.

The presentation of the help system in online mode significantly differs from the system in offline mode. The initial page offers less options and a second search field is added to the UI. This search field is interfacing to Microsoft Bing and allows users to search for help online. The major difference of using the search area of the help system compared to using Microsoft Bing directly from a web browser, seems to be a strong preference to return results referring to content hosted on support.microsoft.com.

The mere existence of two separate search fields suggests a different functionality for both search fields. A search using the upper field that is also available in offline mode might for example exclusively search for offline content and the second Bing related search field for online content. However, both search fields perform the same action and both search for off- and online content. The results retrieved according to user queries are displayed below the search field. Selecting one of the results opens the result in the help system window in case of content stored locally or opens a browser window displaying online content.

From our perspective the help system as implemented in Microsoft Office 2010 tremendously increases extraneous processing and cognitive load on users. The distinction between offline and online help adds unnecessary complexity to the system. Especially that the system offers two different search fields performing the same action, seems odd. The existence of two different search fields suggests a difference for the action performed using either field. That there actually is no difference is irrelevant and users have to make a decision which search field to use for their queries nevertheless. Accordingly the second search field was removed from the help systems of later versions of Microsoft Office featuring only one unified search field.

Another aspect of the system adding to its complexity is that content can either be displayed directly within the UI of the help system or in a browser window that is automatically opened, when a search result referring to external content is clicked. Thus,
3. Contemporary Help Systems and Resources for User Assistance

Figure 3.5: UI of the help pages integrated into Microsoft Office for Windows 2010 in offline mode (top) and online mode (bottom).
there are two different types of results having different effects and this difference might cause confusion. From a software vendor’s perspective the approach of splitting the help system into an offline and an online component has certain advantages. The content available offline can still provide users with basic help if no Internet connection is available. The online content of the system can easily be kept up to date and it is possible to adapt content according to user requests. Software vendors can create help material targeting specific problems identified by the evaluation of search queries. This can help to improve the usefulness of the help system tremendously. By addressing actual problems and the according exclusion of extraneous material, the coherence of the help system can be improved. If users for example do not have problems to use a certain feature of the software, there is no need to provide them with a tutorial on how to use this feature. Not offering assistance for aspects of the system that do not require assistance reduces the amount of extraneous material included into the help system.

Nevertheless, the split between off- and online content adds complexity to the help system and the limited capabilities of the systems in offline mode make the implementation of an offline mode appear to be quite dispensable. Especially, since users might be disappointed of the limited features and amount of content offered by the system, when they start the help system in offline mode for the first time. Thus, it might be advantageous to abandon the idea of offering some very restricted and basic offline assistance in favor of only providing help online. To a certain degree this aspect is reflected by the help system implemented in Microsoft Office 2013. This help system does not provide users with traditional help pages in offline mode at all. It still has some assistive functionality when offline, but this functionality is so restricted that it is discussed in another context.
3. Contemporary Help Systems and Resources for User Assistance

(see Section 3.5). Other software vendors even go one step further and exclusively offer help on the Internet. Examples for this kind of help systems will be given in the next section.

3.1.3. Help Pages Available Online

In this case there is no help available offline for the program at all and all content has to be retrieved from the Internet. This approach is typically chosen for web browser. Examples for browsers exclusively offering assistance online are Google Chrome\(^1\) (see Figure 3.7) and Mozilla\(^®\) Firefox\(^®\) (see Figure 3.8).

While this approach has the obvious disadvantage of requiring a working Internet connection, this restriction seems to be less severe in the context of web browsers, since using a browser more or less implies having a working Internet connection. For other software this approach is usually not chosen yet. This might change in the near future with working Internet connections being a prerequisite of using an increasing number of software products.

3.1.4. General Drawbacks

Regardless of whether the help content is stored somewhere on the Internet or locally the help page based approach to user assistance has some serious disadvantages from a user’s as well as from a developer’s perspective. Usually help pages contain a lot of information in order to cover all aspects and features of a UI. Accordingly, similar to the UI explained, help pages contain more and less important or relevant information, as help pages have to equally cover these more and less important aspects of the described software. This unavoidably seems to cause a lack of coherence for help pages. Extraneous material not relevant to solve a users current problems has to be included into the help pages and it is up to the user to find the help content applicable for his problems and needs. Due to this necessary inclusion of extraneous material typical “lost in hyperspace” Theng et al. [1996]; DeStefano and LeFevre [2007] situations may ensue. Take for example the help page of Gimp displayed in Figure 3.3. This text contains various links to material that might be extraneous concerning the problem that caused the user to consult the help pages or it might be highly relevant. The decision of reading the linked text first and proceeding afterwards is up to the user. This way adding extraneous cognitive load to the problem solving capacity of the user.

Another drawback of this approach from the users’ perspective is the lack of spatial contiguity. The content is presented to the user separate from the UI of the program. This means if a user found a solution to a problem he has to read the instructions and apply the information provided to the UI of the software. If the solution to the problem is provided by means of a tutorial or guide, users permanently have to switch between help content and UI. This includes reading and understanding textual descriptions and finding elements of the UI referred to in the tutorial based on images or text characterising the

\(^1\)http://support.google.com/chrome/ [accessed: 2. August 2014]
3.1. Traditional Help Pages

![Google Chrome help page](image1)

Figure 3.7.: Online help pages of *Google Chrome*\(^1\).

![Mozilla Firefox help page](image2)

Figure 3.8.: Online help pages of *Mozilla Firefox*\(^2\).
3. Contemporary Help Systems and Resources for User Assistance

element and its position in the UI. This adds extraneous cognitive load to the user and thus might inhibit learning.

The decoupling of help pages and UI of the program is also not beneficial from the developers’ perspective. Changes to the UI have to be carried over to the help pages inducing additional effort. This means for example if the icon or the name of an element in the UI is changed, this icon or name has to be replaced in the help content as well. Or if the position of an element in the UI is changed all tutorials in the help pages using this element have to be adapted to reflect these changes. Forgetting about adapting the help pages after making changes to the UI might cause the content provided to be of no use to the user anymore, that way even adding another problem to the existing one that was the reason for the user to consult the help system in the first place.

3.2. Tooltips

Screen real estate is limited. This usually causes the information on elements of the UI presented to users to be reduced to a bare minimum and elements and controls are usually displayed as icons or text labels only. A system to overcome this problem of limited screen space for buttons and other UI elements are Tooltips. Tooltips are small pieces of text that are displayed as soon as the mouse pointer rests over an interface element for a fixed – usually quite short – period of time.

One problem of tooltips that comes into play with modern touch based interfaces is the fact that there is no such thing as a mouse pointer. Thus they are not applicable for this kind of interface.

Another problem is that they are only displayed while the mouse pointer is resting still, or moving only within a rather small boundary after the tooltip is displayed. A small shake of the mouse is usually enough to make the tooltip disappear. In case a Tooltip contains lengthy text that takes more time reading an accidental move of the mouse causing the tooltip to disappear is quite annoying and a rather common event. This is why tooltips usually contain quite small bits of text, allowing users to quickly read the content. Most of the time they just give a name to a button that is displayed as an icon along with a short hint, like the keyboard shortcut activating the button or similar (see Figure 3.9).

Another restriction of tooltips is that the underlying framework usually confines them to displaying text only. This way reducing their possible expressiveness.

One reason why tooltips are very useful in the context of user assistance is their ability to give names to controls that are otherwise represented exclusively as icons. Having a precise semantic name for an abstract icon can be useful by itself and is also extremely important to find further help. It enables users to search for the given term on the Internet or the Help pages integrated into the given software and it also introduces users into the jargon used in the context of the program.

Thus, tooltips are still an integral part of user assistance of virtually all WIMP based modern operating system.
3.3. “What’s this?”

An attempt to overcome the limited expressiveness of tooltips is the so-called *What’s this* feature. This feature has been – and is still today – a part of applications and operating systems. Usually, this feature is integrated into the *Help* menu of programs and represented as a combined “pointer and question mark” icon along with the text “What’s this” (see the upper part of Figure 3.10). Activating the option turns the mouse pointer into the combined “pointer and question mark” icon from the menu (see the middle part of Figure 3.10) and clicking a control, for which a “What’s this” entry is available, shows an extended description of the selected control (see lower part of Figure 3.10).

Compared to tooltips that are restricted to short text messages, “What’s this” entries can usually contain images and formatted text.

To highlight the availability of this feature the *KDE project* integrated a question mark button into the window decoration right next to the “Minimize”, “Maximize”, and “Close” button (see Figure 3.11).

The behaviour of this question mark button is quite the same as for the example shown from Figure 3.10. After clicking it the mouse pointer is changed and shown as a combined question mark and pointer, when it is above an element with an available “What’s this” explanation and as pointer along with a “forbidden sign” otherwise (see Figure 3.12).

Even though this feature is presented quite prominently in *KDE* the developers seem to be quite aware of one of the major drawbacks of the “What’s this” approach. A KDE Techbase article for example states:

The problem with this approach is that the user can’t see whether a widget provides help or not. When the user activates the question mark button and doesn’t get any help window when clicking on a user interface element, he will get frustrated very quickly. [Gehrmann, 2008]
3. Contemporary Help Systems and Resources for User Assistance

Figure 3.10.: The “What’s this” feature in Microsoft Word 97.

Thus, this feature is not even widely used in KDE applications and finding an appropriate working example in openSUSE 13.1 to include it into this thesis was not easy and induced at least some of the frustration mentioned in the quote. Accordingly, while the “What’s this” approach could be a valuable source for further help concerning interface elements, the lack of visibility in this context is surely one of the major reasons why it hardly seems to be of any practical use.

3.4. Extended Tooltips

A straightforward approach to add information to tooltips is to extend its capabilities. Thus, in the context of Ribbon based interfaces [Dostál, 2010] extended tooltips have been introduced. Similar to “What’s this” entries these tooltips can contain formatted text and images (see Figure 3.13). Those tooltips are always displayed in an area below the controls of the interface and thus are never obscuring other interface elements. We consider extended tooltips are a beneficial extension to the traditional Tooltip paradigm and – if they contain useful information – can be a valuable source of assistance for users learning how to use the UI of a program. Due to their good spatial contiguity and

Figure 3.11.: The button for the “What’s this” feature in a KDE application.
3.4. Extended Tooltips

Figure 3.12.: The “What’s this” feature in KDE.

Figure 3.13.: Extended tooltip for the “Format Painter” UI control in Microsoft Word 2013 for Windows.
coherence they are especially useful to assist users during exploratory learning. Nevertheless, this feature also does not allow users to identify controls important for task completion and does not foster “pareto efficient” software usage.

3.5. Ubuntu HUD, macOS Sierra Help Menu Search, Office 2013 Offline Help

The so called HUD feature of Ubuntu from version 12.04 onwards to the current version (14.04) offers a solution to the problem of finding a menu item in the menu bar of a given program. By tapping the Alt key a special overlay window is opened and the user can start entering the name of the menu item. After the first letter is entered an incremental search is started and the list of matching items is displayed below the search field. The position of the item in the menu is displayed in parenthesis next to the item (see Figure 3.14). Users can select the desired item from the list using keyboard arrow keys or the mouse. The action triggered is the same as if the item was selected from the menu bar.

A similar feature is integrated into the help system of macOS from version 10.5 (Mac OS X Leopard) onwards. This system also allows users to search for menu items. A special search field is integrated into the “Help” menu of applications. As soon as the user starts entering letters into this field an incremental search is started and matching menu items are listed below the search field. Selecting one of the items by hovering it with the mouse or using the arrow keys, activates the sub menu containing this item and the position of the selected item is indicated with an arrow (see Figure 3.15). When the item in the list is activated, either by clicking or hitting the Enter key, the action associated to the menu item is performed. In addition to searching the menu bar of the active program the Help Menu Search also searches the help pages related to the active program. Results of this search are shown in another list below the results for menu
3.5. Ubuntu HUD, macOS Sierra Help Menu Search, Office 2013 Offline Help

Figure 3.15.: Interface of the Help Menu Search feature of macOS Sierra.

Compared to the HUD feature of Ubuntu this approach has the advantage of actually showing users the position of the item in the underlying menu structure. It also offers an easily accessible and highly visible interface to search for relevant help pages.

Another help system that falls into this category is the offline help integrated into Microsoft Office 2013 for Windows. While the help system in Microsoft Office 2010 for Windows still covers some basic content (see Section 3.1), the help system of Office 2013 – if activated in offline mode – only features an input field that allows to search for controls of the Ribbon UI (see Figure 3.16 on the left) by name.

This search functionality is not incremental. Thus, after entering the name of an UI element users have to hit Enter or click the “magnifying glass” icon next to the text area to activate the search process. Then matching results are displayed below the search field (see Figure 3.16 in the middle). Clicking one of the results opens another page, explaining where to find the element, showing an icon of the element and presenting an unformatted, floating text version of the Extended Tooltip (Section 3.4) associated with this element (see Figure 3.16 on the right).

While these features enable users to find menu items, they are of no use if the user does not remember or know the name of the menu item. Thus, all three features are not suitable to introduce users to the features of a given software or teaching them the jargon used in the context of unfamiliar software.

Due to the limited signaling capabilities with respect to helping users to distinguish less from more important elements of the UI and the accordingly bad coherence, the features are also not optimal from an extraneous processing perspective. Concerning spatial contiguity and signaling with respect to helping users find the element searched
for, the Help Menu Search in macOS is superior to the other two systems. It actually points out the element searched for in the UI of the program while the other two systems only give a textual explanation, where to find the element.

3.6. Tips and Tours at Installation, Startup, or First Start

Some software products show tips on how to use the software at startup or start a tour, when the software is started for the first time, after an update that caused changes to the UI or feature set of the software, or even during installation. An early example for those tips could be found in Windows 95. The dialogue presented to users when Windows 95 was started is shown in Figure 3.17. It featured a “Did you know…” area giving users tips on how to use the software, offered a button to start an introductory tour and an easy way to access the help system through the “What’s New” button. This approach of introducing users to new software is still used today in a wide area of applications. An example for a introductory tour of a software presented during installation is for instance Ubuntu 13.04. During the installation process various screens are presented to users giving a rough overview of the most important features and software included with the OS. A selection of those screens is shown in Figure 3.18. The approach of showing some introductory material during software installation seems to be a good idea. Information is presented to users during idle time, when there is nothing else to do and users have to wait anyway. They can cover aspects of the software important to a wide range of users by addressing typical use cases. For the example from Figure 3.18, these use cases are word processing, web browsing and listening to music (not shown in Figure 3.18) or how to install new software and change the appearance of the OS.
3.6. Tips and Tours at Installation, Startup, or First Start

Figure 3.17.: Help dialogue presented to users after booting Windows 95.

Figure 3.18.: Selection of introductory screens presented to users during the Ubuntu 13.04 installation process.
the last screen points out further possibilities users can refer to, if they are in need for assistance. In this context it is also interesting to note that the last screen does not mention the internal help of software as a point of reference for user assistance. This shows that Canonical's strategy is to crowdsource user assistance and the focus is not on providing help along with the software product. By pointing out typical use cases and differences of the software included compared to the software most users switching from other OS might be used to, this introductory screens have a signaling function and help users to identify applications relevant to achieve their objectives. However, the material presented during installation lacks an introduction to the basic principles of the UI used for Ubuntu. As the Unity desktop metaphor implemented in current versions of Ubuntu differs widely from the UI of other OS novice users might not be familiar with the concepts introduced in the context of Unity. Why such an introduction is not part of the tour presented during the installation process might be due to developers realising the major drawback inherent to this approach of user assistance, its limited spatial and temporal contiguity. The information provided is not directly useful and applicable at the moment it is presented. Thus, introducing the key concepts of the UI at this stage might not make sense since it will take an extensive amount of time, till the installation process is finished and users can actually use the software and try out what they have learned during such an introductory tutorial. Hence chances are high that users have already forgotten the information provided in the mean time, especially since due to the lack of spatial contiguity this information would have to be provided in a rather abstract way. Nevertheless, we consider using the time needed to install software to provide information assisting users with using a software product a good idea, since it is a very unobtrusive way to provide this kind of information making use of time that would otherwise be wasted.

Concerning temporal contiguity it seems preferable to provide assistive information, when a software product is started for the first time, as this allows users to directly apply the learned content by trying it in the context of the new software. A more recent example for this approach is for instance the introductory tour of Microsoft PowerPoint 2013 as shown in Figure 3.19. The functionality of this kind of introductory tour or tips at startup varies widely. Some tours are merely listing the feature set of a software – as is the case for Microsoft PowerPoint 2013 – some are pointing out essential features of a software offering tips like the initial Windows 95 example, some are pointing out new features and allow users to enable these features, and others are introducing users to concepts of the UI of the software (see Figure 3.20 for examples).

Depending on the functionality of these initial messages or tours the usefulness also differs widely. The most important aspect this approach to user assistance can have is its signaling function, pointing out important elements and key concepts of a UI and introducing users to the given software. However, the coherence of this kind of assistance is often rather limited and the information covered by the tips or the initial tour are often not very relevant for achieving a users' objective. Take for example a user that wants to create a presentation using PowerPoint. The information provided by the tour does not help the user with achieving this goal at all. The tour rather introduces extraneous material and is thus increasing users' extraneous cognitive load. The same is true for
3.6. Tips and Tours at Installation, Startup, or First Start

Figure 3.19.: First screens of the tour presented to users of Microsoft PowerPoint 2013.

Figure 3.20.: Typical notification messages as displayed after updates of Apps on mobile phones and tablet computers introducing new features or changes to the UI.
tips shown at software startup like the tips of Windows 95. While the information provided with this tips might be useful in general, they are usually not covering aspects fundamental for task completion, introduce extraneous, nice to know material and are thus adding to users cognitive load. The impact on users' cognitive load is especially high if this kind of help is offered after an update and a user is not prepared for the interruption of his workflow introduced by a tip or tour. Consider for example the messages displayed after starting the Apps from Figure 3.20. In both cases the user had used the software before and is usually aware of its features. Thus, the applications are usually started with a very concrete objective, like "I want to go running now" or "I need to share a document with my coworkers using OneDrive" in mind. In both situations the users' workflow is interrupted by the messages and he has to draw decisions and make adaptations that are currently not relevant for his objective.

Thus, while the approach of user assistance by showing tips or tours at startup is usually well intended and can serve an important signaling purpose by pointing out important elements and concepts of a UI, it often fails to do so due to limitations in coherence. Especially for feature rich software the information provided by this kind of help is usually not very relevant for task completion and thus rather inhibiting efficient usage of software by adding extraneous material.

### 3.7. Clippy the Paperclip

One of the boldest and most popular (or unpopular) attempts to establish a new kind of user assistance was the Office Assistant introduced with Microsoft Office for Windows 97 in 1996. This Office Assistant is usually referred to using its official name "Clippit" or the misnomer "Clippy" that was used for the default representation of the assistant in the English version of Microsoft Office for Windows resembling a paper clip (see Figure 3.21).

A basic feature of Clippy was that it offered an interface to the help system of Microsoft Office products and allowed users to enter search queries. Matches to those search queries were presented in Clippy's speech bubble and selecting one of the options opened the corresponding help pages (see Figure 3.22).

In addition to this the assistant incorporated technology developed in the context of the Microsoft Lumiere project [Horvitz et al., 1998]. It employed Bayesian User Models [Jameson, 1996] to infer users' needs and give appropriate advice based on this needs. This advice could consist of short tips offered to users, when the system assumed that users might require assistance. In these cases an image of a light bulb was displayed next to the representation of Clippy. Clicking the light bulb activated Clippy's speech bubble and the tip was displayed (see Figure 3.23).

Another form of advice that could be offered by Clippy was suggesting users to use wizards (see Section 3.9) that were available for various, specific use cases. Most famous among these wizards is surely the wizard assisting users with writing letters. Clippy's attempts to force this feature on users every time they seemed to be writing a letter, surely has contributed to this wizard's fame. The typical process in Microsoft Office
3.7. Clippy the Paperclip

Figure 3.21.: Representation of Clippit the Microsoft Office Assistant in Microsoft Office for Windows 97 and Microsoft Office for Windows 2003.

Figure 3.22.: Clippy answering user queries.

Figure 3.23.: Clippy offering advice to users.
3. Contemporary Help Systems and Resources for User Assistance

Figure 3.24.: Clippy suggesting to use the letter writing wizard, some dialogues of the wizard and the resulting letter.

For Windows 97 of starting to write a letter, Clippy suggesting to use the wizard, some steps of the wizard and the resulting letter are shown in Figure 3.24. In retrospect and from our perspective the wizard as well as Clippy suggesting its usage are both rather nice features of the help system. The feature set of the wizard was convenient, it was relatively easy to use, the input already entered was parsed correctly and transferred to the corresponding text fields of the wizard, and the design and layout of the resulting letter was quite appealing.

Nevertheless, Clippy was widely reviewed negatively by users [Xiao et al., 2004] and this is also conforming to findings of our preliminary study mentioned earlier [Lang et al., 2013a]. While only two out of 45 participants could not remember Clippy when a picture of him was shown, more than 70 percent of subjects characterized him as annoying. The level of annoyance Clippy caused even granted him a spot in popular culture. He had for example appearances in TV shows like The Simpsons and Family Guy, easter eggs in Microsoft software products, or a subreddit of reddit\(^3\) dedicated to him. As soon as Google started using Pegman to introduce users to features of Google Maps it did not

3.8. Virtual Agents on E-Commerce and Retail Web Sites

take long until it was mocked as Clippy imitation. When the team responsible for the
development of Microsoft Excel answered questions on reddit they explicitly mentioned
that they were also willing to answer questions related to Clippy\(^4\). While it is hard to
tell how much of this dubious fame can actually be attributed to Clippy being perceived
to be annoying, and how much is just caused by the very vocal hype surrounding him is
unclear. Nevertheless, the impact Clippy had on users and the development of software
help systems is undeniable and might have contributed to the stagnation in this area of
HCI.

Taking this into account it is surprising that the reasons for the negative reception of
Clippy have not been investigated more thoroughly. A point that is usually mentioned
in this context is Clippy’s feature to proactively offer assistance to users and thus
interrupting and intruding users’ workflow [Kim and Baylor, 2006; Dix et al., 2010]. Consider
for example the suggestion of using the letter writing wizard from Figure 3.24. This sug-
gestion is made after important, rather administrative parts – the return and recipient’s
addresses – have already been completed and the user had started writing the actual
content of the letter, by entering the salutation. At this stage the usual workflow would
be to write the letter’s text content first and take care of the formatting afterwards.
Hence, a user’s mind is occupied with what he wants to write and the message he wants
to convey with the letter. At this point Clippy offers assistance and interrupts the users’
workflow. As the assistance offered rather refers to the formatting of the letter, this adds
extraneous cognitive load and requires extraneous processing on behalf of the user.

Another aspect of Clippy that might have been perceived to be annoying is the ex-
tensive use of animations. Clippy was animated and drew users' attention towards him
when he had a tip or when he was idle for a certain amount of time. This idle animations
might have caused a feeling of being monitored by the agent and also might have con-
tributed to a feeling of anxiety in users that can be caused in the context virtual agents
(see Section 2.2). The animations also might have drawn attention towards the agent
and increased users’ cognitive load accordingly.

3.8. Virtual Agents on E-Commerce and Retail Web Sites

Despite of the negative reception of Clippy, various companies are currently trying to
exploit the persona effect by adding virtual agents to their web sites. There are different
vendors providing companies with the technology needed to embed virtual agents into
their web pages. There are for example systems available from Nuance\(^5\) whose virtual
agent Nina is used as Jess on the homepage of JetStar\(^6\) and without a visual represen-
tation of the agent on the Coca Cola\(^7\) web page (see Figure 3.25), Artificial Solutions\(^8\),

\(^{4}\text{http://www.reddit.com/r/IAmA/comments/227tme/we_are_the_microsoft_excel_team_ask_us_}
\text{anything/}[accessed 15. August 2014]


\(^{8}\text{http://www.artificial-solutions.com/natural-language-interaction-products/}

49
3. Contemporary Help Systems and Resources for User Assistance

Figure 3.25: Nina Virtual Assistant as displayed on the Nuance\(^5\) (left), Jetstar\(^6\) (middle), and Coca Cola\(^7\) (right) web sites.

the company responsible for IKEA Anna, a virtual agent available at the web pages of IKEA\(^9\) furniture stores (see Figure 3.26).

Those systems usually offer a search support function presenting information on products and services according to criteria specified by the user [Chattaraman et al., 2012]. Some agents also offer a basic decision support function allowing users to compare different alternatives [Sproule and Archer, 2000]. Interaction with the UI of the underlying web page is usually restricted to opening links and thus changing the content of the page displayed behind the agent. Systems are typically not able to directly interact with the content of the page, like pointing at specific elements. This decoupling of the virtual agent from the underlying UI can lead to unpleasant results. Like for example in case of IKEA Anna after using the input field to tell Anna to “Show me some bookcases please” and selecting the “Bookcase” category from the “Product types” list offered afterwards, a page showing bookcases is opened behind the window containing the UI of Anna. There Anna tells the user to click the “Read more” option, below the price to get further information on certain products. However, this option is not available in the UI of the displayed web page (see Figure 3.27). It seems like the UI was changed and it was forgotten to update the corresponding help text provided by Anna. To avoid this kind of errors there should be a closer coupling between UI of the web page and the virtual agent. How such a close coupling was achieved for the prototype system implemented in the scope of this thesis will be explained in Section 5.3.2

Examining the representation of agents from the example above, shows that virtual agents used productively on retail web sites seem to be employing very human like virtual agents. At least in case of Jetstar’s Jess (see Figure 3.25) the uncanny valley might have been reached, especially considering Jessi’s “creepy” smile. While this aspect might be highly subjective, the question arises nevertheless if a less human like, more comical representation of agents would not be a better choice to avoid the uncanny valley in this


50
3.9. Wizards

Another way to help users solving tasks is implementing so called wizards. Wizards consist of a sequence of dialogues leading users through a well defined series of steps. If wizards are a part of the UI of software or should be considered as part of the help system depends on the functionality offered by the wizard. Sometimes wizards are the only means implemented to access certain features of software. Typical setup wizards/assistants guiding users through the installation process of software are an example for this kind of wizard. Settings like the installation directory can only be specified using the wizard and can not be changed using the UI of the program afterwards. Thus, for these settings we consider the wizard to be part of the UI and not part of the help system. However, in addition to this, wizards also request information required to set up the program installed. Examples for this kind of information are the location of a user’s “Music” folder during the installation of an audio player managing music collections, or requesting a password during the installation of an OS. This kind of settings usually can be changed afterwards using the “regular” UI of the software. Since this functionality of wizards adds a layer of abstraction to the underlying UI and assists users with completing tasks without having to familiarize with the underlying UI first, we consider this kind of wizards to be part of the help system. As wizards enable users to achieve goals quickly by only asking for

Figure 3.26: IKEA Anna as presented on the web page of IKEA⁹.
3. Contemporary Help Systems and Resources for User Assistance

Figure 3.27: IKEA Anna telling users to click the non-existent “Read more” option to get further information on specific items.9
3.9. Wizards

An example for a wizard abstracting from the UI of the underlying program is Clippy’s letter writing wizard from Section 3.7. A more recent example for this kind of user assistance is the network connection assistant of macOS Sierra. This wizard guides users through setting up network connections using a sequence of dialogues (see Figure 3.28).

Despite of the benefits of implementing wizards to abstract from the complexity of the underlying UI of computer programs, this approach has one major drawback. This drawback is, that it is completely decoupled from the underlying UI. While after completing the wizard or assistant users have attained results quickly, the wizard does not provide information to users on how to refine these results using the UI of the program. For the example of the letter writing wizard from Microsoft Word 97 this means, users have a nicely formatted letter, but they did not learn anything about formatting letters or documents in general. If a user now wants to write a document for which an appropriate wizard is not available he has to learn how to do this from scratch, since using the wizard did not provide him with any information on how to use the UI of Microsoft Word to write and format generic documents. Thus, while wizards are a nice feature to enable users to get results quickly, by decoupling the wizard from the UI they are not appropriate to teach users how to use a software and hence do not foster learning.

Figure 3.28.: Screen shots of the Network Connection Assistant of macOS Sierra.
3. Contemporary Help Systems and Resources for User Assistance

3.10. Crowdsourced Online Assistance

When a user is confronted with a certain problem related to a software product, it is likely that other users also experienced this problem before and found a solution to it, or a more experienced user is able to assist novice users in solving problems. Providing users with an easy way to state their problems on the Internet and allowing them to share their experience is an interesting way to provide users with the help they need. This way the “ask a friend or coworker” strategy, that is often used to solve software problems by non-expert users [Lang et al., 2013a], is applied to the huge community of the Internet. However, this approach also has some drawbacks. Users have to be familiar with the correct terminology to be able to correctly formulate search queries applicable to find solutions for their problems. This is especially hard for novice users not used to the lingo of programs [Chilana et al., 2012]. Due to the huge amount of information on the Internet finding an appropriate solution can be quite time consuming. As the help content available on the Internet is decoupled from the actual software, the information concerning the problem at hand might refer to a different version of the software and the controls needed to solve the problem might have been moved to different locations, been renamed, have a different icon or even have been removed from the UI of the software completely. While it was possible to include the software version into the search query for software distributed in a traditional “buy a data medium containing the software and install” approach, this is hard to impossible for web-based applications and Apps considering the rapid update cycles. For this kind of software products the version and UI can change without the user even noticing. It was for example easy to compose a query specifically searching help for Microsoft Word 2010 by searching for “text bold Word 2010”, but targeting a specific version of the web-based Word 365 is virtually impossible. At first this seems not to be a problem, since all users always use the current version of web-based software and thus only need help for this version. However, there might be help available for a previous version of Word 365 and since it is not possible to explicitly restrict search queries to a version, this outdated help might come up and the solution provided might be no longer applicable.

Even if the user is able to find an appropriate solution there is still the problem of spatial contiguity. The help content is presented separately from the UI of the software and if the the found solution consists of a tutorial containing several steps users permanently have to switch between the tutorial and the UI.

Concerning extraneous cognitive processing characteristics of using the Internet to solve software problems this approach seems to be the worst of all. The Internet is full of extraneous material and there is virtually no signaling available. Of course material found can both be coherent and have excellent signaling but as a whole the Internet based approach lacks both desirable characteristics and increases extraneous processing and cognitive load on users. With respect to spatial contiguity the problems of this approach are the same as for the user support paradigm based on help pages. The help content is displayed in the browser window and separated from the UI of the software.
3.11. StackExchange

The idea behind *StackExchange*\(^{10}\) is to allow people to ask questions that can be answered by other people. One user could for example ask “What is the opposite of an Epiphany?” Everybody could provide an answer for this question and the answers given can be rated in an upvote, downvote manner. That way an adequate or the best answer to this question can be found collaboratively. Providing answers others find helpful earns the person providing this answers reputation and additional privileges introducing elements of gamification [Deterding, 2012].

While the idea of *StackExchange* is quite general this approach has proven to be very useful in the context of problems with computers and software [Vasilescu et al., 2014]. There are for example dedicated sites – called communities in the context of *StackExchange* – like *Ask Ubuntu*\(^{11}\) and *Android Enthusiasts*\(^{12}\) covering certain aspects of using computers, *superuser*\(^{13}\) dedicated to questions in the context of computing in general and *stackoverflow*\(^{14}\) for programmers. At those sites users can state their computer related questions and get answers from the community. Figure 3.29 shows an example for questions related to *Microsoft Word* in the *superuser* community of *StackExchange*\(^ {13}\).

For sites that have reached enough popularity, questions are usually answered quite quickly. The *stackoverflow* site was even coined the “fastest Q&A site in the west” by Mamykina et al. [2011], with 92% of questions being answered in a median time of only 11 minutes [Vasilescu et al., 2014].

It is rather obvious that – especially in the communities concerned with computing – *StackExchange* is not a site novice users visit regularly to learn how to use software. It is rather a site consulted in case a user is confronted with an actual problem he can not solve on his own. Then searching for a similar question can be very helpful to find solutions to the problem at hand. If there is no solution to the problem yet, users can still post their problem and hope to get an answer from an expert in the community.

This approach is especially interesting since it allows real users to ask questions using terms they consider to be adequate to describe their problem. If for example a user not familiar to the lingo used in context of this questions, posts a question there, chances are higher that another novice user utilises the same or similar terms, when searching for a solution to the same problem. If an expert user now sees this question, he can answer it directly or refer the novice user to an answer of his question that was stated using the correct terminology. That way not only answering the user’s question but also introducing him to the correct terminology.

This might become more clear giving an example. The author of this thesis was using *Microsoft Excel* to create a histogram. Since he did not use *Microsoft Excel* recently, he was not used to the new *Ribbon* style interface of recent version of *Microsoft Office* products. However, he has created some histograms with a current version of *Excel*.

---


3. Contemporary Help Systems and Resources for User Assistance

Figure 3.29.: Questions concerning *Microsoft Word* in the *superuser* community of *StackExchange*\(^\text{13}\).
before and hence was not a complete novice. Now he wanted to change the style of the x-axis labels in said histogram. He knew that usually there is a sidebar allowing to do exactly this. However at that instant this sidebar was not visible. The author now used an Internet search engine and entered “open sidebar excel 2013”. The results returned for this query were not helpful at all to solve the given problem. Somehow, the author managed to open the sidebar using a trial and error approach afterwards clicking wildly in the Excel UI until he realized that a double click on an x-axis label reopens the sidebar. If the author would have stated a question like “How to open sidebar in excel 2013?” and posted a problem description stating his problem to the superuser community of Stack Exchange, an expert reading the question could easily have answered it or referred the author to an answer to a question not using the general term “sidebar”, but the more appropriate term “Task Pane”. If another user experienced the same problem now and used the same terminology to describe it, he would be able to find the correct solution to his problem even though his problem statement would not be completely correct.

However to gain a critical mass of novice users posting questions to this kind of question and answer pages novice users have to be aware of the existence of this sites and know about the possibility of asking questions there. However, as the results of our preliminary study [Lang et al., 2013a] suggest especially novice users tend to avoid using the internet in case of problems. Thus this kind of users might be under-represented at this kind of help forum too and thus not be able to benefit from this advantage. Additionally, while users might find solutions to their problems on sites like Stack Exchange by using internet search engines, they have to be aware of the existence of this sites and the possibility to post their own questions in order to contribute.

To check if Stack Exchange is known to a wide audience of users we asked the participants of our initial study [Lang et al., 2013a] if they were aware that this site existed and out of 45 participants only three stated that they would know of it. Participants were also asked to state their opinion of the site in free text. One participant answering that he was aware of the site wrote “negative because it causes program crashes” and the second one stated “I guess this is something related to stacks in computing”, clearly indicating that they were actually meaning something else. Only one participant wrote that this would be an interesting site, but the information he finds there would be partially inaccurate. Thus the pages from the Stack Exchange network are clearly not known to a wide range of users in the non-expert computer user community and thus this group surely does not exploit the undeniable advantages of this kind of community based assistance.

Yet another aspect of Stack Exchange might contribute to novice users being deterred from using it. There is not only a voting system for answers, there is also a voting system for questions. The up- and downvote buttons for questions are labelled “This question shows research effort; it is useful and clear” and “This question does not show any research effort; it is unclear or not useful” respectively. While a naively formulated question might be clear and useful to other novice users, an expert user might consider the question to lack research effort, by not using the correct terminology. That way questions from novice users might not receive the attention they deserve. In general Stack Exchange communities seem to focus on expert users and the descriptions of the sites usually state that rather clearly. The description of the superuser community for example reads “Super
User is a question and answer site for computer enthusiasts and power users" and the description of Stack Overflow “Stack Overflow is a question and answer site for professional and enthusiast programmers”. That way already narrowing down the intended audience of the pages and kind of explicitly excluding novice, non-expert users. This quite surely deters novice users from posting questions there. However, there are also sites like Ask Ubuntu – “Ask Ubuntu is a question and answer site for Ubuntu users and developers” -- that do not require question askers to have an expert status. Of course it is questionable how many Ubuntu users are actually naive, non-experts, but it shows that StackExchange sites with a wider, more general intended audience exist. If this kind of page would gain a critical mass of users it would surely be able to serve as a powerful resource for user assistance. However, as especially non-expert users tend to not use the Internet to find solutions to software problems [Lang et al., 2013a] it seems like this approach is of limited use in this context.

Thus, it may be stated that despite of potential benefits, sites like StackExchange are still more useful to expert users than they are for novice users and while given a certain popularity, these sites could help to overcome the disadvantage of having to know the correct terms to start appropriate search queries in internet search engines. However, right now those sites are of limited use to non-expert users.

3.12. Conclusion

The capabilities and features of help systems have changed little over time. Most of the features implemented in modern help systems integrated into software have for example already been available in the help system of Microsoft Office for Windows 97. Those typical features are:

- A set of help pages explaining components of the UI and providing tutorials and guides to use these features.
- Tooltips or a “What’s this” feature providing assistive information on elements of the UI.
- A tour introducing the software at the first start or pointing out changes after an update.
- Tips offered to the user when the software is started.
- Wizards abstracting from the underlying UI and guiding users to achieve certain objectives by processing a sequence of dialogues.

Compared to the functionality implemented in the help system of Microsoft Office for Windows 97 some of these features have been slightly improved, like the feature to find menu items based on their name in Ubuntu and macOS, or extended tooltips in the Ribbon based interfaces used in recent versions of Microsoft software. However, the basic functionality remained the same. The major progress made in this area seems to have been to avoid aspects of the help system of Microsoft Office for Windows 97 that
might have caused Clippy to be perceived as annoying. This features were proactivity and using an animated agent as an interface to the help system.

None of the help systems introduced in this chapter is capable of efficiently assisting users in identifying elements of the UI that are important to achieve certain objectives and to differentiate between highly relevant and unimportant elements. Using worked examples to achieve this seems to be an appropriate way to foster this kind of introduction of users to a UI. During a worked example important elements of the UI could be pointed out and users could efficiently be guided to quickly achieve results. While help pages are theoretically capable of providing users with the information needed for a worked example based introduction to the UI employing help pages has various disadvantages. By summarizing this kind of information in help pages the content provided quickly becomes quite extensive leading to bad coherence and causing users to have problems finding the relevant information. Additionally, the spatial contiguity of information provided on help pages separated from the UI is not optimal. It requires users to integrate information from two different sources, increases extraneous cognitive load and might inhibit learning accordingly.

Crowdsourcing user assistance and offering help online is an approach becoming more and more relevant. However, especially non-expert users tend not to consult online resources for assistance in case of software problems. In addition to this using the Internet to find answers concerning issues with software introduces additional problems like information found not being applicable for the software version used. They also require users to be familiar with the terms used in the context of specific software products in order to find relevant information.
Part II.

Advanced Help System
4. Requirements of an Advanced Help System

Based on the theoretical considerations and on the evaluation of existing help systems from the previous chapters, we inferred requirements for an advanced help system that we consider to be beneficial to users as well as developers of computer software. These requirements as well as the ensuing advantages from a user's and from a developer's perspective will be introduced within the following sections. The final part of this chapter will formally describe software programmes in general and how a help system can be integrated into such a software programme.

4.1. From a User’s Perspective

Existing help systems do not take the proficiency level of users into account. Usually the means of assistance implemented by software are split among different features, like hints at startup, or tooltips. Nevertheless, the major resource for assistance usually consists of an extensive set of documents providing information on how to use the software at hand. For users at different proficiency levels however, different ways of assistance should be offered. For novice users employing worked examples seems to be the best approach to introduce them to the UI of software products. Accordingly, an advanced help system should be capable of providing users with tutorials on how to achieve certain goals using a worked example based approach. Apart from pointing out essential elements of a UI during the worked examples and thus fostering “pareto efficient” software usage offering worked examples to users at this stage has additional advantages. An appropriate set of examples suggested to users by the help system can serve as an indicator introducing users to the capabilities of a software product and hence assist users in choosing the correct software for a given task. Another, and maybe the most important advantage of worked examples in this context is that – if the set of worked examples provided sufficiently reflects possible use cases of the software – they allow users to quickly attain first results using the software.

In order to reduce extraneous cognitive load on users and to avoid split attention effects, elements of the UI involved during the worked example should be pointed out to the user directly within the context of the UI and not in an a separate window. Straightforwardly integrating additional information on elements into the UI also increases spatial contiguity and further reduces extraneous processing required to complete the tutorial realizing the worked example. From our perspective this approach to reducing extraneous processing will have a beneficial effect on learning as well as on users’ efficiency working with a
software product and none of the existing help systems introduced in Chapter 3 is capable of providing this kind of assistance.

After getting acquainted with the essential aspects of a UI following the initial worked examples, users need to refine the results attained using these worked examples to better reflect their needs. These intermediate users already have a limited understanding of how to use the software and might require further information on certain aspects of the UI that were not introduced during the worked examples. At this stage an exploratory approach to learning about these additional features seems to be most beneficial. Hence, a help system should be capable of assisting users during this exploratory learning process providing information on elements of the UI and their intended usage. Existing features like “What’s this” (see Section 3.3) and extended tooltips (see Section 3.4) are means to provide this kind of assistance that have good spatial and temporal contiguity. However, these systems introduce a separate resource for help into the software that is not integrated into the general help system. Having a look at the offline help feature of Microsoft Office 2013 (see Section 3.5) this discrepancy becomes quite clear. The help system offers two quite distinct ways to get information on certain elements of the UI. The first way is by pointing at an element and waiting till the tooltip is displayed and the second way is to search for an element by name using the help system. The information acquired with both approaches is essentially the same. Of course, the integrated help system should be capable of providing this kind of information on UI elements, but adding this additional information to the content of the help system reduces coherence by adding potentially extraneous material, adds redundancy, and increases extraneous processing accordingly. A way to improve on this situation is to better integrate this kind of functionality into the general help system by exclusively offering a simple way to access this feature from within the UI of the help system.

Finally, after attaining satisfactory results using the software, expert users might want to get some kind of in depth knowledge of the software they use. This kind of knowledge might for example consist of “nice to know” information rather extraneous to users’ original objectives. An advanced help system should be capable of providing this kind of information. However, due to the extraneous nature of the material covered extensive care has to be taken to distinguish this kind of information from the more essential, less extraneous information available through the help system and advanced signaling strategies have to be implemented in order to achieve this behaviour and reduce extraneous cognitive load on users.

Another drawback of help page based systems is their lack of context dependence. The information provided by the system is completely independent of the current state of the software in use. If a user has for example activated the “Paintbrush Tool” in a bitmap graphics program and opens the help system it is reasonable to assume that the user might need assistance with this particular tool and offer him help concerning this topic. However, help systems are currently lacking this kind of capability and help is usually offered in a context independent way. While it is surely hard for certain types of software – like word processors – to establish a reasonable model of the current state of the system in order to provide context dependent help, a certain degree of context sensitivity can easily be achieved for other kinds of software. Hence, we consider offering help according
4.2. From a Software Vendor's Perspective

Currently, help systems are not very well integrated into software products. By using the typical document based approach the content presented within the help system has to be written separately. Even simple changes to the UI of software require updates of the content provided by the help system. These changes require additional effort from developers and the manual adaptation of content is rather error prone. Forgetting to update the help content after applying changes to the UI can cause the help content provided to become worthless for users and might even increase users' frustration when experiencing problems and not finding applicable solutions referring to the help system. However, this source for errors can easily be eliminated by tighter coupling the help system and the UI of software. If links between descriptions and elements of the UI are made explicit using unique pointers rather than providing implicit semantical or graphical references to elements within floating text, the validity of this links can be verified programmatically, problems can be detected automatically, and to a certain degree the consistency of information provided by the help system and the current status of the UI can be ensured. Accordingly an advanced help system should implement a possibility to explicitly refer to elements of the UI and this way reduce possible sources for errors and reduce additional efforts required, when applying changes to the UI.

The unrestricted nature of document based help content makes the explicit definition of style guides and best practices for writing help content necessary. There are no implicit affordances or constraints and authors have to learn how to write appropriate content in order to ensure uniform appearance and usefulness of content. There are two distinctive ways for software vendors to create the content for such a help system. There can
4. Requirements of an Advanced Help System

be a special team dedicated to authoring the content or developers involved with the implementation of the UI are supposed to write this content. Of course there can also be hybrid forms, where UI developers can for example provide the content for tooltips and specialized employees write content for the dedicated help pages. Either way writing the help content is a collaborative process and a lack of implicit constraints can lead to a high level of inconsistency in the content provided by the system. Additionally, splitting up content into various sources like tooltips and help pages makes reusing content difficult. To solve this issue an advanced help system should be capable of automatically integrating content from various sources and contributors into a single resource of assistance and present this content to the user in a uniform way. This way relieving authors – at least to a certain degree – from learning about the correct way to write appropriate content, having to manually integrate their content into the overall help pages, and verifying the consistency of information between different sources.

4.3. Concept

A tight integration of help content and functional application code is most practicable at the source code level. This way providing help content when integrating new functionality into software becomes an integral part of the development lifecycle. Accordingly, a framework fostering this inclusion of help content at the source code level has to be an essential component of software products. Apart from artefacts like the typical \texttt{element.setTooltip(String text)} that is part of virtually every WIMP UI framework, existing software implementations lack the capability of defining assistive, context dependent information on UI elements. However, the extensive usage of tooltips in existing software – usually every icon in a toolbar comes with a tooltip – confirms the hypothesis that a programmatic approach of amending a UI with additional usage information from within the program code can lead to a high level of acceptance and usage on the developers’ side. A framework fostering advanced user assistance capabilities has to take this aspect into account and should provide software vendors with a well defined way of adding user assistance features into their software products. Similar to the already mentioned tooltips that can easily be attached to UI elements an adequate framework has to enable software developers in collaboration with technical writers to implement a broad range of assistance. This means that a implementation of a help system framework should be based on enriching the actual source code of programs with usage information. To increase the benefits of this kind of information to users the framework should foster links between those distinct information units in order to integrate them into larger guides or tutorials. This means that authors have to be able to define separate bits of usage information for certain UI elements and embed this information into a larger context. How such a help system framework can formally be integrated into software programmes will be described in the following sections.
4.3.1. Formal Definition of Help Systems

Software Programmes

Analogous to automata the a software programme can be represented formally by a 4-tuple \((Q, \Sigma, \Delta, q_0)\) consisting of:

- \(Q\) a set of states
- \(\Sigma\) a set of input symbols (e.g. UI elements)
- \(\Delta\) a transition relation \(\Delta \subseteq (Q \times \Sigma) \times Q\)
- \(q_0 \in Q\) an initial state

**Behaviour** The software programme starts at the initial state \(q_0 \in Q\) and a user interacts with the software using a control \(\sigma \in \Sigma\). This interaction puts the software into a subsequent state \(q_1 \in Q\) as defined by \(\Delta\). If \(\Delta = \{Q \times \Sigma \times Q\}\) the software does not impose any constraints on the possible user input at any state. Since this is normally not the case for typical software, usually \(\Delta \subseteq \{Q \times \Sigma \times Q\}\) holds. In this context the difference \(\{Q \times \Sigma \times Q\} \setminus \Delta\) can be considered as implementing constraints in the sense of Norman [2002].

The set of all possible results \(R \subseteq Q\) attainable with a given software programme is defined as \(r \in R \iff \exists \delta_0 = q_0 \circ \sigma_0 \rightarrow q_1, \delta_1 = q_1 \circ \sigma_1 \rightarrow q_2, \ldots, \delta_n = q_n \circ \sigma_n \rightarrow r \in \Delta \wedge q_0 \circ \sigma_0 \rightarrow q_1 \circ \sigma_1 \rightarrow q_2 \ldots \rightarrow q_n \circ \sigma_n \rightarrow r\). This means a result or state \(r \in Q\) can be achieved using the software, if there is a finite sequence of steps \(\omega = \sigma_0, \ldots, \sigma_n\) transferring the state of the software programme from initial state \(q_0\) to state \(r\).

Applying an interaction sequence \(\omega \in \Sigma^*\) to a software in state \(q \in Q\) that causes the software to reach state \(r \in Q\) is denoted as \((q_0, \omega) \rightarrow r\).

A task \(t\) is defined to be solvable by the software if \(\exists r \in Q : \Upsilon(r, t)\), where \(\Upsilon(r, t)\) denotes that the user considers the software in state \(r\) an acceptable solution to problem \(-\) or task \(-\) \(t\). Accordingly, the set \(T = \{t \mid \exists r \in Q \wedge \Upsilon(r, t)\}\) is the set of all tasks solvable using the software. As there are typically multiple acceptable solutions for a problem \(t\), there is a set \(A(t) = \{r \mid \Upsilon(r, t)\}\). This set of solutions \(A(t)\) is ordered with \(r_1 > r_2 : r_1, r_2 \in A(t)\), if the user prefers solution \(r_1\) over solution \(r_2\).

Given a task \(t\) a user now has to determine a sequence of steps \(\sigma_0, \ldots, \sigma_n\) that transfers the state of the software from the initial state \(q_0\) to an accepting state \(r \in A(t)\). For a given task \(t\), a perfectly usable software would satisfy \(\exists! \sigma_0, \ldots, \sigma_n \wedge (q_0, \sigma_0) \ldots (q_n, \sigma_n) r \wedge |\Delta| = n \wedge \Upsilon(r, t) \wedge A(t) = \{r\}\). Thus, at every state only one user action is possible and following a predetermined deterministic sequence of interaction steps provides the user with a solution to the problem at hand. A real world example for such a software would be a slide show presentation software that displays a button for each slide but only allows users to click the button for the next slide and to close the application when the last slide is displayed. As the example shows, while the software offers a solution to the problem “sequentially display a slide show”, restricting the functionality of a software programme to such an extreme degree is usually not reasonable and reduces the applicability of a
4. Requirements of an Advanced Help System

programme to a broader range of problems. Hence, designing usable software always involves a tradeoff between constraining users too much by unnecessarily reducing the cardinality of $\Delta$ and consequently $Q$ and leaving users with too many options by not constraining interaction sufficiently.

Using the formalism introduced above different components of a computer programme with respect to a given problem or task can be identified. These are the:

- imperative component
- enhancement component
- dispensable component
- and the inhibiting component

The following sections will provide a formal definition of each of these components.

**Imp erative Component of a Software Programme**

Given a set of solutions $A(t)$ to a problem $t$ the imperative component of a programme $P = (Q, \Sigma, \Delta, q_0)$ is defined by the triplet $I(P, A(t)) = (Q_I, \Sigma_I, \Delta_I)$ fulfilling the following conditions:

- $Q_I \subseteq Q$
- $\Sigma_I \subseteq \Sigma$
- $\Delta_I \subseteq \Delta$

- $\exists \delta_0 = q_0 \circ \sigma_0 \rightarrow q_1, \delta_1 = q_1 \circ \sigma_1 \rightarrow q_2, \ldots, \delta_n = q_n \circ \sigma_n \rightarrow r \land$
  - $q_0 \circ \sigma_0 \rightarrow q_1 \circ \sigma_1 \rightarrow q_2 \circ \ldots \rightarrow q_n \circ \sigma_n \rightarrow r \land r \in A(t) \land$
  - $\{q_0, q_1, \ldots, q_n\} \cap Q_I = Q_I \land$
  - $\{\sigma_0, \sigma_1, \ldots, \sigma_n\} \cap \Sigma_I = \Sigma_I \land$
  - $\{\delta_0, \delta_1, \ldots, \delta_n\} \cap \Delta_I = \Delta_I$

This defines the imperative component as those elements of the software that have to be used to solve a given problem.

**Redundant Component of a Software Programme**

The redundant component of a software programme $P = (Q, \Sigma, \Delta, q_0)$ is defined by the triplet $R(P, A(t)) = (Q_R, \Sigma_R, \Delta_R)$ where:

- $Q_R \not\subseteq Q$
- $\Sigma_R \not\subseteq \Sigma$
- $\Delta_R \not\subseteq \Delta$
4.3. Concept

\[
\begin{align*}
\exists \delta R_0 &= q R_0 \circ \sigma R_0 \rightarrow q R_1, \delta R_1 = q R_1 \circ \sigma R_1 \rightarrow q R_2, \ldots, \delta R_n = q R_n \circ \sigma R_n \rightarrow r R \land \\
q R_0 \circ \sigma R_0 &= q R_1 \circ \sigma R_1 \rightarrow q R_2 \circ \ldots \rightarrow q R_n \circ \sigma R_n \rightarrow r R \land r R \in A(t) \land \\
\{q R_0, q R_1, \ldots, q R_n\} &\cap Q R \neq \emptyset \land \\
\{\sigma R_0, \sigma R_1, \ldots, \sigma R_n\} &\cap \Sigma R \neq \emptyset \land \\
\{\delta R_0, \delta R_1, \ldots, \delta R_n\} &\cap \Delta R \neq \emptyset \land \\
\exists \delta R_0 &= q R_0 \circ \sigma R_0 \rightarrow q R_1, \delta R_1 = q R_1 \circ \sigma R_1 \rightarrow q R_2, \ldots, \delta R_n = q R_n \circ \sigma R_n \rightarrow r R^c \land \\
q R_0 \circ \sigma R_0 &= q R_1 \circ \sigma R_1 \rightarrow q R_2 \circ \ldots \rightarrow q R_n \circ \sigma R_n \rightarrow r R^c \land r R^c \in A(t) \land \\
\{q R_0, q R_1, \ldots, q R_n\} &\cap Q R = \emptyset \land \\
\{\sigma R_0, \sigma R_1, \ldots, \sigma R_n\} &\cap \Sigma R = \emptyset \land \\
\{\delta R_0, \delta R_1, \ldots, \delta R_n\} &\cap \Delta R = \emptyset \land \\
r R^c &= r R
\end{align*}
\]

This means, every solution \(A(t)\) that can be achieved using a redundant component can also be achieved using another component of the system. This also implies that \(Q_I \cap Q_R = \emptyset\).

Enhancement Component of a Software Programme

Given a set of solutions \(A(t)\) to a problem \(t\) the enhancement component of a programme \(P = (Q, \Sigma, \Delta, q_0)\) is defined by the triplet \(E(P, A(t)) = (Q_E, \Sigma_E, \Delta_E)\) fulfilling the following conditions:

- \(Q_E \subsetneq Q\)
- \(\Sigma_E \subsetneq \Sigma\)
- \(\Delta_E \subsetneq \Delta\)

\[
\begin{align*}
\exists \delta E_0 &= q E_0 \circ \sigma E_0 \rightarrow q E_1, \delta E_1 = q E_1 \circ \sigma E_1 \rightarrow q E_2, \ldots, \delta E_n = q E_n \circ \sigma E_n \rightarrow r E \land \\
q E_0 \circ \sigma E_0 &= q E_1 \circ \sigma E_1 \rightarrow q E_2 \circ \ldots \rightarrow q E_n \circ \sigma E_n \rightarrow r E \land r E \in A(t) \land \\
\{q E_0, q E_1, \ldots, q E_n\} &\cap Q E \neq \emptyset \land \\
\{\sigma E_0, \sigma E_1, \ldots, \sigma E_n\} &\cap \Sigma E \neq \emptyset \land \\
\{\delta E_0, \delta E_1, \ldots, \delta E_n\} &\cap \Delta E \neq \emptyset \land \\
\exists \delta E_0 &= q E_0 \circ \sigma E_0 \rightarrow q E_1, \delta E_1 = q E_1 \circ \sigma E_1 \rightarrow q E_2, \ldots, \delta E_n = q E_n \circ \sigma E_n \rightarrow r E^c \land \\
q E_0 \circ \sigma E_0 &= q E_1 \circ \sigma E_1 \rightarrow q E_2 \circ \ldots \rightarrow q E_n \circ \sigma E_n \rightarrow r E^c \land r E^c \in A(t) \land \\
\{q E_0, q E_1, \ldots, q E_n\} &\cap Q E = \emptyset \land \\
\{\sigma E_0, \sigma E_1, \ldots, \sigma E_n\} &\cap \Sigma E = \emptyset \land \\
\{\delta E_0, \delta E_1, \ldots, \delta E_n\} &\cap \Delta E = \emptyset \land \\
r E^c &< r E
\end{align*}
\]

Hence, the enhancement part of the system is defined as the part that is not imperatively required to solve the given problem or task but using elements of this class yields preferable results.
4. Requirements of an Advanced Help System

Dispensable Component of a Software Programme

Similar to the enhancement component, the dispensable component of a programme \( P = (Q, \Sigma, \Delta, q_0) \) for given set of solutions \( A(t) \) to a problem \( t \) is defined by the triplet \( D(P, A(t)) = (Q_D, \Sigma_D, \Delta_D) \) that fulfills the following conditions:

- \( Q_D \subset Q \)
- \( \Sigma_D \subset \Sigma \)
- \( \Delta_D \subset \Delta \)

\[ \exists \delta_{D_0} = q_{D_0} \circ \sigma_{D_0} \rightarrow q_{D_1}, \delta_{D_1} = q_{D_1} \circ \sigma_{D_1} \rightarrow q_{D_2}, \ldots, \delta_{D_n} = q_{D_n} \circ \sigma_{D_n} \rightarrow r_D \land \\
\{q_{D_0}, q_{D_1}, \ldots, q_{D_n}\} \cap Q_D \neq \emptyset \land \\
\{\sigma_{D_0}, \sigma_{D_1}, \ldots, \sigma_{D_n}\} \cap \Sigma_D \neq \emptyset \land \\
\{\delta_{D_0}, \delta_{D_1}, \ldots, \delta_{D_n}\} \cap \Delta_D \neq \emptyset \land \\
\exists \delta_{D_0^c} = q_{D_0^c} \circ \sigma_{D_0^c} \rightarrow q_{D_1^c}, \delta_{D_1^c} = q_{D_1^c} \circ \sigma_{D_1^c} \rightarrow q_{D_2^c}, \ldots, \delta_{D_n^c} = q_{D_n^c} \circ \sigma_{D_n^c} \rightarrow r_{D^c} \land \\
\{q_{D_0^c}, q_{D_1^c}, \ldots, q_{D_n^c}\} \cap Q_D = \emptyset \land \\
\{\sigma_{D_0^c}, \sigma_{D_1^c}, \ldots, \sigma_{D_n^c}\} \cap \Sigma_D = \emptyset \land \\
\{\delta_{D_0^c}, \delta_{D_1^c}, \ldots, \delta_{D_n^c}\} \cap \Delta_D = \emptyset \land \\
r_{D^c} > r_D \]

This means the dispensable component of a software system with respect to a given task is defined as those components that can be used but reduce the quality of results.

Inhibiting Component of a Software Programme

The inhibiting component can be considered as the compliment to the imperative component of the software programme. In the context of a programme \( P = (Q, \Sigma, \Delta, q_0) \) and a given set of solutions \( A(t) \) to a task \( t \) it is defined by the triplet \( X(P, A(t)) = (Q_X, \Sigma_X, \Delta_X) \) where:

- \( Q_X \subset Q \)
- \( \Sigma_X \subset \Sigma \)
- \( \Delta_X \subset \Delta \)

\[ \exists \delta_{X_0} = q_{X_0} \circ \sigma_{X_0} \rightarrow q_{X_1}, \delta_{X_1} = q_{X_1} \circ \sigma_{X_1} \rightarrow q_{X_2}, \ldots, \delta_{X_n} = q_{X_n} \circ \sigma_{X_n} \rightarrow r_X \land \\
\{q_{X_0}, q_{X_1}, \ldots, q_{X_n}\} \cap Q_X \neq \emptyset \land \\
\{\sigma_{X_0}, \sigma_{X_1}, \ldots, \sigma_{X_n}\} \cap \Sigma_X \neq \emptyset \land \\
\{\delta_{X_0}, \delta_{X_1}, \ldots, \delta_{X_n}\} \cap \Delta_X \neq \emptyset \land \\
r_X > r_D \]

Accordingly, none of the elements being part of the inhibitive component may be used to solve a given task.
4.3.2. Help Systems

When implementing a software programme developers have set of tasks $G \subseteq T$ in mind that should be solvable using the software. For such a software programme $P = (Q, \Sigma, \Delta, q_0)$ the help system part of that software can be formally defined as a triplet $H(P, G)(Q_H, \Sigma_H, \Delta_H)$ that satisfies all of the following conditions:

- $Q_H \subseteq Q$
- $\Sigma_H \subseteq \Sigma$
- $\Delta_H \subseteq \Delta$

- $\forall t \in G \land \forall \tau : Y(r, t) \exists \delta_0 = q_0 \circ \sigma_0 \rightarrow q_1, \delta_1 = q_1 \circ \sigma_1 \rightarrow q_2, \ldots, \delta_n = q_n \circ \sigma_n \rightarrow r \land \{q_0, q_1, \ldots, q_n\} \cap Q_H = \emptyset \land \{\sigma_0, \sigma_1, \ldots, \sigma_n\} \cap \Sigma_H = \emptyset \land \{\delta_0, \delta_1, \ldots, \delta_n\} \cap \Delta_H = \emptyset$

- $\exists \delta = q_1 \circ \sigma \rightarrow q_2 \in \Delta_H \Rightarrow \beta \delta = q_1 \circ \sigma \rightarrow q_2 \in \Delta \forall q_1, q_2 \in Q \land \forall \sigma \in \Sigma$

Thus, the help system is defined as that part of a software that is not necessarily involved in solving the tasks or problems the software was designed to solve. Or, in other words, for every task that is completed employing the help system part of a system, there is another sequence of steps yielding an equivalent result that is not relying on the functionality of the help system. The last rule is necessary, since without it, two different controls offering the same functionality – like a menu item and corresponding toolbar button – would cause all but one of the controls to be considered as part of the help system.

Applying the component definitions from above this definition formally classifies the help system either as a part of the redundant (like e.g. the wizards from Section 3.9 or the virtual agents from Section 3.8) or dispensable (like e.g. the help pages described in Section 3.1) component of a software programme. However, this classification does not sufficiently reflect the impact an efficient help system can have on software usage. In order to correctly classify help systems a new less utilitarian component of software programmes has to be defined. This component can be called the enabling component. The benefits of this component stem from the fact that it is able to establish a semantic relation between $\sigma \in \Sigma$ and $q_1 \circ \sigma \rightarrow q_2$ and can assist users in establishing a mental link between $\omega \in \Sigma^*$ and $G$. Existing help systems establish a link between an action $\sigma$ and its effect $q_2$ on $q_1$ for example by means of using tooltips to give a short textual description of the effect caused by $\sigma$. On the other hand, help pages are used to provide users with tutorials, or sequences of steps that have to be performed in order to solve a certain task and thus assist users in establishing a mental link between tasks and action sequences. Accordingly, the enabling component or help system of a computer programme $P = (Q, \Sigma, \Delta, q_0)$ for a set of tasks $G \subseteq T$ could be defined as 6-tupel $H(P, G) = (Q_H, \Sigma_H, \Delta_H, H, L, \Phi)$ where:

71
4. Requirements of an Advanced Help System

- $H$ a set of descriptions with semantic meaning
- $L$ a help relation $H \subseteq Q \times \Delta$
- $\Phi$ a sequential help relation $\Phi \subseteq H \times H$

**Behaviour**

The relation $H$ specifies semantic descriptions for operations that are linked to UI elements defined in $\Delta$ by means of the help relation $L$. These descriptions are conveyed to the user depending on the current state $q$ of the software and depending on the action $\sigma$ chosen by the user. In this context is $\sigma \in \Sigma_H$ for systems with purely reactive help and $\sigma \in \Sigma$ for systems that proactively offer help.

The relation $\Phi$ establishes a connection between different controls of the system depending on the current system state $q$. This way fostering tutorials for certain tasks $t \in G$ with $\Upsilon(r,t)$, by establishing links between different $\delta \in \Delta$ that contribute to a sequence of steps $\omega \in \Sigma^*$ with $(q,\omega) \rightarrow r$.

**Current Situation**

Especially the sequential help relation $\Phi$ is not sufficiently implemented in current help systems. There is usually no way to specify relations between different parts of the help system apart from hypertext links. Additionally, the relation between UI elements and help content is realized on a semantic and not on a syntactic level in this context. This means that the relation between the written text and the UI element the text refers to is exclusively established by visual or textual semantics and not by programmatic means.

The following chapter will introduce a prototype system that solves this issue and allows the programmatic definition of $H$, $L$, and $\Phi$ at the source code level. This way it introduces an advanced framework for user assistance in software programmes.
5. The Prototype System

This chapter will describe how we implemented the requirements for an advanced help system defined in the previous chapter in the context of a web-based interface to the Grid computing facilities of the bwGRiD\(^1\). The initial situation we faced when we started designing the system along with the framework conditions will be explained in the following sections. Afterwards the features implementing the requirements for an advanced help system – as defined in Chapter 4 – in the context of our prototype system will be described afterwards introducing the features of the help system from a users’ perspective. The final part of this chapter is introducing the technical realization of the system implementing the requirements defined for the an advanced help system from a software vendor’s perspective.

5.1. Project Context

The realization of the described help system was funded by the Ministry of Science, Research and the Arts Baden-Württemberg in the context of the bwGRiD project. It was a joint effort of different facilities – universities, universities of applied science, and research institutions – throughout Baden-Württemberg to provide scientists with grid computing resources on different sites [Dynowski et al., 2012]. The sites involved were:

- Stuttgart
- Karlsruhe
- Ulm/Konstanz
- Tübingen
- Heidelberg
- Mannheim
- Freiburg
- Esslingen

The computing resources installed at every site were jointly shared by users at each location and members of each partner facility were able to access the systems of all partners. The idea behind this approach was to enable scientists to run computationally

\(^1\)www.bw-grid.de
5. The Prototype System

Figure 5.1.: Classification of grid computing according to Foster et al. [2008]

intensive calculations and to increase the overall utilization of the hardware installed at each site by sharing it among partners. In order to facilitate this resource sharing a grid infrastructure was established.

A detailed introduction into the complex concepts involved with grid computing is beyond the scope of this thesis. Thus, the following section will only give a very concise summary of grid computing and focus on the problems users are facing when trying to access grid computing resources.

5.1.1. Grid Computing

Figure 5.1 illustrates that it is hard to classify the concept of grid computing and that it lies somewhere in the middle between super computing and clusters and the newer concepts of cloud computing and web 2.0. Hence, there is no clear distinction between those technologies and grid computing is closely related to those other paradigms.

Foster et al. [2003] define the problem underlying the Grid computing concept as “coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.”

In the case of the bwGRiD the shared resources were the High Performance Computing (HPC) clusters and software installed at different partner sites. At the beginning of the project the clusters at each site were rather homogeneous with respect to the hardware installed. Due to varying requirements at the partner sites the clusters were expanded in a heterogeneous way [Dynowski et al., 2012]. Allowing to integrate heterogeneous hardware into the grid is an advantage of grid computing. However, as soon as there is a heterogeneous hardware infrastructure there are certain resources that are better suited to run certain kinds of calculations than others. The decisions on which cluster is best suited to run a user’s calculation is not trivial and in the end has to be made by the user.

Another point of heterogeneity is the software installed at each site. While one software package might be available at one site it might not be installed at another site. In addition
5.1. Project Context

to this different license policies might allow a member of one partner institution to use a software at cluster A and B but not on the cluster of institution C.

The concept of virtual organizations needed for the authentication and authorization of users in the context of the bwGRiD was implemented using X.509-certificates [Adams et al., 2005]. A detailed description of the public key infrastructure involved with this kind of certificate is way beyond the scope of this thesis. In this context it is sufficient to say that getting hold of one of those certificates involved visiting a certain web page, filling out an application form with sensible information like your identity card number, printing said form, and finally find an authorized person to sign and further process the application. After the certificate request was granted users had to import the certificate into their browsers and apply for a membership to a Virtual Organization using said certificate. After the membership to the Virtual Organization was granted users were able to use the certificate to login to computers on the grid. However, in order to allow for the delegation needed to run “serious” computations users had to use their certificates to generate a so called proxy certificate. Those are short lived certificates that are transferred and stored on a MyProxy server and protected by a username and password [Novotny et al., 2001]. As it is not essential in the context of this thesis to understand this concepts, we also considered users trying to use the grid not to be especially interested in the technicalities of the authentication process. We expected users intending to use the bwGRID to run their calculations to rather be interested to get their work done quickly and not be bothered with extraneous material explaining these concepts.

To implement the objectives from the initial quote usually a so called middleware, like the Globus Toolkit [Foster, 2005], gLite [Laure et al., 2006], or UNICORE [Erwin, 2002] is used. In case of the bwGRiD the Globus Toolkit was chosen since it supports X.509 based authentication [Dynowski et al., 2012].

On the users’ side interaction with this Globus middleware usually involves installing and using a set of command line interfaces. Learning how to use this command line based programs imposes a high obstacle especially to first time users not familiar with this kind of interface [Altmann, 1987].

Taking all this into account the steps before users can utilise the resources provided by a grid usually are:

1. Get a certificate
2. Install the software prerequisites needed to interact with the grid middleware
3. Learn how to use the installed software
4. Use the installed software and the certificate to create a proxy credential
5. Write the script/code performing the actual computation
6. Define software, hardware and additional requirements that best suite your job
7. Select a site/queue that best suits the requirements of your computation
8. Use the command line programs installed earlier to submit your job to the grid
5. The Prototype System

9. Periodically check on the status of your job and wait till it is finished

10. Check your results, and restart from 5 to debug and refine your script/code

Thus it may be stated that the initial obstacles users have to overcome in order to use grid computing are rather high. Users have to learn a large number of extraneous material, not directly relevant to task completion, in order to employ grid computing facilities. Since it was one of the major objectives of the bwGrid project to promote the utilization of grid computing and thus to make the installed hardware available to a wide audience of scientists from various fields, different approaches were planned in the context of the bwGrid project in order to lower these initial obstacles. The efforts taken to achieve this objective will be illustrated in the next section.

5.1.2. The bwGrid Portal Project

To relieve users from the burden of installing software and using command line based applications to access the grid, a web portal based Graphical User Interface (GUI) has been implemented [Mosch et al., 2014]. As the portal was supposed to be used in teaching one focus was on easy learnability for first time student users not familiar with the concepts of grid computing and to introduce them to those concepts.

Using a web based interface that was accessible through standard web browsers offered various advantages over implementing a standalone application [Mosch et al., 2014]:

- Users were not required to install additional software.
- It was available regardless of users’ OS.
- Changes of the UI and features of the portal were instantly available to all users without updating the software.
- Changes to the volatile software structure of the grid could be integrated transparently into the portal and without requiring further actions on behalf of users.

To make software from various fields of application available through the GUI of the portal participating institutions contributed components for different aspects of computational science. Responsibilities were allocated as follows:

**Freiburg** microsystems technology

**Mannheim/Heidelberg** medicine

**Karlsruhe** engineering

**Stuttgart** workflow management

**Tübingen** bioinformatics

**Ulm** chemistry
5.1. Project Context

To foster easy integration of software components developed at these sites GridSphere [Novotny et al., 2004] was chosen to be used as framework for the portal. GridSphere is a so called portlet container [Wege, 2002]. The following sections will give a brief overview of the concepts involved in the context of GridSphere and portlet containers in general in order to give an impression of the framework the help system was integrated into.

5.1.3. bwGRiD Portal – Architectural Overview

As already mentioned the framework used in the bwGRiD portal was the portlet container [Wege, 2002] GridSphere [Novotny et al., 2004]. Portlets and portlet containers are closely related to servlets and servlet containers. In fact GridSphere itself is implemented as a servlet running on top of a servlet container. In case of the bwGRiD portal project this servlet container was tomcat 5.5 [Apache Software Foundation, 2004]. Thus, to understand the concept of a portlet, servlets should be introduced first.

5.1.4. Servlets

Servlets are specified by Java Specification Request (JSR)-154 [Coward, 2001] and define a way to use Java programs on a server in order to generate responses to client request. While Servlets are not limited to handle Hypertext Transfer Protocol (HTTP) requests they are usually employed in a web context generating HTML content as responses to HTTP GET or POST requests.

A flowchart showing the steps involved processing a client request and generating the according response using a servlet container like tomcat 5 is depicted in Figure 5.2.

A user enters a Uniform Resource Locator (URL) into the address bar of his browser or clicks on a link referring to a given URL. The browser interprets the URL and connects to the server specified. Additionally it sends data related to the request to the server.
5. The Prototype System

The servlet container running on the server is responsible for mapping incoming requests to different servlets by matching the URL to patterns defined in web.xml files. Thus for example it is possible to have requests to URLs containing the pattern “/media/” to be handled by servlet A and requests containing “/text/” by servlet B. Hence, a servlet container is able to run multiple servlets for multiple purposes. Due to the self-contained and well-defined structure of servlets and the interoperability between servlet containers it is easy to deploy servlets and reuse them in different servlet containers on different sites.

After determining the suitable servlet for the given request the container calls a designated function of the servlet and hands over some data related to the request as arguments to this function. The servlet then generates the response data. Servlets can produce virtually any kind of digital output. While they usually generate HTML-pages they can also generate for example bitmap data or even videos on the fly. The response generated by the servlet is afterwards delivered to the user and presented in the user’s browser.

A minimal implementation of a servlet is displayed in Listing 5.1. The output of this listing would be basic HTML data, displaying the string “Hello world!” and numbers from 1 to 99 to the user.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("application/xhtml+xml;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<!DOCTYPE html>");
        out.println("<html xml:lang="http://www.w3.org/1999/xhtml"]");
        out.println("<!DOCTYPE html>html>");
        out.println("<!DOCTYPE html>html>");
        out.println("<!DOCTYPE html>html>");
        out.println("<head>Simple HTML Document</title>");
        out.println("<head>Simple HTML Document</title>");
        out.println("<head>Simple HTML Document</title>");
        out.println("<head>Simple HTML Document</title>");
        for(int i = 0; i < 100; i++) {
            out.println("<p>Hello World!</p>" + i);
        }
        out.println("</body>");
        out.println("</html>");
    }
}
```

Listing 5.1: Source code of a Java servlet

One disadvantage of using servlets to generate web pages is the fact that web developers have to learn the Java programming language in order to employ this framework. Another downside is that since the output is generated by printing strings to an output stream the program code can get quite messy and hard to debug, especially when it comes to formatting, well-formedness and validity of the HTML markup generated. This is
quite obvious having a look at Listing 5.1. The output is generated using `out.println` statements and it is quite hard to tell the actual output of the program by just checking the source code. In Listing 5.1 the initial `<html>` tag on line 10 for example is missing the closing “>”. This kind of errors is hard to spot and assistance like syntax highlighting is usually not available for this way of output generation. To overcome this issues JavaServer Pages (JSP) were introduced.

5.1.5. JavaServer Pages

JSP are specified by [Roth and Pelegri-Llopart, 2003] and add a layer of high-level abstraction to servlets allowing web developers not familiar with the Java programming language to utilize the servlet technology. The example of Listing 5.1 converted to a JSP would look like Listing 5.2.

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8"/>
  <title>Simple HTML Document</title>
</head>
<body>
  <p>Hello World!</p>
  <% for(int i = 0; i < 100; i++) { %>
    <p><%= i %></p>
  <% } %>
</body>
</html>
```

Listing 5.2: Sample of a JSP file

The Java source code generating HTML output has been replaced with HTML markup containing some additional markup to implement the program logic of the servlet (lines 9 to 11 of Listing 5.2). Internally JSP files are programmatically converted to Java source files and compiled as ordinary servlets. The output of this programmatic conversion of the JSP file from Listing 5.2 would then again look similar to the source code of Listing 5.1.

While the syntax of JSP pages is much cleaner than that of the Java source concerning output readability, it still contains parts that require the web developer to be familiar with the Java programming language. In the example of Listing 5.2 this is the `for` loop implemented on lines 9 to 11. The parts enclosed in “<% %>” are treated as Java source code and added to the programmatically generated source code of the resulting servlet as is. The JSP specification defines various ways to extend JSP files in order to add program logic. Without this functionality JSP pages would actually just be a complicated way to deliver predetermined content. However, in order to properly use JSP web developers then again have to be familiar with the Java programming language.

To add another layer of abstraction, allowing to extend the functional range of JSP in a transparent way that does not require web developers to learn Java, tag-libraries have been introduced.
5. The Prototype System

5.1.6. Tag-libraries

Tag-libraries are part of the JSP specification [Roth and Pelegri-Llopart, 2003]. They allow Java developers to implement additional tags that can be used within the scope of JSP files.

The example from Listing 5.2 using a tag-library instead of adding program code directly into the file would look like Listing 5.3.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8"/>
  <title>Simple HTML Document</title>
</head>
<body>
  <p>Hello World!</p>
  <c:forEach var="i" begin="1" end="99">
    <p>${i}</p>
  </c:forEach>
</body>
</html>
```

Listing 5.3: Sample of a JSP file using a tag-library

In line 1 of Listing 5.3 a tag-library is made available within the scope of the JSP file under the prefix “c”. Tags defined by this tag-library can now be used under the given prefix. This was done on lines 11 to 12 to achieve the for loop like behaviour of Listing 5.2.

A nice feature of tag-libraries is that they allow to implement tags offering every kind of feature and that they can be easily distributed and shared. The GridSphere portal framework for example comes with an extensive tag-library implementing functionalities like message boxes or tabbed panes.

5.1.7. Portlets

Servlets always generate a whole response to a request. This for example means in a web context, that the servlet returns the whole HTML markup making up the content of a web page. In case of web portals this behaviour is not optimal. As the general layout of web portals is usually the same for every page and only certain parts differ from page to page. To reflect this property of web portals Portlets were introduced and specified in JSR-168 [Abdelnur and Hepp, 2003]. Portlets are very similar to servlets except that portlets are only responsible to generate parts of responses. Figure 5.3 shows the process of response generation for a web portal using portlets. On top of the servlet container runs a so called portlet container. This portlet container is responsible to render the overall web page, to identify portlets that should be displayed on a given page, trigger those portlets to generate their output, integrate this output into the overall response, and layout the content generated by the portlets on the resulting web page.
5.2. Features of the Help System

Figure 5.3.: Flowchart of a portlet-based web application.

Figure 5.4 displays how a page containing two portlets was rendered by the GridSphere portlet container.

Similar to servlets, portlets adhering to the JSR-168 specification can easily be distributed and shared. This made them perfect for the concept behind the bwGRID portal project, with different sites contributing different components for various areas of application. Those components could easily be implemented as portlets, distributed and integrated into the portal. Thus there were for example portlets implementing UIs to access Amber [Weiner and Kollman, 1981], Ansys, Gaussian [Frisch et al.], or Geant4 [Agostinelli et al., 2003]. These portlets were delivered to the portal administrator and integrated into the portal. For the implemented help system this meant that the help content created collaboratively for applications from different sites had to be combined in order to establish a unified resource for user assistance.

5.2. Features of the Help System

The main objective of the prototype system was to provide users with assistance at various stages and to be useful to novice as well as expert users. For novice users the system was supposed to be able to assist them with quickly and efficiently attaining first results. After users have attained some initial results the system was supposed to be capable to assist users in refining these initial results to better suite their needs. The final functionality the system was supposed to implement, was to provide further information concerning concepts of the portal and application programs made available by the portal to interested users. How those objectives were achieved will be explained within the next section.
5. The Prototype System

Figure 5.4.: Page of GridSphere based web portal containing two portlets.

5.2.1. General UI

In the text based version of the help system as used in the productive version of the bwGRID portal the help system was displayed as a small portrait of a chemist in a lab coat and glasses (see Figure 5.5) when it was not active.

In this mode the agent did not perform any kind of idle animation. To make sure that the agent was not interfering with the content of the page, it was possible to move it to different positions by means of drag and drop. This manually set position of the agent was retained between page loads. Thus, the agent permanently remained at the spot chosen by the user. To indicate that the agent could be moved to another location arrows were superimposed when the mouse pointer hovered above the portrait of the agent (see Figure 5.6).

Figure 5.5.: Representation of the agent when inactive
5.2. Features of the Help System

To distinguish the help system from the UI of the portal we decided to use a rather dark colour scheme for the help system compared to the quite light colours used in the portal.

The “Assist me” button below the portrait of the agent activated the help system. After it was clicked the whole content of the current page was dimmed and an image of the agent, along with a speech bubble, and an interactive area below the image of the agent was displayed in a box in the center of the screen (see Figure 5.7).

At this stage the speech bubble contained some text introducing the agent as “Dr. Nick”, asking how the agent could help and told users that it was possible to enter a question in the interactive area below, or to select one of the proposals. This text was formulated using first person speech. When text was displayed in the speech bubble the representation of the agent moved its lips for an amount of time that was loosely related to the amount of text displayed in order to give a slight impression of the agent speaking to the user. Lip movement was implemented to be rather unnatural and erratic to match the comical nature of the representation of the agent and not to be perceived as uncanny.

There were always at least two proposed options by the agent. The first one was “Help me with an element of this page!”. This option will be discussed in Section 5.2.3. The second option that was always available was “Guide me through all this step by step!”. This option will be discussed in Section 5.2.2. It was possible for portlet developers and the site administrator to add arbitrary proposals to the list of options suggested (see Figure 5.8). Those proposals were context sensitive and could be defined at a page or at an element level. That way it was for example possible to suggest a question concerning how to get a certificate at the login page or an option opening an explanation of the structure of input files for Gaussian [Frisch et al.], when the user was at an input field that required him to provide syntax for such an input file. Using this approach, we ensured that the agent was always able to make appropriate suggestions in the given context.

The text input field above the proposed options allowed users to enter arbitrary questions. Search queries were used to search available help content using a keyword based approach. Results matching the query were displayed in the box below the search field and replaced the initially suggested options (see Figure 5.9). Questions and pages at which the questions were asked were anonymously logged at the server. That way it
5. The Prototype System

![Figure 5.7.: Representation of the agent after activation](image1)

Figure 5.7.: Representation of the agent after activation

![Figure 5.8.: The agent with an adapted list of proposed options](image2)

Figure 5.8.: The agent with an adapted list of proposed options
would have been possible to compile a list of actual Frequently Asked Questions (FAQ) for pages. The idea was to evaluate those questions and incorporate answers and suggestions into the system according to the questions actually asked by users. Regrettfully the bwGRiD portal did not reach the popularity and user base required to put this idea into action.

There were three different kinds of answers to questions that could be provided by the agent. The type of answer was indicated by the small symbol next to the option opening the answer in the interactive area below the agent. Those types were:

**Step by Step Guides** This kind of answer was indicated by prepending the option with the iconic representation of the numbers 1 and 2. Step by Step guides will be introduced in Section 5.2.2.

**Simple Answers** Simple answers were presented by replacing the text in the speech bubble of the agent. Accordingly, they were indicated by a small speech bubble in front of the option. Section 5.2.4 will give an overview of this kind of answer.

**Complex Answers** Complex answers were used to explain rather complicated concepts employing a slide show like approach. Hence a symbol resembling a screen was shown next to the option to distinguish this kind of answer. Complex answers will be discussed in Section 5.2.4.

The following sections will provide a further discussion of the features of the system and explain its intended use.

### 5.2.2. Step by Step Guidance for Novice Users

An integral part of the help system was its capability to guide users to complete tasks. Thus step by step guides were implemented that allowed users to quickly attain results...
5. The Prototype System

The first page of the UI of the bwGRiD Science Portal.

using a worked example based approach. As worked examples are especially useful for novice users (see Section 2.1.4) the major objective of step by step tutorials was to introduce users to elements of the UI essential for task completion. This way allowing users to distinguish less important elements of the UI from more relevant features, fostering “pareto efficient” usage of the portal.

At any given page of the portal users could start an appropriate step by step guide by selecting the “Guide me through all this step by step!” option displayed after the help system was activated (see Figure 5.7). Since this default option was formulated quite generic, portlet developers were encouraged to define more specific suggestions for their portlets. Thus, this default option was usually replaced by a more specific one like “Show me how to submit a Gaussian job!”, if the user was at a page containing a portlet able to submit this kind of job. Even if no options were defined explicitly a sophisticated algorithm (see Section 5.3.2) made sure that the default “Guide me through all this step by step!” option always started a meaningful and appropriate tutorial for the given page.

A typical first time user of the portal now logs in to the portal and has a particular objective, like “I need to run a Gaussian calculation on the grid”. After logging in he is confronted with the first page of the UI of the portal (see Figure 5.10).

This page contains a lot of text and some of the text is even marked in red to indicate importance. However, since the user did not log into the portal to read a lot of text, but to get some computational chemistry done with Gaussian, we expected that he will try
5.2. Features of the Help System

Figure 5.11.: The “NWChem/Gaussian” subtab of the “Chem” section of the *bwGRiD Science Portal*.

...to avoid reading the text and potential extraneous material introduced within the text and will just return to it, in case he experiences problems. Thus we assumed a typical first time user to first search the navigational area for tabs related to his objective. The given user now spots the “Chem” tab and the “Gaussian/NWChem” subtab and visits that page. At that page he is confronted with two warning messages and a form asking for information related to a thing called “MyProxy” (see Figure 5.11).

We considered a typical first time user of the portal to be quite frustrated at this stage. “MyProxy” has nothing to do with computational chemistry and a typical chemist quite surely is not aware of the concepts and requirements related to the “MyProxy” approach of grid usage authorisation. Thus, at this point a typical user is definitely in need for assistance. Usually, this would mean for this user to return to the introductory “Doc” page of the portal and start reading the text provided there. However, the warning messages of the page (see Figure 5.11) tell users to consult “the avatar (the small chemist icon)” for help if they needed assistance. After clicking the “Assist me” button below the icon of the agent, the help system is started and offers different options the administrator of the portal or the portlet developers considered useful at this stage (see Figure 5.12).

The first option is a generic option that offers help for specific elements of the page. This option is not likely to be appropriate for the hypothetical first time user and thus will not be selected by this kind of user. The second option perfectly reflects the objective...
5. The Prototype System

Figure 5.12.: The options for assistance offered by the agent after activation in the “NWChem/Gaussian” subtab of the “Chem” section of the bwGRiD Science Portal.

the user had in mind when he logged into the portal and thus the user will most likely choose this option. Selecting this option starts a step by step tutorial. This step by step tutorial consists of multiple dynamic parts, making sure that all requirements to submit a Gaussian job are met, a suitable Gaussian job is created and the user is enabled to check the status of running and completed jobs. Step by step guides were able to cover multiple portlets spread among multiple pages. It was important to us to implement this kind of behaviour, since if it was for example not possible for step by step tutorials to cover multiple portlets the only assistance that could be offered by the system at this stage would have been something along the lines “Show me how to retrieve a MyProxy credential!”. However, even if this kind of assistance would still be useful it does not reflect the user’s intention of running a Gaussian job on the grid and would not be appropriate to adequately assist users in reaching their ultimate objectives. It would also be hard for users to realize how creating a credential was a necessary step required to complete their tasks.

During step by step guides all elements of the UI essential to complete the given task are distinctively marked and sequentially numbered (see Figure 5.13). Each labelled element was sequentially visited by the agent and the agent provided the information required to provide correct input for the element. To make the sequential visiting of elements during step by step guides easier the sequence of input fields visited by hitting the Tab key of the keyboard was adapted to reflect the sequence of elements that needed editing. Additionally the fields could also be activated by clicking the orange markers or
5.2. Features of the Help System

Figure 5.13.: The agent explaining all elements of the interface that need to be edited to achieve a given objective.

using a “Move on to the next step!” or “Go back to the previous step!” option offered in the interactive area below the depiction of the agent (see Figure 5.13).

After the first part of the step by step guide of the example was finished and the user acquired a credential, the actual UI of the “Gaussian/NWChem” portlet becomes active and the warning messages disappear (see Figure 5.14). The agent now automatically proceeds with the guide and again visits all elements of the UI that need to be edited. While for the first part of the guide virtually all input elements of the “MyProxy” portlet are important for task completion, Figure 5.14 demonstrates a major advantage of the given help system. For the “Gaussian/NWChem” portlet only some of the controls were actually required to get a first job running on the grid. If a user not employing the help system was confronted with this kind of interface, there would be the typical problem to distinguish between essential UI elements and elements added to the interface to enable extra features. Take for example the “Upload File” element in the lower part of Figure 5.14, a typical first time user might wonder if it is essential to upload such a file before running a job and what the contents of such a file were supposed to be. The same is true for the “Maximum memory” and “Number of Nodes” parts of the interface. The user just is not sure if he has to worry about the values entered there and if he has to adapt them, even if he was just about running a small first test job, to verify if everything worked and was set up correctly. By indicating that these elements are not of major importance at first, the help system helps users to identify essential interface elements and efficiently work with the system in order to attain first results quickly. Another aspect of importance in this context is the system’s capability to provide users with exemplary input for input fields. This is for example crucial for the “Input data” field of
5. The Prototype System

Figure 5.14.: The agent proceeding to explain the input fields needed to achieve the given objective, after the UI of the “Gaussian/NWChem” portlet became available.

the “Gaussian/NWChem” portlet. This field enables users to enter the syntax of a Gaussian input file defining the actual computation that was supposed to run on the grid. When the agent stepped over an input field for which exemplary input was available, an option indicating that the agent was able to provide exemplary input for this field was displayed in the interactive area below the image of the agent (see Figure 5.15). Selecting this option filled the input field with example data that was guaranteed to work under the given settings. This template data could later be refined by the user and was able to demonstrate what kind of input was expected for this input field. Having an example guaranteed to work under the given conditions was also nice for debugging reasons. In case a user ran a calculation that failed he would be able to verify if the reason for the failure was his script or something else, by using the example script provided by the agent. If this script also failed the reason must be something else and if the exemplary script worked the reason must be the user’s script.

The help system was also able to relief users from other deficiencies in the underlying UI of the portal. Take for example the “Maximum Wall Time” element of the “Gaussian/NWChem” portlet from the example. This time was supposed to be entered in seconds. Calculating longer runtimes of for example days would have required the user to use a calculator to determine an appropriate value. The help system allowed users to enter this time by specifying days, hours, and minutes separately (see Figure 5.16). Of course one could argue that it is better to change the underlying UI to allow to specify times in a human readable format and this was actually also the approach chosen in
5.2. Features of the Help System

Figure 5.15.: The agent proposing to paste a working example for a Gaussian input file that was required for the input field.

Figure 5.16.: The help system allowing users to enter “Maximum Wall Time” by days, hours, and minutes instead of seconds.
5. The Prototype System

Figure 5.17.: The help system helping users to navigate to portlets involved in a step by step tutorial.

case of the portal. Actual portlets usually allowed users to enter times in a “dd:hh:mm” format instead of using seconds and the format was changed for demonstration purposes in this example. However, even this approach was not the perfect solution. Consider for example a user that actually measured the runtime of a script. The result of this measurements would quite surely be in seconds. Thus, to enter a correct value into the “dd:hh:mm” input field this user would also need a calculator. The functionality of the agent in this case would than be the other way around, allowing users to enter times in seconds. Since the format of this kind of input field was changed quite late in the development process of the portal, this possibility was not implemented in the help system. This aspect nevertheless shows that to increase usefulness of help systems this kind of additional assistance can and should be provided and the help system is a perfect place to add this additional functionality.

In case a step by step tutorial involved portlets on different pages the agent told users to navigate to the pages containing this portlets (see Figure 5.17).

After a step by step guide was completed the UI of the portal was dimmed again, the agent was presented in the center of the screen and two options were offered to dismiss the agent in the interactive area below the image of the agent. A positive option that contained a “Thank you...” text and a derogatory option that contained a “Pff...” message. Those options were added to evaluate social responses towards the agent (see Chapter 12).

During step by step guides users were able to relocate the agent to keep him from interfering with the UI of the portal. Similar to the agent in minimized state an image of arrows was superimposed, when the mouser pointer hovered over the image of the agent (see Figure 5.18).

Additionally the Ctrl key of the keyboard could be used to bring help markers to front and make them clickable (see Figure 5.19).

Step by step guides could be interrupted anytime by the user by clicking the minimize button of the agent visible when the mouse cursor hovered over the area of the screen.
5.2. Features of the Help System

Figure 5.18.: The help system, when the mouse pointer hovers over the representation of the agent. Arrows to indicate movability and “minimize” button in the top right corner.

Figure 5.19.: Markers brought to front using the \texttt{Ctrl} key.
5. The Prototype System

Figure 5.20.: Representation of the agent when inactive with added option to resume an interrupted step by step tutorial

containing the agent (see top right of Figure 5.18). If a guide was interrupted users could resume it anytime using the “Resume” button that was displayed in the lower part of the representation of the help system in idle state (see Figure 5.20).

The orange numbers displayed at the top right of the inactive agent (see Figure 5.5) served as shortcut to the first step by step guide available at the given page. When a user clicked this shortcut the guide was started immediately. To indicate this relation between the shortcut and the first step by step guide proposed, the first guide proposed was always coloured orange in the list of options displayed after the help system was activated (see Figure 5.12).

After the step by step guide was finished the user visiting the portal to run his Gaussian script, had attained a credential, submitted his first job to the grid and knew how to monitor the progress of his job on the grid. This is what this user came to do at the portal and what definitely might have taken him more time than it would have taken him using the portal without the help system. After submitting this first exemplary job, the user might feel the need to refine the results attained after this worked example by adapting the example script provided by the help system and by making further adjustments using the other controls of the UI that were not essential for his first jobs. At this stage we considered exploratory learning of UI elements to be an appropriate approach for users to find out about these additional features. The knowledge of essential UI elements attained processing the step by step guides was supposed to be sufficient to guide users during this exploratory learning process.

5.2.3. Assistance to Refine Initial Results

Since the objective of the step by step guides is to help users to accommodate with the UI quickly and efficiently those guides deliberately do not cover all aspects of the UI of a portlet. However, those aspects were added to the UI by a reason and are thus useful to refine the initial results attained after using the step by step guides. To provide users with information on interface elements not covered during step by step guides the “Help me with an element of this page!” feature was added to the help system. When a user clicks this option elements for which the help system can provide additional information
5.2. Features of the Help System

![Figure 5.21.: Representation of the agent after the “Help me with an element of this page!” function was activated.](image)

are distinctively marked using a similar marker as for the elements of step by step guides (see Figure 5.21). In this mode it is also possible to relocate the agent and bringing the markers to front similar to this functionality during step by step guides. After the user selected one of the marked elements the agent is displayed right next to the element and provides further information about this element similar to the information provided during step by step guides (see Figure 5.13). While the default behaviour is to display the same information in the “Help me with” mode as during step by step guides, it is also possible to provide different help texts in this mode. That way enabling for example a condensed help text covering only essential aspects of an input field during the step by step guide for novice users and a more in depth explanation for users explicitly asking for additional information on an element.

This feature is similar to the “What’s this” feature (see Section 3.3) implemented in other software products. However, since all elements for which further information is available are distinctively marked, it avoids the problem of visibility that arises in the context of traditional “What’s this” implementations. While additional information is usually just provided for input fields and elements of the interface that offer some kind of interactivity, like buttons, the system is flexible enough to allow to add information to virtually every part of the UI, even floating text. This is for example useful to provide further information on parts of the text that might not be directly understandable by
5. The Prototype System

Figure 5.22.: The agent explaining the meaning of a term used in the text of the UI.

normal users, like the concept of a mebibyte that was mentioned in the “Maximum Memory” part of the UI (see Figure 5.22). Of course this information can also be added to the text provided by the system, when the user asks for help concerning the “Maximum Memory” input field, but this would bloat the help text provided for this input field and reduce coherence. Apart from the specifics of using the MiB approach to measure a megabyte there is still enough information that should be covered in the context of the “Maximum Memory” input field and sourcing out the MiB information makes this information more easily accessible.

The small question mark on the top right of the portrait of the inactive agent serves as a shortcut to activating this feature, without having to activate the help system and choosing the “Help me with an element of this page!” option from the list of suggestions made by the help system after activation.

Using the “Help me with” feature of the system users are able to efficiently refine the results attained after using initially step by step guidance. However, up to now they are not familiar with the underlying principles of the portal. They are for example able to retrieve a credential and submit jobs and thus are able to use the portal to get work done. Nevertheless, we also wanted the help system to be able to provide further information covering aspects of the portal not directly related to task completion. How the help system could be used to teach users this more in-depth kind of knowledge will be demonstrated in the next section.
5.2. Features of the Help System

Figure 5.23.: The help system answering a user question in the speech balloon of the agent.

5.2.4. Providing Further Information to Interested Users

Allowing the help system to cover further aspects of the portal, the agent was able to answer a variety of additional questions not directly related to task completion. Those questions could either be related to concepts of grid computing, like credentials or additional information on the portlets made available by the portal. Questions could either be entered directly using the search box displayed in the interactive area below the image of the agent after the help system was activated (see Figure 5.12) or suggested by the agent, on a portlet or element level. Thus if a user for example was on a page containing the “MyProxy” portlet a suggestion related to the credential mechanism could be suggested by the agent (see Figure 5.12), or if the user selected an element that required him to enter a Gaussian input file an option explaining the structure of such an input file could be added (see Figure 5.15). Suggesting possible questions to users gave an impression of the kind of question the help system was able to answer and thus could help users formulating their own questions.

At this stage there were two different relevant types of answers provided by the help system.

Short answers replaced the text in the speech balloon of the agent (see Figure 5.23). It was possible to include HTML markup and links into this kind of answer.

For more complex concepts a second kind of answer was provided. Selecting one of the options prepended by the small “screen” symbol (see for example the last option proposed by the agent in Figure 5.23) opened a slide based presentation containing text and images and the agent was able to make remarks on arbitrary elements of each slide.
5. The Prototype System

Figure 5.24.: The help system explaining the concept of certificates and MyProxy credentials to the user.

This way the agent served as a lecturer introducing users to complicated concepts. The agent could be moved forward and backward during the explanation process using the buttons provided in the interactive area below the image of the agent or the Tab key of the keyboard. To allow users to quickly jump to certain parts of the explanation each slide had a button in the top left corner allowing users to activate markers for each part of the slide explained by the agent (see Figure 5.25).

If such a presentation was closed by the user, it could easily be resumed clicking the “Resume” option added to the presentation of the help system in idle mode (see Figure 5.26). The explanation then continued at the same spot as it was left before.

Using this approach the help system was able to provide interested users with further information regarding the underlying principles of the portal and grid computing as well
Figure 5.25.: The help system in complex answer mode with sequential markers activated.

Figure 5.26.: The option to resume a presentation added to the UI of the help system in idle mode.
5. The Prototype System

![Image of the help system with spoken output enabled.](image)

Figure 5.27.: The UI of the help system with spoken out- and input enabled.

as information on the application programs integrated into the portal. While this kind of information is not essential for task completion interested users might nevertheless want to know something about the systems they are using. This kind of information can also be useful in case of errors. As one of the aspects of the portal was to be used in teaching the ability of the help system to introduce students to the concepts of grid computing was also of importance.

5.2.5. Spoken In- and Output

A feature that was not used in the help system shipped with the productive version of the portal, but that was implemented out of usability and evaluation considerations was spoken in- and output for the help system. The modality principle (see Section 2.1.3) suggests that it might improve learning and usability by reducing cognitive load on users, if the help system uses a different modality for interaction than the actual UI of a program. When spoken in- and output was enabled the text in the speech balloon was replaced with a process indicator (see Figure 5.27) and the output that would usually be written in the balloon was spoken out loud by the agent. Users were able to seamlessly switch between spoken and textual output clicking the button on the lower left part of the speech balloon right to its pin. In addition to the text in the speech balloon the options displayed within the interactive area below the image of the agent were also read out loud. Users could choose one of these options by saying the number prepending the option.

The controls displayed in the speech balloon allowed users to pause, rewind, and jump to an arbitrary location of the speech output by clicking on the progress bar, or jump to
5.2. Features of the Help System

a specific option by clicking the numbers above the progress bar. The process indicator of the progress bar was at the point labeled “1” when the speech output of the agent was explaining the option labeled “1” in the interactive box below the image of the agent.

It was possible to define different output for the spoken and textual modalities. This was for example useful to avoid lengthy recurring options like “Move on to the next step!” to be replaced with shorter options like “Next” for spoken output. It was also useful in case the textual output contained additional HTML markup. While this markup was usually just ignored for spoken output, an \textit{HTML} tag emphasizing enclosing text could have been replaced with a \textit{stress} marker in the text defined for spoken output. That way, retaining the stress of importance on the emphasized part of the textual output for the spoken modality.

5.2.6. Proactivity

Another feature of the system that was disabled in the version of the help system used on the productive portal was its capability to proactively offer help to users. Proactivity might be a useful feature for help systems to make users aware of the existence of the help system and to introduce the users to its capabilities. However, since the need for assistance for first time users in the context of the portal was quite obvious, considering the complexity of grid computing, we decided to not use this feature in the productive portal. Instead we decided to suggest users to use the help system at points in the interaction process where we expected a further need for assistance, like in case of warning messages (see for example the warning messages in Figure 5.11). Accordingly, we considered offering help proactively to be dispensable in case of the portal.

5.2.7. Input Validation

A feature that was integrated into early versions of the help system was to validate input on the client site. If a user for example entered numbers into an input field that only allowed for alphabetic character input the help system was activated and the agent pointed out that only alphabetic character input was allowed for this kind of input field. While the feature was not removed from the functionality set of the agent, its use was discouraged for various reasons. First of all adding this feature required additional effort on the side ofportlet developers and a server side input verification was required nevertheless. Thus, we decided that input verification should solely be performed on the server side and the server side should inform users of malformed input. The task of the help system was to inform users of restrictions to input fields only if users explicitly asked for help concerning this input field. That way we also wanted to prevent the agent from being perceived as “smart-ass” and not give users the impression of being monitored by the help system.

5.2.8. Conclusion

The implemented help system is able to assist users at various proficiency levels. First time users are enabled to quickly accommodate with the UI of the portal and to quickly
5. The Prototype System

attain first results, without having to acquire an in-depth knowledge to the concepts of grid computing first. Building on these initial results users can refine their jobs using additional features of the system. The help system is able to assist users in exploring these additional features by providing help with elements of the UI. In case users are actually interested in the concepts involved with grid computing the system is also able to introduce them to these concepts. All this functionality is integrated into a single point of reference users can refer to in case they feel a need for assistance.

Compared to providing help using a help pages based approach, assistance is provided along with the actual UI of the program. That way users do not have to switch their focus back and forth between the help pages and the UI of the program. Pointing out the elements that need to be manipulated to achieve given objectives spares users from searching elements of the UI mentioned in tutorials of help pages based on images or textual descriptions. This way decreasing extraneous cognitive load on users and maximizing the potential benefits of using worked examples during learning. This aspect was also stressed by users when asked what they would like about the help system. A lot of users mentioned that they liked that “the agent pointed at things”.

5.3. Technical Realization

To allow portlet developers to easily add the functionalities of the help system into existing application specific portlets and to integrate the help system into the existing portal framework - as described in Section 5.1.3 - various components were developed.

5.3.1. Architectural Overview

As the system was web based the components of the system were separated into a client side part and a server side part. This distribution makes it hard to apply a traditional design pattern to the structure of the help system. Nevertheless, the system was designed following the principles of the Model-View-Controller (MVC) paradigm [Reenskaug, 1979; Krasner and Pope, 1998] to allow to easily port the implemented components to different architectures. The idea behind the MVC pattern is to split applications into three logical components as illustrated in Figure 5.28. One component is responsible to generate and hold the model of the application. This model is manipulated by the user by means of the controller components and the last part, the view component is responsible to generate a view of the model that is presented to the user. By separating these parts and defining the interfaces between the three components it is for example possible to easily change each component without having to change the other components as well, as long as the interface used to transfer information between components does not change.

Another aspect of the help system was that different entities were manipulating the model of the portal in different ways. While users changed the state in a user and session specific way, site admins changed the general model of the portal by changing its layout or by adding and removing portlets. Both kinds of changes to the model had to be reflected by the help system. Figure 5.29 illustrates the interactions between components in the context of the client server based architecture of the system.
5.3. Technical Realization

Figure 5.28.: Visualisation of the MVC pattern.

Figure 5.29.: Interaction of components from the perspective of the help system.
It is important to note that the controller component made no changes to the model of the portal. This way the query mechanism from the client-side controller to the server-side model could be implemented in a stateless and idempotent manner.

The model of the help system on the server-side consisted of three parts. The first part was the state of the portal in general and information on portlets made available through the portal. This information was required by the help system to suggest appropriate help content, to generate step-by-step tutorials and to assist users in navigating to the portlets occurring in these step-by-step tutorials. It also contained the set of answers available for user questions. How this part of the model was maintained, kept up-to-date and passed to the client to make up the client side model of the help system will be explained in Section 5.3.2.

The second part of the model on the server-side was the state of portlets and users progress using them. It contained the explanations available for elements of the UI of the portlet along with their sequence for step-by-step guides. This information was needed by the help system to verify if a user has successfully finished using a portlet and can proceed to the next portlet in step-by-step guides. This part of the model will be discussed in Section 5.3.2.

On the client side the model comprised information on the current state of the help system, like the current position in the step-by-step guide of a portlet, or the currently active mode of the help system (e.g. idle, explaining elements, step-by-step tutorial). Further details on this part of the model will be given in Section 5.3.3.

The client-side of the help system was implemented completely using JavaScript [ECMA International, 2011] and did not require additional plugins to be installed on the client. An overview of this component will be provided in Section 5.3.3.

In case of the controller part of applications and thus also for the help system this component was mostly predetermined by the framework used for the portal. Different aspects of the controller will be discussed at the appropriate points throughout the following sections.

5.3.2. Server-Side Implementation

The server-side implementation seamlessly integrated into the framework used by the portal. It consisted of several components that will be elucidated within the next sections.

Server-Side Model – Portal Layout

To allow the help system to keep track of the layout of the portal, installed portlets and the relation between portlets a twofold approach was implemented. To determine the layout of the portal and the distribution of portlets among pages, a file defining this layout is scanned each time a query from the controller component of the client is issued to the server. This way it is guaranteed that the model of the help system always reflects the current state of the portal and changes to the layout are automatically incorporated into the client- and server-side models. Since queries for this kind of information are rather rare a pull based approach was preferred over a push based approach notifying
5.3. Technical Realization

the help system of changes to the portal layout, as it is easier to maintain and implement in case of the portal.

Server-Side Model – Portlet Relations

Information on portlets and their relation is gathered from avatar.xml files that are part of every portlet incorporating help system functionalities.

```
<portletAppAvatarDescription>
    <portlet class="de.uniulm.bwgrid.portal.portlets.GaussianNWChem">
        <description key="Description"/>
        <keywords>gaussian, nwchem, submit, job</keywords>
        <marker>gaussianNWChem</marker>
        <prev>MyProxy</prev>
        <next>JobMonitor</next>
        <optionSelectText key="DefaultOptionSelectText"/>
        <sbsPath>
            <marker>
                useExistingFile
            </marker>
            <keywords>gaussian, nwchem, existing, input file</keywords>
            <optionSelectText key="UseExistingOptionSelectText"/>
            <prev>MyProxy:createDifferentCredential</prev>
        </sbsPath>
    </portlet>
</portletAppAvatarDescription>
```

Listing 5.4: Sample avatar.xml file.

This file establishes a connection between the help system and the portlet implementation using the fully qualified name of the class implementing the portlet (see line 2 of Listing 5.4). The key attribute refers to strings defined in separate files and facilitates multilinguality. This is the same approach for multilinguality as available in GridSphere and portlet developers only have to add help system specific strings to already existing files. The keywords defined on line 4 are needed to enable the keyword based search feature of the help system. In case the step by step guide of the portlet is in the list of results of a user search, the text of the option displayed in the interactive box of Figure 5.8 is defined by the <optionSelectText> tag in line 14. The <marker> tag is used to give a simple name to the portlet. This name is used by other portlets to refer to the given portlet within the <prev> and <next> tags. These tags define the portlets users have to complete, before the given portlet could be processed or after interaction with it is completed during step by step tutorials. In case of the example in Listing 5.4, this means that a user has to complete the default step by step guide of a portlet with the
5. The Prototype System

marker “MyProxy” before he can use the “gaussianNWChem” portlet. After completing interaction with the “gaussianNWChem” portlet, the step by step guide of a portlet with the marker “JobMonitor” has to be completed. In addition to this default step by step guide, it is possible to define multiple step by step guides for portlets using the <sbsPath> tag of line 15. To refer to different step by step tutorials of other portlets in the <prev> and <next> tags of step by step guide definitions, the explicit step by step tutorial name is preceded by the marker of the portlet separated by a colon (see line 21).

Avatar.xml files of portlets are constantly monitored during server runtime and changes are instantly propagated to be reflected in the model of the help system.

Server-Side Model – Answers to User Questions

Answers to user questions are collectively stored in a folder that is searched, when the user submits a query from the search field of Figure 5.7. Answers can be provided using multiple languages and the language of answers is determined from the filename of the answer. According to the kind of answer (see Section 5.2) the files contain two different types of content. An example of a simple answer is shown in Listing 5.5.

```
<av:answer xmlns:av="http://uni-ulm.de/bwgrid/portal/avatar"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://uni-ulm.de/bwgrid/portal/avatar/schema/
    answer.xsd"
    marker="getCert">
    <av:keyWords>
        certificate
    </av:keyWords>
    <av:optionSelectText>
        Show me where I can get a certificate!
    </av:optionSelectText>
    <av:simple>
        Getting a certificate is quite complicated and differs from site to site.<br/>
        Links to further documentation concerning this topic can be found on
        <a href="http://www.bw-grid.de/en/bwgrid-users/access/" target="_blank">
        this page</a> of the bwGriD home page.
    </av:simple>
</av:answer>
```

Listing 5.5: Sample file containing a simple answer to a user question in English.

Similar to the avatar.xml file of Listing 5.4 files, these files define a marker used to refer to the answer within the help system. Those files also contain a <keywords> and <optionSelectText> tag used for the same purpose as the corresponding tags of the avatar.xml file. The major part of the file consists of the actual answer within the simple tag (lines 12-12). The text defined there is the text displayed in the speech bubble of the agent, after the user selects the option referring to this answer (see Figure 5.23).
This kind of answer can contain arbitrary HTML markup. However, since the text is rendered within the speech bubble of the help system content can not be very complex.

For more complex answers another type of file is used. A file defining a complex answer is listed in Listing 5.6.

```
<?xml version="1.0" encoding="UTF-8"?>
<av:answer xmlns:av="http://uni-ulm.de/bwgrid/portal/avatar"
  xmlns="http://www.w3.org/1999/xhtml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://uni-ulm.de/bwgrid/portal/avatar/schema/
  answer.xsd"
  marker="generalCredential">
  <av:keywords>
    How, credential, work
  </av:keywords>
  <av:optionSelectText>
    How does the credential thing work?
  </av:optionSelectText>
  <av:complex>
    <av:slide>
      <av:title>
        <span id="headingSlide1">The Authentication Dilemma</span>
      </av:title>
      <av:content>
        <img src="$media$/credential_layout1.png" id="image1" />
      </av:content>
      <av:explainPoint>
        <av:text>
          Before I explain how the actual...
        </av:text>
        <av:script>
          document.getElementById("image1").src="$media$/
          credential_layout1.png"
        </av:script>
        <av:ref>heading1</av:ref>
        <av:pos>EAST</av:pos>
      </av:explainPoint>
      <av:explainPoint>
        <av:text>
          Imagine you are...
        </av:text>
        <av:script>
          document.getElementById("image1").src="$media$/
          credential_layout2.png"
        </av:script>
        <av:ref>image1</av:ref>
        <av:pos>CENTER</av:pos>
        <av:posX>15</av:posX>
        <av:posY>26</av:posY>
        <av:width>120</av:width>
        <av:height>167</av:height>
      </av:explainPoint>
    </av:slide>
```

Listing 5.6: Sample file containing a complex answer to a user question in English.

The preamble of complex answers is the same as for simple answers. The main difference is that the content is defined within a `<complex>` tag (see line 13 of Listing 5.6). Following the presentation metaphor used in this context a `<complex>` tag comprises a number of sub tags called `<slide>`. Each slide has a title (see line 14), and content for this slide that consists of arbitrary HTML markup. Thus a slide can virtually contain anything, from text over images to videos. The content of the slide from the exemplary Listing 5.6 is a single image, as defined on line 18. In addition to the content rendered to the user, each slide features an arbitrary number of `<explainPoint>` tags. These define the points of the slides explained by the agent during the presentation. The points are visited by the agent following document order. This means the point occurring first in the list of `<explainPoint>` tags, is the point explained first by the agent. To refer to an element the agent is supposed to point to, when a point in the list is reached during presentation, `<explainPoint>` tags contain a `<ref>` tag referencing arbitrary elements of the slide based on the id attribute of these elements. A complementary `<pos>` tag specifies to which part of the element the agent should point to. Possible values are for example CENTER, NORTH_WEST, NORTH, NORTH_EAST and so on. To enable the agent to point to arbitrary points within elements of a slide it is also possible to define areas within elements as point targets using the tags of lines 39–43 from Listing 5.6. This is for example useful if the content of the slide consists of a single image and the agent is supposed to explain different parts of that image. To increase interactivity of presentations each `<explainPoint>` can contain a `<script>` tag. The content of this tag can be arbitrary JavaScript code that is executed, when the point is reached during presentation. This allows for example to change images displayed on slides during presentation, as it is shown on line 26 of Listing 5.6.

At the client complex answers are then presented to the user like it is shown in Figure 5.24 and Figure 5.25.

Server-Side Model – Portlets

To add content for the help system to portlets a tag library was implemented. This tag library allows portlet developers to easily add the features provided by the help system to their portlets. Tags of this library can be added to existing JSP files (see Section 5.1.3). The library contains the following tags:

- `<init>` This tag was added to the library for convenience reasons only and could be used to initialize the help system if no other help system related tags were used.

- `<compDesc>` This tag is the most important one. It is used to define explanations for elements of the UI of portlets for the “Help me with an element” and step by step guides of the help system. It is also used to define the sequence of elements visited during step by step guides.
<conditionalFinished> This tag is used to indicate if a step by step guide is finished. It checks if a flag was set by the program logic of a portlet and sets a marker indicating successful completion of the step by step guide of a portlet.

<finished> The functionality of this tag is essentially the same as for the conditionalFinished tag, without the check for the flag set by the program logic.

<questSuggest> This tag allows portlet developers to specify questions that should be suggested by the agent as soon as a user requests help on a page containing the portlet.

Listing 5.7 presents a minimal example of a JSP file utilising the features of the help system.

```
<%@ taglib uri="http://uni-ulm.de/bwgrid/avatar/" prefix="av" %>
<%@ taglib uri="/portletUI" prefix="ui"%>

<av:questSuggest key="key1" id="id1" type="COMPLEX" marker="generalCredential"/>
<av:questSuggest key="key2" id="id2" type="SBS" marker="gaussianNWChem"/>

<ui:form>
  <input type="text" maxlength="40" size="20" id="input" />
  <av:compDesc ref="id:input" next="submitDesc" id="inputDesc" key="key3"/>
  <ui:actionSubmit action="doFinishPortlet" key="key4" id="submit"/>
</ui:form>

<av:conditionalFinished />
```

Listing 5.7: Minimal example for a JSP file utilising the features of the help system.

Line 1 makes the tag library of the help system available under the prefix av. To inform the help system that it should propose two options for assistance to users, if they request help on a page containing the given portlet the <questSuggest> tag is used on Line 4 and Line 5. One of these options refers to a complex answer with the marker "generalCredential" from Listing 5.6 and the second option starts the default step by step guide of the portlet marked as "gaussianNWChem" as defined in Listing 5.4. To specify additional information for elements of the UI the <compDesc> tags are used. These tags refer to the elements of the UI utilising the value set for the id attribute in referenced elements in their own ref attribute. Thus, the <compDesc> tag of line 9 in Listing 5.7 refers to the element with id="input" defined on line 8 and the description from line 11 to the element with the id="submit" also defined right above the definition of the description on line 10. The <compDesc> tag also specifies the sequence of elements visited by the agent during step by step guides utilising the next attribute. Hence, the sequence for the step by step tutorial of Listing 5.7 would be:

```
inputDesc ⇒ submitDesc
```
The `compDesc` tag offers a lot more attributes to customize the behaviour of the help system. It is for example possible to define different texts to be displayed if the agent explains an element during a step by step tutorial or when the user requests help for this particular element in the “Help me with an element!” mode. It is also possible to define text different to the text in textual output, when the speech output feature of the system is enabled. A `type` attribute allows developers to specify a certain type of help marker that should be displayed when the element is visited. Like the help marker assisting users when entering durations as shown in Figure 5.16. There are also attributes enabling developers to let the help system suggest options, when the element referred to by the description is visited, and attributes to facilitate the sample input data for input fields functionality as shown in Figure 5.15.

Tags related to the help system are transformed into hidden HTML `<div>` elements, integrated into the reply created by the portlet and delivered to the client. Examples for the HTML output generated for the `<questSuggest>` from line 4 and the `<compDesc>` tag defined on line 9 of Listing 5.7 is shown in Listing 5.8 and Listing 5.9 respectively.

```
1 <div style="display: none;" id="avatar_id1_suggestedQuestion">
2  <div id="avatar_id1_portlet">avatar_demo#GaussianNWChem</div>
3  <div id="avatar_id1_value">Show me how to run a Gaussian\NWChem job on the grid!</div>
4  <div id="avatar_id1_type">SBS</div>
5  <div id="avatar_id1_marker">gaussianNWChem</div>
6 </div>
```

Listing 5.8: HTML output generated for the `<questSuggest>` tag from line 4 of Listing 5.7.

```
1 <div id="avatar_gaussianNWChem_compDesc" style="display: none;">
2  <div id="avatar_gaussianNWChem_compDesc_portlet">avatar_demo#GaussianNWChem</div>
3  <div id="avatar_gaussianNWChem_compDesc_text">Text of explanation</div>
4  <div id="avatar_gaussianNWChem_compDesc_ref">id:input</div>
5  <div id="avatar_gaussianNWChem_compDesc_next">submitDesc</div>
6 </div>
```

Listing 5.9: HTML output generated for the `<compDesc>` tag from line 9 from Listing 5.7.

The last tag in Listing 5.7 related to the help system is the `<conditionalFinished>` tag on line 14. As already mentioned this tag is used to inform the help system if the step by step guide of a portlet was finished successfully. Therefor it checked if a flag was set in the program logic of the portal and adds corresponding output to the resulting page.

On the client side the content generated by the tags is integrated into the client side model and interpreted by the `view` component of the client side implementation.

Adding the information required by the help system directly into source code making up the UI of the portlet has various advantages. First of all it closely couples the help content to the actual UI. Changes to the layout of the UI are automatically reflected by the help system. There is no need to make for example new images or update the content of help pages to reflect the changes. The fact that the help content is provided in close
proximity of the elements defining the UI, reminds developers to update the content as soon as they make changes to the functionality of elements. This way integrating the maintenance of help content into the workflow of UI development.

**Server-Side Model – Querying**

To allow the controller component of the client-side to retrieve information concerning the server-side model a Servlet was implemented. This servlet is accepting HTTP POST requests and generates corresponding responses according to the current state of the portal. The content of such a POST was in Extensible Markup Language (XML). Listing 5.10 shows an example for the syntax of such a request.

```
<av:question xmlns:av="http://uni-ulm.de/bwgrid/portal/avatar"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <av:text lang="en">What%20is%20a%20credential%3F</av:text>
  <av:context>
    <av:navPosition>
      <av:layoutPage>logged in</av:layoutPage>
      <av:nav>3</av:nav>
      <av:subNav>1</av:subNav>
    </av:navPosition>
    <av:url>
      http%3A%2F%2Flocalhost%3A9080%2Fgridsphere%2Fgridsphere%2FloggedIn%2F85%2F</av:url>
  </av:context>
</av:question>
```

Listing 5.10: Request as issued to the Servlet of the help system.

This syntax is quite simple and consists mostly of two parts, the first part contains the text entered by the user (line 4 of Listing 5.10) and the second part comprises information on the page at which the question was entered (lines 5–13). Question and page were logged at the server. The idea behind this is to evaluate this information and to compile a list of FAQ and add additional answers to the system based on this data. The request from Listing 5.10 was triggered by a user entering a search phrase into the search field available after the help system was activated (see Figure 5.8).

Upon receiving such a request the help system searches the set of available step by step guides and answers for matching keywords according to the algorithm described in Section 5.3.2 and generates a response. An example for such a response is displayed in Listing 5.10

```
<?xml version="1.0" encoding="UTF-8" ?>
<av:answer xmlns:av="http://uni-ulm.de/bwgrid/portal/avatar"
xmns:xsi="http://www.w3.org/1999/xhtml"
xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://uni-ulm.de/bwgrid/portal/avatar/schema/answer.xsd">
  <av:followUp>
    <av:text>
```

111
I have found some answers...

<av:alternative marker="general.credential" type="COMPLEX">
  <![CDATA[How does the credential thing work?]]>
</av:alternative>

<av:alternative marker="MyProxy" type="SBS">
  <![CDATA[Show me how to create a credential!]]>
</av:alternative>

<av:alternative marker="getCert" type="SIMPLE">
  <![CDATA[Show me where I can get a certificate!]]>
</av:alternative>

</av:followUp>
</av:answer>

Listing 5.11: Servlet reply to the request of Listing 5.10

For this kind of questions, the result usually consists of a list of matching answers found in the context of the help system. Those matching answers are listed in the reply using the <alternative> tag (see Listing 5.11 lines 10, 13, and 16).

In addition to search available answers using keywords the Servlet is also able to find and deliver specific answers. The syntax of POST requests targeting specific answers is essentially the same as the syntax of Listing 5.10, except that the <text> tag from line 4 is replaced with a <marker> tag referring to the marker of an answer or step by step guide. If the marker refers to an answer as described in Section 5.3.2 the content of the file defining the answer is replied to the client without major changes.

If the marker refers to a step by step guide this step by step guide is constructed internally, evaluating required and follow up step by step guides (see Section 5.3.2) for the guide requested. The pages containing each portlet involved in the overall guide are determined (see Section 5.3.2), and the resulting guide is returned to the client as shown in Listing 5.12.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<answer xmlns="http://uni-ulm.de/bwgrid/portal/avatar"
xmli:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://uni-ulm.de/bwgrid/portal/avatar/schema/answer.xsd"
marker="gaussianNWChem">
  <sbsGuide marker='gaussianNWChem'>
    <step nav="3" subNav="1" marker="MyProxy"
      id="a9dc8e97-86d3-4544-9f3f-c1f2effc058c">
      <className>de.uniulm.bwgrid.portal.portlets.MyProxy</className>
      <gridSphereID/avatar_demo#MyProxy</gridSphereID>
      <next nextID="a02170d8-78f3-4e02-9ef1-df94665feca"/>
    </step>

    <step nav="3" subNav="1" marker="gaussianNWChem"
      id="a02170d8-78f3-4e02-9ef1-df94665feca">
      <className>de.uniulm.bwgrid.portal.portlets.GaussianNWChem</className>
      <gridSphereID/avatar_demo#GaussianNWChem</gridSphereID>
    </step>
  </sbsGuide>
</answer>
```
Listing 5.12: Servlet reply to a request concerning a step by step guide

Such response mainly consists of a sequential list of step by step guides and the position of involved protlets within the portal layout.

**Answer Ranking Algorithm**

To rank the answers provided by the help system based on the search phrase entered by users the following algorithm is used:

Let $A = (a_1, a_2, \ldots, a_n)$ be the list of answers available. The list of keyword lists (each answer could define multiple keyword lists) defined in the context of an answer is denoted as $K_i = (K_{i1}, K_{i2}, \ldots, K_{im})$ and $K = (K_{11}, K_{12}, \ldots, K_{1m}, K_{21}, K_{22}, \ldots, K_{nm})$ is a list containing all lists of keywords. $S = (\omega_1, \omega_2, \ldots, \omega_r)$ is defined to be the list of words in the search string after removing all non-alphabetical characters and converting all characters to lowercase.

1. Create a list $R = (R_{11}, R_{12}, \ldots, R_{1m}, R_{21}, R_{22}, \ldots, R_{nm})$ of matches for each keyword list in $K$ with:

   $$\omega \in R_{ij} \iff \omega \in S \land \exists \kappa \in K_{ij} : \frac{l(\omega, \kappa)}{|\omega|} < .34 \quad (5.1)$$

   Where $l(\omega, \kappa)$ denotes the Levenshtein distance [Levenshtein, 1966] between $\omega$ and $\kappa$, and $|\omega|$ the length of string $\omega$.

2. Compute the number of occurrences $\delta_t$ of each word $\omega_t$ from $S$ in all keyword lists of $K$:

   $$\delta_t = \sum_{i=1}^{n} \sum_{j=1}^{m} [\omega_t \in R_{ij}] : t = 1 \ldots r \quad (5.2)$$

   Where squared brackets are used as Iverson brackets with $[P] = \begin{cases} 1 & \text{if } P \text{ is true;} \\ 0 & \text{otherwise.} \end{cases}$

3. Calculate a match value $\mu_i$ for each answer $i$ in $A$:

   $$\mu_i = \max_{j=1 \ldots m} \sum_{t=1}^{r} \frac{[\omega_t \in R_{ij}]}{\delta_t} : i = 1 \ldots n \quad (5.3)$$

4. Sort the answers in $A$ according to $\mu_i$ and return the sorted list.

Using the Levenshtein distance for string matching in Equation 5.1 facilitates a certain fuzziness in the keyword search and allows for spelling errors. Dividing the distance by word length allows more spelling errors in longer words than in shorter words. Given the limit of .34 as stated in Equation 5.1 allows for example one spelling error for a
word consisting of three characters and three errors for a word with ten characters. Not
requiring an exact match also allows to automatically cover plurals in search strings.

Counting the number of occurrences of words from the search string that were matched
with the keyword string was used as a simple measurement to determine keyword
importance. If a user for example entered a search phrase like: “How do I submit a Gaussian
job?”, the words “How”, “submit”, “job” usually will occur in a lot of keyword lists related
to step by step guides. Thus, those words are less relevant to finding an appropriate
answer than the term “gaussian”, since this term will be in the keyword list of highly
relevant answers only. This difference of importance is reflected by setting the contribu-
tion of a found string to the overall match value in relation to the number of matches in
Equation 5.3.

Even though the algorithm is quite simple and straightforward, it works surprisingly
well. Thus, despite of plans to improve it at a later stage, by finetuning the computation
of the match value, there was no need to actually make any changes during the productive
phase of the bwGRiD portal and it was used throughout the lifetime of the project.

Step by Step Guide Selection Algorithm

If no option for a step by step guide was explicitly added to a given portlet of a page, an
algorithm was implemented to select a guide that deemed to be the most appropriate for
the given page. This algorithm acts on the assumption that the site administrator puts
portlets on pages in a meaningful way. Thus when such a generic request for a step by
step guide is received at the server, all portlets contained on the given page are searched
for available step by step guides and all possible step by step guides for this page are
examined for the number of portlets on the page that were involved following this step by
step guide. The step by step guide containing the most portlets of the page is afterwards
returned to the user.

To make this clear consider the following example, a page containing a MyProxy-
portlet, a Job Submit-portlet and a Job Monitoring-portlet. The MyProxy-portlet defines
an elementary step by step guide showing users how to get a credential. Since the Job
Monitoring-portlet is not usable without retrieving a credential first, it defines the
MyProxy-portlet to be a prerequisite for its step by step guide. The same is true for the
Job Submit-portlet thus it also defines the MyProxy-portlet as prerequisite. Additionally
the step by step guide of the Job Submit-portlet also defines the step by step guide of
the Job Monitoring-portlet as follow up.

From an administrators perspective, putting all three portlets on a single page is
reasonable, because it allows users to get a credential, submit their job and monitor it
without changing the page.

When a request for a step by step guide for this page is now received at the server,
the portlets of this page are examined and all potential paths for step by step guides are
constructed:

1. MyProxy

2. MyProxy ⇒ Job Submit ⇒ Job Monitoring
3. MyProxy ⇒ Job Monitoring

Of these possible step by step guides, the second one is returned, since this is the only one involving all three portlets of the page and thus reflecting the administrator's intentions making him put all portlets on the same page.

5.3.3. Client-Side Implementation

On the client-side the system was completely implemented using JavaScript [ECMA International, 2011] and can be used without additional plugins. For the features of the help system enabled in the productive version of the portal high compatibility standards were kept and the system worked acceptably even with older browsers at the time. Due to its strict compliance with web standards the system works without limitations in contemporary web browsers. For the features of the system that were not activated in the productive system this compatibility was not enforced. Thus, the speech recognition feature for example only works in contemporary versions of Google Chrome and Apple Safari web browsers.

The components of the system were implemented following an object oriented approach.

Model

The client side model mostly consists of the information added to the HTML pages making up the UI of the portal as described in Section 5.3.2 and the information retrieved from the server using the querying mechanism described in Section 5.3.2. Additionally the model keeps track of users’ progress in step by step tutorials and the current state of the help system. To keep this information persistent during page reloads HTML Web Storage [Hickson, 2013] was used if available and Cookies [Barth, 2011] otherwise.

View

The view component of the system consists of a set of classes that all inherit basic properties, like their position on the screen, an overlay and background image from a VisualObject class. This class serves as a layer of abstraction to the underlying HTML elements that are used to actually render the content of VisualObjects. It also unifies event handling and allows to register the event handlers of the controller component to be notified of user actions.

That way the UI of the help system is split into various classes implementing different components. There is for example a Balloon class for the speech balloon of the agent, a IQBox class to render the interactive box below the portrait of the agent, a HelpMarker class responsible to render the markers displayed above elements during step by step guides and element explanation, or a DrNick class responsible to render the visual representation of the agent. This approach allows to quickly and easily change parts of the UI of the help system without affecting other parts.
Controller

An important task of the controller component is to handle requests to the server and incorporate the responses received into the client side model. This is achieved issuing HTTP POST requests to the portal server using the XMLHttpRequest Application Programming Interface (API) [Van Kesteren et al., 2014] available in browsers. Querying information is performed synchronously and user interaction is halted until a reply is received from the server. During that time the view component of the system shows a “thinking” representation of the agent and an animation in a thought bubble that replaces the original speech balloon (see Figure 5.30).

After receiving an answer from the server (see Section 5.3.2) the view component was updated accordingly to present the received content.

5.3.4. Spoken In- and Output

As shown in Figure 5.31 a spoken dialogue system consists of several parts. According to [Minker and Bennacef, 2004] it comprises different components responsible for speech recognition, semantic analysis, dialogue management and response generation. How those components are integrated into the existing implementation of the help system will be
described shortly within the following sections.

**Speech Recognition**

Speech recognition is performed in the context of the control component of the client-side implementation utilising the HTML Web Speech API [Shires and Wennborg, 2012] that is currently – August 2014 – available in Google Chrome and Apple Safari browsers. This API allows to realise speech recognition directly from within the browser. This speech recognition is usually performed using a distributed approach as described in Schmitt et al. [2009] and speech data is transferred to a server for recognition. However, the API specification does not dictate a specific approach for speech recognition and the whole recognition process is performed transparently. Performance is usually excellent even under rather noisy conditions. The result returned by the speech recognition process consists of lists of words and utterances recognized. The speech recognition component of the help system was implemented to be continuous way. This means speech recognition was permanently active and expecting user input. Thus, there was no need to activate the speech recognition feature by pressing a button or uttering a keyword.

**Semantic Analysis – Dialogue Management**

Semantic analysis is also performed in the control component of the client-side implementation of the help system. As soon as the speech recognition component indicates that there are speech recognition results available, the list of results is searched for commands and if a result matches one of the defined commands the corresponding action is performed by the dialogue management component also part of the control component. Commands usually are the numbers associated to options presented in the interactive area below the image of the agent (see Figure 5.27).

**Text Generation**

The text generation part of the system is integrated into the view component of the system. It builds upon the textual output feature and maintains a list of the current content of the speech balloon and of the interactive box of the help system. These parts are all combined and are the basis used for the spoken output of the system.

**Speech Synthesis**

While the Web Speech API specifies an interface for SpeechSynthesis this feature was not available in browsers yet, when the help system was implemented. Hence, speech synthesis was performed at the server-side. Therefore all strings defined in the context of the help system are automatically converted to audio files, when they are changed or accessed for the first time using the Text-To-Speech (TTS) engine of the Loquendo Speech Suite 7. Links to the audio files generated are added to the HTML output of the help system specific tags described in Section 5.3.2. In the view component of the client those audio files are only played in sequence afterwards.
5. The Prototype System

5.3.5. Documentation

One focus during development of the help system was to make the extension of existing portlets as easy and transparent to portlet developers as possible. Thus, while the system offers a plethora of possibilities for customization, the default behaviour implemented into the system always makes sure that even basic support of help system functionalities from the side of a portlet could have a huge benefit to users. To a certain degree the implementation of the help system follows our general design principle of supporting users to attain first results quickly by making the addition of basic help system functionality to portlets easy and allowing users to refine their first results offering various additional features allowing for further customization. One example for this approach is the functionality of step by step guides. Portlet developers can define options that are suggested by the agent if the user asked for assistance on a page containing their portlet (see Figure 5.11). However if such a suggestion is not explicitly defined it is made sure that the default functionality of the “Guide me through all this step by step!” option (see Figure 5.8) also starts an appropriate step by step guide using an intelligent algorithm (see Section 5.3.2). Another example is the possibility to define different texts for explanation if an element is visited during a step by step guide or if a user asked for help concerning this particular element in the “Help me with an element of this page” mode. The default behaviour of the help system is to show the same explanation for the element in the speech balloon of the agent for both cases. However, if the portlet developer considered it more appropriate to provide two different texts it was possible to do this. Extensive care was taken to always implement such a reasonable default behaviour to keep the additional workload induced on the portlet developers by adding help system functionalities to a minimum, while interested and motivated developers are enabled to provide highly customized and flexible resources for assistance to users.

To allow for easy integration of the help system into portlets the documentation of the help system was integrated with the portal templates used during portlet development at partner sites. Those portal templates were more or less copies of the production portal that could be installed and run locally at portlet development sites, to serve as a test bed and foster the implementation and testing of application specific portlets implemented at each site. A portlet developed using the current version of the portal template could easily be distributed to partner sites and without further effort be integrated into the productive bwGRiD portal. The documentation for the help system was available under the “Avatar” subtab in the “Doc” section of the portal template, when the user logged into the portal as site admin or portlet developer. Figure 5.32 shows the options provided to portlet developers to learn about using the help system in their own portlets. Those options included tutorials with examples and listings (see Figure 5.33 and Figure 5.34).

To assist developers to finetune the basic functionality of the help system added to the portlets after following the tutorials, additional documentation was provided explaining all elements and attributes defined in the tag library (see Figure 5.35).

In addition to enabling portlet developers to embed the functionality of the help system into their portlets, another important aspect of documentation was to increase the reusability of components. Thus, all parts of the system are extensively documented.
5.3. Technical Realization

Figure 5.32.: The introductory page of the documentation for the help system.

Figure 5.33.: An overview of files that need to be edited to add specific help system functionality to a portlet.
5. The Prototype System

Figure 5.34.: A listing from a tutorial showing how to use the tag library of the help system to add specific features to an existing portlet.

Figure 5.35.: Excerpt from the documentation of the tag library explaining attributes of a tag along with their usage.
Figure 5.36.: Excerpt from the documentation of the client side implementation of the help system.

Figure 5.36 shows an example for the documentation of a component of the client-side javascript based implementation of the help system.

5.3.6. Conclusion

Compared to the traditional approach of providing documentation for programs by means of dedicated help pages the approach introduced in this section offers various advantages from a developer's and site administrator's perspective:

- There is no need to write extensive documentation and tutorials for user assistance.
- The help content is integrated into the UI of the portlet. Changes of the position, label or icon of an element are reflected instantly in the help system without updating the documentation.
- Changes to the layout of the portal are also reflected instantly. If a tab in the navigational area of the portal was for example renamed all occurrences of the old tab name in tutorials provided in the help section of the portal would have to be adapted to reflect this change in name. If the documentation included images those images would have to be replaced too.
5. The Prototype System

- Parts of tutorials often refer to other tutorials. Take for example a tutorial concerning the submission of \textit{Gaussian} jobs. Since users need to create a \textit{MyProxy} credential before being able to submit a \textit{Gaussian} job the page containing the tutorial for Gaussian has to either contain a link to a tutorial explaining how to create a \textit{MyProxy} credential or has to directly include the steps to create the credential. If the first option was chosen, chances are high to lose the user somewhere in the process and in case of the second approach the author of the \textit{Gaussian} tutorial has to monitor the “MyProxy” portlet for changes and has to adapt his tutorial everytime the UI of this portlet changes.

Keeping the portability of components in mind during the implementation process proved especially useful. It was for example planned to switch from the \textit{GridSphere} portal framework to \textit{Liferay} [Sezov, 2012], since the support for \textit{GridSphere} was discontinued during the bwGRiD portal project runtime. Porting the help system components to the new framework was accomplished within days. This shows that the parts of the system implemented in the scope of the project could easily be ported. Even though the help system was used for a very specific scope of application it can easily be adapted to cover different areas and it would even be possible to use the system in a general web based context for all web pages available on the internet by using for example reverse proxies.
Part III.

Evaluation and Experiments
6. General Methodology

The following chapters discuss experiments and evaluations carried out in the context of this thesis. We were especially interested in three different aspects of our system. First of all we wanted to find out how our system performs compared to a more traditional help system based on help pages. After this rather obvious experiment that focused on measuring the increased efficiency fostered by our advanced help system we wanted to evaluate different aspects influencing users' subjective acceptance of help systems. Those aspects were the influence of using an anthropomorphic agent to represent the help system and the influence of offering help to users proactively. Additionally, we wanted to evaluate the effects using spoken in- and output to interact with the help system had on users' performance, cognitive load and subjective evaluation of our help system.

In order to ensure reproducibility of results the methodology used for all experiments conducted was very similar. For each condition of the experiments only particular, separate aspects of the help system or evaluation method have been changed between conditions, keeping the general methodology consistent between experiments. This general methodology will be introduced within the next sections.

6.1. Scenario

As the system used in the productive portal (see Section 5) could not be used for the evaluation of the system due to rather high access restrictions, we decided to implement another scenario to evaluate the help system. This scenario consisted of a nuclear power plant simulation where a simplified virtual power plant had to be controlled in order to accomplish given tasks. This scenario allowed us to implement a rather complex model discouraging subjects from solving tasks without assistance. We wanted to confront participants with rather abstract, complex problems that would be hard to impossible to solve without help, thus maximizing participants’ dependency on the help system and increasing interaction time with the help system.

6.1.1. Nuclear Power Plant Simulation

The user interface was displayed in a web-browser and the upper part of the interface consisted of some generic images of happy young kids and pictures related to nuclear power plants as can be seen in Figure 6.1. Next to this images the generic capital letters “NPPL” resembling some kind of initialism and a generic version number were displayed. While we did not expect this part of the interface to get much attention from the participants due to the banner blindness effect Benway [1998], we nevertheless
6. General Methodology

Figure 6.1.: The banner and navigational area of the nuclear power plant GUI

took care that this part looked pleasant and had a reasonably professional appeal. The lower part of Figure 6.1 shows the navigational menu of the user interface. The controls of the power plant were split into different functional units that were accessible by the navigational links. Since all studies were conducted in Germany the UI is in German. A translation of the items in the navigational area along with a short description is given below.

Kontrolle | control – This section offers a “meta control” view of the power plant. As depicted in Figure 6.2 it contains several input fields to control the different parts of the plant during “normal” operation. Making changes in this part of the interface had effects on the other parts of the interface. Changing the overall reactor performance for example causes the control rods in the reactor view to be automatically set to the value desired.

Reaktor | reactor – The reactor view is shown in Figure 6.3. In this view users can manually adjust the control rods of the reactor chamber. This had to be done as soon as the system reported faulty operation of one or more control rods. In this cause the automatic control of the control view was disabled and users had to adjust the rods manually.

Primärkreislauf | primary circuit – The primary circuit view allows participants to separately control the performance of the water pump performance in the primary circuit. The UI of this view is depicted in Figure 6.4. Similar to the reactor view users only have to use these controls when one of the water pumps reports an error and the automatic controls of the control view are disabled.

Sekundärkreislauf | secondary circuit – The controls for the secondary circuit are the same as for the primary circuit.

Kühlkreislauf | cooling circuit – The cooling circuit controls is also the same as for the primary circuit.

Krisenreaktion | crisis response – The crisis response tab contained several different controls needed to protect the power plant in case of unforeseen events like a fog grenade launcher or the standby generators.
6.1.2. Tasks

Participants had to complete seven tasks using the controls provided by the interface. At the beginning of each task a description was displayed informing participants of the task’s objective. As can be seen in Figure 6.6 this description was integrated into the regular UI of the power plant in a light red box. Next to the task description this box contained a button labeled “gelesen” (read).

Before participants clicked this “read” button all controls of the power plant were disabled and replaced by a message reading “Please confirm that you have read the current task before proceeding!”. After the “read” button was clicked, the controls of the power plant became available and the actual interaction with the system commenced (see Figure 6.7). Thus, clicking this button established a well defined point of reference allowing to exactly measure participants’ interaction time with the system.

The box containing the task description was permanently displayed below the navigational area of the UI, allowing participants to verify the current objective during interaction. As can be seen in Figure 6.7, the box also featured a button labeled “Neustart” (restart) at this stage. This button allowed users to start solving the current task all over again and all controls of the power plant were reset to the state they had at the beginning of the task. Thus, in case participants had the impression they had messed something up, they were always able to undo their actions and return to a save reference point.

To add variety to the tasks participants were interrupted during three of the seven tasks with problems that required immediate intervention. The problems arising were displayed as warning messages. An example for such a warning message is shown in Figure 6.8.

The warnings featured the same light red box as the task descriptions and the same “read” button. Different to normal task descriptions they displayed a warning sign and the UI was greyed out during presentation in order to focus participants’ attention at the warning message and to convey an impression of urgency.

The objective of each task along with the arising problems was as follows:

**Login** Log in to the system!

1. Bring the power plant to work at optimal performance!

2. Due to good weather conditions regenerative energy sources are at peak performance. Hence you can reduce the power output to 12800 MW.

3. The weather conditions have worsened. The plant has to be brought back to optimal performance!

**Problem** There was a failure of a pump in the primary circuit. Out of security considerations the performance of the remaining pumps was set to 100 % and the performance of the pumps has to be adjusted manually. After doing so, you should file a repair order for the broken pump.
6. General Methodology

Figure 6.2.: The section of the GUI labeled “Kontrolle”

Figure 6.3.: The section of the GUI labeled “Reaktor”
6.1. Scenario

Figure 6.4.: The section of the GUI labeled “Primärkreislauf”

Figure 6.5.: The section of the GUI labeled “Krisenreaktion”
6. General Methodology

4 There is a football match and the overall need for power increases. Despite of the broken water pump you have to increase the performance of the power plant above the 100% threshold to 1920 MW.

5 The game is over and you can reduce the performance of the plant to the optimal level again. In the meantime the water pump in the primary circuit has been fixed. Thus you do not have to control the primary circuit manually any more.

5 Problem Due to the excessive heat generated running the plant above optimal performance, three control rods in the reactor chamber have failed. To avoid critical conditions the positions of the control rods has to be adjusted manually.

6 Power consumption after the football match is surprisingly low. You can reduce the performance to 1280 MW.

6 Problem There is an unexpected external thread. Take all precautions to prepare the power plant for a potential terrorist attack. An unidentified flying object entered restricted airspace surrounding the plant.

7 The plant is shut down now. This is the perfect situation to repair the failed control rods. Thus you should take care of that now.

Logout Your shift is over and you have done a good job by sparing humankind from a terrible nuclear disaster one more day. So please log out of the system and fill out the final questionnaire. If you are having trouble finding the right questionnaire please refer to the supervisor.
Solving the tasks required participants to provide different input to various controls of the power plant. It was virtually impossible to solve the tasks without assistance. However, some aspects of tasks were designed to be repetitive. For example the first part of task 3 required participants to provide the same input as for task 1. Thus if a participant remembered the correct input values from task 1 he could solve the first part of task 3 without assistance. This property was added to the scenario to verify how likely participants were to use the help system in this kind of situation.

6.1.3. Assistance

In order to find out about the input fields that needed to be manipulated and the correct values to be entered participants had to use different help systems dependent on condition. Independent of the condition the help systems all provided the same information. There was an overview of the power plant and the different functional units of the plant were introduced and the relationship between the units was explained (see Figure 6.9).

Additionally participants were able to find out how the correct input values for different states of the power plant were computed (see Figure 6.10).

Using this information it would have been possible to manually solve all of the given tasks, except the one involving the terrorist thread, without further assistance. However, due to the non trivial nature of the model constituting the power plant simulation a manual approach was rather hard to impossible to work out.

The more obvious option to solve a given task however, was to use one of the step by step guides that were also available. Those step by step guides indicated all input fields.

Figure 6.7.: The UI after clicking the “read” button.
6. General Methodology

![Image](image.png)

Figure 6.8.: The UI after clicking the “read” button.

that needed to be edited and provided participants with the correct values to enter (see Figure 6.11).

When the help system was activated during the experiments the first option offered (see Figure 5.8) was always the correct step by step guide for the current task. This way the help content suggested by the system was always highly relevant for task completion.

6.2. Participants

Participants for our studies where mostly students recruited randomly at the hallway in front of the canteen or at the cafeteria at the University of Ulm. An attempt to recruit participants using flyers was not very successful and abandoned after the first attempt. Thus, the vast amount of participants where approached by one of the conductors and asked to participate in a face to face situation. Using a lottery for cinema tickets (1:4 probability) and a chocolate bar as incentive to participate was also not very efficient. So the incentive was changed to 5 € in cash after the first recruiting cycle.

Overall 203 subjects participated in the experiments performed in the context of this thesis. We considered participants not finishing at least 4 of the 7 tasks not to be able to reliably rate the performance of the different help systems. Thus 13 participants were excluded from the final analysis, leaving 190 subjects. The reasons why certain subjects decided to abort interaction could not have been further investigated. The most probable case might have been that those subjects were simply running out of time. Even though supervisors clearly stated that the experiment usually takes about 45 minutes to finish,
6.3. Procedure

Figure 6.9.: The agent explaining the different functional units of the plant.

Those participants might have been hoping to be quicker.

Since the subjects were mostly students the average age was quite low with 22.0 years ($SD = 3.47$). 93 of the subjects were female and 95 male with two subjects not specifying their gender.

6.3. Procedure

All experiments were performed in typical seminar rooms for 20 to 30 people. Usually six laptops were set up and six subjects could participate in parallel. The number of subjects working in parallel differed. Before interaction with the help system took place participants filled out some questionnaires (see Section 6.4.1). After completing these initial questionnaires users interacted with the respective help system. When subjects had finished the interaction process with the system, they filled out the evaluation questionnaires (see Section 6.4.2). The overall time needed for the whole procedure was usually about 45 minutes.
6. General Methodology

Figure 6.10.: The agent showing how the performance of the power plant is computed.

6.4. Questionnaires

All questionnaires used throughout the experiments were online forms implemented with LimeSurvey [Schmitz, 2012]. The only exceptions are the pen and paper and agent conditions in Chapter 12. The modes used for the questionnaires in this experiment will be illustrated in the explanation of the corresponding experiment.

The following subsections will give an overview of the questionnaires used before and after interaction and will explain the ideas we had integrating particular questionnaires into the test procedure.

6.4.1. Before Interaction

The objective of the questions asked before interaction was to cover some personal data and personality traits. This data was collected to check for possible imbalances between groups and to find influences of personality traits on the rating of system and participants performance.

Personal Data

The first group of questions assessed some general demographic data like age, gender, highest school-leaving qualification acquired, and current profession.
Big Five

The idea behind the Big Five personality traits stems from the idea that the most important personality characteristics are reflected in natural language [Ashton and Lee, 2005]. This means, for important characteristics describing differences between individuals, there is supposed to be a word describing that one person has this specific property. Take for example the word “proud”. This word describing a certain property of an individual is important enough to designate a word in natural language to describe it. Since this should be true for all important characteristics a human can have, there should be a word – at least for all the important ones – in natural language. Acting on this assumption Allport and Odbert [1936] for example compiled a list of 17,953 single word terms describing personal properties [Block, 1995]. Building on this work five relative stable factors could be identified describing the structure of this personality lexicon [Goldberg, 1992]. Those five personality dimensions, along with some associated traits are [Barrick and Mount, 1991]:

**Extraversion:** sociable, gregarious, assertive, talkative, and active

**Neuroticism:** anxious, depressed, angry, embarrassed, emotional, worried, and insecure
6. General Methodology

**Agreeableness**: courteous, flexible, trusting, good-natured, cooperative, forgiving, soft-hearted, and tolerant

**Conscientiousness**: careful, thorough, responsible, organized, and planful

**Openness**: imaginative, cultured, curious, original, broad-minded, intelligent, and artistically sensitive

Those dimensions are usually called the “Big Five”. Based on this Big Five model of personality structure Costa and McCrae [1992] developed a questionnaire containing 60 items allowing to estimate subject characteristics for those five given scales. A German version of this questionnaire was introduced by Borkenau and Ostendorf [1993]. While the original version of this questionnaire consists of 12 items per scale, we decided to reduce the number of items per scale to 9, in order to reduce overall questionnaire length.

The reliability for the five scales attained with the shortened version of the questionnaire are summarised in table 6.1. Those values are comparable to values attained with other condensed versions of NEO-FFI, for example Köner et al. [2008]. Concerning the equivalence of using the NEO-FFI in an online version compared to a pen and paper based version Rammstedt et al. [2004] showed that results attained with either mode are quite comparable and there seem not to be systematic differences.

<table>
<thead>
<tr>
<th>scale</th>
<th>sample item</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>extraversion</td>
<td>“I like to have a lot of people around me”</td>
<td>.77</td>
</tr>
<tr>
<td>neuroticism</td>
<td>“I often feel inferior to others”</td>
<td>.78</td>
</tr>
<tr>
<td>agreeableness</td>
<td>“I try to be courteous to everyone I meet”</td>
<td>.73</td>
</tr>
<tr>
<td>conscientiousness</td>
<td>“I keep my belongings clean and neat”</td>
<td>.79</td>
</tr>
<tr>
<td>openness</td>
<td>“I often try new and foreign food”</td>
<td>.63</td>
</tr>
</tbody>
</table>

Table 6.1.: Cronbach’s alpha for the condensed version of the NEO-FFI scales

**Self-Efficacy**

Another aspect of personality we considered to possibly have an influence on subjects’ rating of different treatments, especially for the system evaluated, was self-efficacy. A common definition of self-efficacy is that it “refers to one’s belief in one’s capability to perform a specific task” [Gist, 1987]. This quite common task specific definition stems for example from Bandura [1977, 1982]. Compared to this task specific notion of self-efficacy the questionnaire from Schwarzer and Jerusalem [1999] aggregates over a diversity of expected self-efficacy over various domains [Schwarzer and Jerusalem, 1999]. This means it is explicitly not confined to measure self-efficacy for a particular task, but to assess characteristics of a subjects perceived ability to deal with problems in general.

The questionnaire consists of 10 statements, like “Whatever happens, I will be alright” or “I can find a solution for every problem” and subjects can agree or disagree to the statements on a 5 point Likert scale. The achieved internal consistency of the 10 items
was $\alpha = 80$. This is similar to the internal consistency reported by Schwarzer and Jerusalem [1999].

### 6.4.2. After Interaction

The questionnaires after interaction with the help system evaluated different aspects of the interaction process and users’ subjective perception of the help system.

#### Cognitive Load

For some experiments we evaluated the cognitive (see Section 2.1.2) load on users during task performance. To measure cognitive load we used the mental effort rating scale by Paas [1992]. Since this scale does not differentiate between intrinsic, extraneous, and germane cognitive load, we also included an experimental questionnaire by Klepsch et al. [2014]. This questionnaire consists of 12 questions aiming to assess different types of cognitive load. There are four questions for intrinsic cognitive load, four questions concerning extraneous cognitive load, and four questions for germane cognitive load.

Pre-testing showed that three questions concerning extraneous cognitive load were not applicable for our evaluation. Two of these questions explicitly mentioned the words “learn-” or “learning” and since the tasks in the experiments were not about learning and users were neither encouraged nor especially interested in “learning” how to solve the given problems on their own, participants of the pre-tests expressed confusion and found it hard to come up with reasonable answers for this questions. The same was true for a third question on germane cognitive load, asking how hard it was “to integrate central ideas of the task into a single mental representation”. Since the concept behind the test system never was to help users create a mental representation of the system, in order to keep participants dependent on help by the agent, this question did not make sense in the given context as well. Hence, pre-testers also stated, that they were not aware that they were supposed to gather in depth knowledge of the system and they just wanted to finish the tasks. To spare participants from possibly arising confusion, we decided to remove the given questions from our final questionnaire, even if it ment that there was only one item measuring extraneous cognitive load remaining. However, since using the help systems was the main determinant for using the system we concluded that the cognitive load implied on users by the help system to be measured by the intrinsic cognitive load scale making a distinction between extraneous and intrinsic cognitive load questionable in the context of our experiments. Nevertheless, the remaining item targeting extraneous cognitive load was kept in the questionnaire.

As already mentioned the questionnaire by Klepsch et al. [2014] used is still experimental and data supporting its ability to differentiate between different manifestations of cognitive load has not been reported yet. That is why we performed a confirmatory factor analysis to determine to which extent the items can be attributed to the three different factors. At this stage we found that a question assessing germane cognitive load by asking users how much fun they had during task completion hardly correlated with the other items in this group and we removed the item. Another question asking partic-
6. General Methodology

Participants how hard it was to solve the given tasks also did not contribute to the intrinsic scale and was removed accordingly. It might be interesting to note that these questions removed did not use the standard 7 point Likert scales labeled “absolutely not correct” and “absolutely correct” that were used for the other items, but had endpoints labeled “very little”, “very much”, and “very easy”, “very hard” respectively.

The results of the analysis for the remaining 7 items and 42 cases that filled out the questionnaire are summarised in Table 6.2. It can clearly be seen, that the items best load on the factors they were designed to evaluate and clarify 82.6% of the overall variance. The internal consistency for the scales measured by multiple items and all items combined is summarised in Table 6.3.

<table>
<thead>
<tr>
<th>item</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>During the tasks you had to keep a lot of stuff in your head at once</td>
<td>.92</td>
</tr>
<tr>
<td>The tasks were very complex</td>
<td>.91</td>
</tr>
<tr>
<td>The tasks were strenuous</td>
<td>.81</td>
</tr>
<tr>
<td>I made an effort to not only remember certain bits but to comprehend the whole of the subject</td>
<td>.86</td>
</tr>
<tr>
<td>Working on the tasks it was important to me to comprehend everything correctly</td>
<td>.82</td>
</tr>
<tr>
<td>I made an effort solving the tasks</td>
<td>.79</td>
</tr>
<tr>
<td>It is tedious to find the most important bits of information solving the tasks</td>
<td>.91</td>
</tr>
</tbody>
</table>

Table 6.2.: Varimax rotated component matrix for a factor analysis of the items to measure differentiated manifestations of cognitive load. (Coefficients < .4 omitted for clarity, unofficial translation of items from German to English by the author)

<table>
<thead>
<tr>
<th>scale</th>
<th>#</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>intrinsic</td>
<td>3</td>
<td>.90</td>
</tr>
<tr>
<td>germane</td>
<td>3</td>
<td>.79</td>
</tr>
<tr>
<td>all</td>
<td>7</td>
<td>.79</td>
</tr>
</tbody>
</table>

Table 6.3.: Cronbach’s alpha for the scales with several remaining items for the differentiated cognitive load questionnaire by Klepsch et al. [2014]

Additionally, we decided to combine the items from the mental effort scale by Paas [1992] and the differentiated scales from Klepsch et al. [2014] into a combined cognitive load scale. This decision was influenced by the concerns in discriminability of the three different aspects of cognitive load for the given experiments. The internal consistency for the final overall cognitive load scale consisting of 10 items was \( \alpha = .84 \).
6.4. Questionnaires

**Own Questionnaire**

As one of the main focuses of our research was on the acceptance of the help system, we developed a questionnaire assessing subjects' personal stance towards the agent and the help system in general. This questionnaire consisted of nine scales. These scales along with the number of items used for each scale, a sample item and achieved internal consistency for each scale are summarised in Table 6.4.

<table>
<thead>
<tr>
<th>Scale</th>
<th>#</th>
<th>Sample Item</th>
<th>(\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>help behaviour</td>
<td>4</td>
<td>The avatar is doing a good job helping me</td>
<td>.8</td>
</tr>
<tr>
<td>intrusiveness</td>
<td>3</td>
<td>The avatar is intrusive</td>
<td>.71</td>
</tr>
<tr>
<td>distraction</td>
<td>4</td>
<td>The avatar is continually attracting my attention</td>
<td>.87</td>
</tr>
<tr>
<td>perceived dominance</td>
<td>6</td>
<td>The avatar is arrogant and snotty</td>
<td>.62</td>
</tr>
<tr>
<td>optimism</td>
<td>3</td>
<td>The avatar appears to be a good tempered optimist</td>
<td>.8</td>
</tr>
<tr>
<td>tolerance</td>
<td>4</td>
<td>The avatar is friendly</td>
<td>.72</td>
</tr>
<tr>
<td>humanity</td>
<td>8</td>
<td>The avatar seems to be vivid</td>
<td>.78</td>
</tr>
<tr>
<td>competence</td>
<td>9</td>
<td>I can depend on the avatar</td>
<td>.79</td>
</tr>
</tbody>
</table>

Table 6.4.: Scales, number of items, sample item and Cronbach’s alpha for our own questionnaire.

The first three scales — *help behaviour*, *intrusiveness*, *distraction* — were assessing information about how the agents’ actions were rated. The *help behaviour* scale was straightforwardly evaluating how participants perceived the usefulness of the agent to be. The *intrusiveness* scale assessed if participants considered the help system to be intrusive and the *distraction* scale was evaluating if users perceived the help system to distract them from their tasks.

The second set of scales — *optimism*, *tolerance*, *humanity*, and *dominance* evaluated how users rate the agents’ “personality”. For these scales we used the German version of the NEO-FFI [Borkenau and Ostendorf, 1993] as a source of inspiration.

**AttrakDiff**

In addition to our own questionnaire we used the *AttrakDiff* questionnaire developed by Hassenzahl et al. [2003]. This questionnaire consists of 28 items belonging to four different scales with 7 items for each scale. Each item contains a bipolar verbal anchor like “technical – human” or “impractical–practical”. Subjects have to decide on a 7-point Likert scale which adjective better describes the help system. The four scales of AttrakDiff are:

**perceived pragmatic quality** (Cronbach’s \(\alpha = .72\)) This scale measures the potential to support a users’ need to achieve behavioural goals. A sample bipolar verbal anchor is “confusing – clear”.

139
6. General Methodology

hedonic quality – stimulation (Cronbach’s $\alpha = .8$) Through the HQS scale perceived novelty and potential to grab attention is measured. A sample item says “standard – creative”.

hedonic quality – identity (Cronbach’s $\alpha = .75$) A sample bipolar verbal anchor is “cheap – valuable”. This scale assesses potential of identification with the product, which means in our study identification with the agent.

perceived attractiveness (Cronbach’s $\alpha = .83$) This scale assesses the perceived potential of attractiveness. A sample item is “ugly – beautiful”.

CSUQ

The Computer Usability Satisfaction Questionnaire (CSUQ) that was used for some of our experiments was developed by Lewis [1995]. It is a rather straightforward questionnaire focusing on assessing perceived usability. This makes it a nice supplement to the AttrakDiff questionnaire that apart from the pragmatic quality scale rather stresses quite abstract aspects concerning hedonic qualities and perceived attractiveness of the evaluated system. The CSUQ questionnaire consists of 19 items contributing to 4 scales. Scales, number of items used for each scale, an example item, and resulting Cronbach’s $\alpha$ for each scale are summarised in Table 6.5.

<table>
<thead>
<tr>
<th>scale</th>
<th>#</th>
<th>sample item</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysuse</td>
<td>8</td>
<td>I am able to complete my work quickly using the system</td>
<td>.93</td>
</tr>
<tr>
<td>infoqual</td>
<td>7</td>
<td>It is easy to find the information I need</td>
<td>.89</td>
</tr>
<tr>
<td>interqual</td>
<td>3</td>
<td>I like using the interface of the system</td>
<td>.87</td>
</tr>
<tr>
<td>overall</td>
<td>19</td>
<td>Overall, I am satisfied with this system</td>
<td>.95</td>
</tr>
</tbody>
</table>

Table 6.5: Scales, number of items per scale, sample item and Cronbach’s alpha for CSUQ

SASSI

Some of the experiments involved spoken in- and output. To evaluate the performance of this system parts of the Subjective Assessment of Speech System Interfaces (SASSI) questionnaire by Hone and Graham [2000] were used. The original version of this questionnaire comprises 6 scales:

Response Accuracy contains items like “The system is accurate” and relates to whether the system interprets user input correctly and the expected result is attained.

Likeability includes statements of opinion concerning the system along with items expressing feelings, like “the system is friendly”, or “It is easy to learn to use the system”.

Cognitive Demand measures the mental workload and stress inflicted by the system. An example item for this factor is: “A high level of concentration is required when using the system.”
Annoyance contains questions like “The interaction with the system is repetitive” and assesses how annoying users perceive interaction with the system to be.

Habitability consists of items like “I always knew what to say to the system” and relates to the concept of “visibility” by Norman [2002]. This concept states that possible actions, the results of those actions and the state of a system have to be visible in the UI of the system. Since visibility is not a suitable term for systems with spoken in- or output the factor was renamed to “habitability”.

Speed this scale assesses the perceived speed of interaction with the system. The two items of this scale are “The interaction with the system is fast” and “The system responds to slowly”.

We picked three of these scales to be included into our questionnaire. Namely Response Accuracy, Habitability, and Speed. Likeability was excluded since our own questionnaire (Section 6.4.2) and the AttrakDiff questionnaire (Section 6.4.2) already covered this aspect.

The Cognitive Demand part of the questionnaire was also excluded due to its overlapping with the cognitive load questionnaire (Section 6.4.2) used.

Cronbach’s alpha for the scales used in the questionnaires, and the number of items contributing to each scale are summarised in Table 6.6.

<table>
<thead>
<tr>
<th>scale</th>
<th>#</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>response accuracy</td>
<td>9</td>
<td>.68</td>
</tr>
<tr>
<td>habitability</td>
<td>4</td>
<td>.72</td>
</tr>
<tr>
<td>speed</td>
<td>2</td>
<td>.64</td>
</tr>
</tbody>
</table>

Table 6.6.: Scales, number of items per scale, and Cronbach’s alpha for SASSI

6.4.3. Questionnaire Length

Despite of some precautions, like for example shortening the NEO-Five Factor Inventory (NEO-FFI) questionnaire and excluding parts of the SASSI questionnaire, we were a bit afraid our questionnaires might be a bit too long and participants might get fatigued Herzog and Bachman [1981] answering all the questions. Thus, we decided to add a question concerning the overall length of the questionnaires as one of the last items for a questionnaire of typical length in this study. This item was straightforwardly asking participants if they considered the questionnaires to be too extensive and subjects could agree or disagree to this on a 7 point Likert scale.

For the 63 participants asked the median was at 5 and the average at 4.56 with neutral being at 4. This shows that our concerns were not unsubstantiated and that our questionnaire was perceived to be a little bit oversized, but for most participants still in a reasonable not frustrating range. Hence, we do not consider fatigue to have a systematic effect on the results of our experiments.
6. General Methodology

6.5. Objective Data

Two gather objective data on participants’ performance the screen content was captured and recorded during interaction. Additionally, participants’ interaction with the system was logged on the server using timestamps and text based descriptions of performed actions. A typical excerpt of such a log is displayed in Listing 6.1. Line 1 indicates that this participant has finished reading the task description for task 1 at timestamp 1355408006991 (13. Dec 2012 15:13:26), made some input (Line 2), reset the task (Line 3), finished rereading the task (line 4), made some other input (lines 5–7), clicked the “Assist me”-button of the agent, and selected a step by step guide (lines 8 and 9), changed his input once more (lines 10 and 11), clicked the “submit” button of the form (line 12), and line 13 indicates that the participant finished the first task at timestamp 1355408398741 (13. Dec 2012 15:19:58).

<table>
<thead>
<tr>
<th>Time</th>
<th>Description</th>
<th>Input Fields</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1355408006991</td>
<td>Description read: ONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1355408046513</td>
<td>Meta Reaktorleistung input</td>
<td>inputLeistungReaktor: 100</td>
<td></td>
</tr>
<tr>
<td>1355408123118</td>
<td>Task reset: ONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1355408128288</td>
<td>Description read: ONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1355408190427</td>
<td>Meta Reaktorleistung input</td>
<td>inputLeistungReaktor: 70</td>
<td></td>
</tr>
<tr>
<td>1355408268069</td>
<td>Meta Reaktorleistung input</td>
<td>inputLeistungReaktor: 60</td>
<td></td>
</tr>
<tr>
<td>1355408281454</td>
<td>Meta Reaktorleistung input</td>
<td>inputLeistungReaktor: 100</td>
<td></td>
</tr>
<tr>
<td>1355408316108</td>
<td>Avatar: Hilfe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1355408322035</td>
<td>Avatar: Zeig mir wie ich das Kraftwerk auf optimale</td>
<td>Leistung bringe!</td>
<td></td>
</tr>
<tr>
<td>1355408369673</td>
<td>Meta Primæerkreislauf input</td>
<td>inputLeistungPrimæerpumpen: 60</td>
<td></td>
</tr>
<tr>
<td>1355408383932</td>
<td>Meta Sekundaerkreislauf input</td>
<td>inputLeistungSekundaerpumpen: 62</td>
<td></td>
</tr>
<tr>
<td>1355408398563</td>
<td>Meta submit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1355408398741</td>
<td>ONE complete</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Listing 6.1: An excerpt from an interaction log

Using this data we were able to infer different statistics. First of all we calculated the time participants needed to complete every task. This interaction time was defined as the time participants were actually working on solving the task. This means it started after participants clicked the “read” button of the current task and stopped when the task was completed. In case participants restarted a task the time between hitting the “reset” button and the participant confirming having read the task again (see Listing 6.1, lines 3 and 4) was subtracted from the time. That way we made sure to exclusively measure the interaction time.

Secondly we could compute the number of times a participant provided erroneous input by comparing the input fields edited and values entered to a list containing the input fields that actually needed to be edited and the correct values for those input fields. For the example of Listing 6.1 the number of wrong edits would be two, since the input for inputLeistungReaktor on lines 5 and 6 were wrong and the remaining edits were correct. We wanted to use this number as indicator on how much trouble participants experienced during task completion. However, using the plain number of wrong input values provided did not directly serve this purpose. Some participants tried to solve
one of the tasks using an trial and error strategy. This means they were incrementally adapting parameters till the objective was reached. Two subjects for example needed over 100 attempts to complete a task using this method. Since those outliers tremendously skewed the average number of wrong inputs for certain tasks we decided to exclude values for number of wrong input from computing the average number of wrong input if the number of wrong inputs was greater than the number of correct inputs needed to complete the task. So for example if one subject made 10 erroneous inputs for Task 1 and would have been able to solve the task with 4 correct input actions, this subject’s 10 false inputs were not considered calculating the mean of wrong inputs for Task 1. The rationale behind this is that a participant providing more wrong input than correct input to solve the task this participant must have tried to figure out the correct input values by himself and thus – at least at the beginning – must have tried to solve the task without assistance, using a trial and error approach. That way the number of wrong inputs was split into two scales. The first one consisting of the number of wrong inputs made by a user, excluding the number of wrong inputs made by this user when trying to manually solve tasks (≠ wrong input) and a binary scale indicating if the participant was trying to solve a given task by means of trial and error or not (≠ trial and error attempts). Thus the correct ≠ wrong input scale indicates the number of errors made by the participant by mistyping or by misreading or misinterpreting instructions, while the ≠ trial and error attempts summarises participants’ attempts to solve tasks manually.

To compute the overall number of wrong inputs per subject the sum of all wrong inputs for all tasks made by each participant was calculated. Again if all values were included the data would have been tremendously skewed by outliers. Thus we decided to exclude the values identified to originate from trial an error attempts of each subject on a per task basis. So for example if one subject made an attempt to manually solve Task 2 and Task 4 and made 10 and 40 erroneous inputs for those two tasks, those numbers were not considered calculating the sum. However, to compute a meaningful sum there could not be any missing values. Take for example a case with three trial and error attempts. Obviously this subject would make relatively less wrong inputs in the remaining tasks than a subject without any try and error attempt at all. To compensate for this, excluded values were replaced by the mean of wrong inputs made by the remaining subjects in each condition. The idea behind this was that a participant not trying to manually solve a task would make roughly the same amount of erroneous inputs as other subjects in this condition.

Another value yielded from the log files was the number of times participants used the “reset” button to reset the UI to the state it had at the beginning of the task. Since participants used this button in case they had the feeling of having messed something up, the number of times this button was used is an indicator on how confident participants felt solving the tasks.

Depending on condition, there were other objective values that could be calculated using the log files. Those measures, will be introduced in the Methodology sections of the respective experiments.

During some of the experiments the logging mechanism described above was not implemented yet. To gather objective performance data for this experiments nevertheless,
the screencasts recorded during interaction were annotated using ANVIL [Kipp, 2001].

As this video analysis was performed for the first experiments it was quite extensive and included further parameters like if the subject was visiting only pages of the UI relevant for task completion and current state of the agent. Figure 6.12 shows the ANVIL workbench during annotation and gives an impression of the amount of annotations applied. Overall more than 30 hours of video were tagged and more than 10,000 annotations were made.

![Figure 6.12: The ANVIL workbench during annotation.](image)

The results attained with this extensive analysis were used to exploratively identify fundamental parameters characterising the interaction process and helped to decide on the parameters that were later considered during the implementation of the logging mechanism.

The data obtained with both methods - video and log analysis - was practically equivalent. Video annotation was done on a frame by frame basis and videos were recorded at a frame rate of 15 fps. Thus the timestamps computed were exact to about the tenth of second. Taking the delay of the logging mechanism into account the same is true for the log based data. We also discovered a bug in the frame by frame export feature of ANVIL that caused the omission of certain tags in the frame by frame exported data presumably due to a mismatch caused by a rounding error, when tags were placed at certain points of the timeline. Hence this feature was rewritten from scratch and a bug report was e-mailed to the ANVIL contact address.
6.6. Statistical Testing

To test for significant differences between Likert-scaled items Wilcoxon-Mann-Whitney-tests [Wilcoxon, 1945; Mann and Whitney, 1947; IBM Corp., 2012] – in case of comparing two responses – and Kruskal-Wallis-tests [Kruskal and Wallis, 1952; IBM Corp., 2012] – in case of multiple responses – have been used. We decided to prefer a non-parametric test over a parametric one, since testing for normal distribution using a Shapiro-Wilk-test [Shapiro and Wilk, 1965; IBM Corp., 2012] indicated that some items might not follow a normal distribution [Knapp, 1990; IBM Corp., 2012]. For scales computed from several different sub-scales t-tests and ANOVAS Hatzinger and Nagel [2009]; IBM Corp. [2012] have been performed respectively. For post-hoc testing Scheffe’s method was used to compensate for alpha error cumulation in case a Levene Test did not suggest variance inhomogeneity and Bonferroni correction otherwise IBM Corp. [2012].

To validate the reliability of scales Cronbach’s alpha [Cronbach, 1951; IBM Corp., 2012] was used.

6.7. Influence of Personality Traits

We expected personality traits to have an influence on participants’ performance with, and subjective rating of the system dependent on condition. To verify that our groups were balanced with respect to the personality traits we checked our groups for differences personality trait characteristics. For none of the personality traits evaluated, extraversion ($F(8, 178) = .55, p = .82$), neuroticism ($F(8, 178) = .71, p = .68$), agreeableness ($F(8, 178) = .47, p = .88$), conscientiousness ($F(8, 178) = 1.56, p = .05$), openness ($F(8, 178) = .63, p = .75$), and self-efficacy ($F(8, 178) = .47, p = .88$) there was a statistically significant difference between the groups.

To check for possible correlations between personality traits and objective and subjective scales used to evaluate the system Pearson’s $r$ [Pearson, 1895] was calculated. Results for this correlations are reported along with the respective experiment.
7. Advanced Help System vs. the Traditional Approach

In this study we compared a typical hypertext based help system to our system. Hence, there were two interaction conditions. The HTML condition provided a typical text based help system to solve the assigned tasks and in the interactive condition participants were assisted with the interactive help system. The exact methodology of this experiment will be explained in the following section.

7.1. Method

The methodology used for this experiment was consistent with the general methodology described in Chapter 6. Accordingly, the following sections will only provide information on the characteristics of this specific experiment.

7.1.1. Conditions

The help system in the HTML condition consisted of multiple HTML-pages. Those pages could be accessed by a link labeled “Hilfe” (“Help”) in the navigational area at the top of the power plant UI right below the banner (see Figure 7.1). This part covered the same information as the part of the interactive system explaining the different functional units of the plant and the relationship between the units.

The second page of the help content in this condition was labeled “FAQ” (see Figure 7.2). This page gave the same information as the interactive system with respect to the computation of correct input values (see Figure 6.10) and step by step guides to solve the given tasks (see Figure 6.11).

The help on this page was organized as follows:

**Steuerung im Normalbetrieb** | Control during normal operation – This part contained the step by step guides needed to solve the tasks. Clicking one of the options provided in this list expanded a tutorial telling participants which elements to edit and the correct input values to complete the given objective (see Figure 7.3). The sequence of links corresponded to the sequence of tasks. This means to solve the first task, the link had to be clicked.

**Berechnungsformeln** | Calculation formulas – This section covered the formulas behind the model of the nuclear power plant. This information would have been needed to manually solve the tasks without using the tutorials.
7. Advanced Help System vs. the Traditional Approach

Figure 7.1.: The first page of the help provided in the HTML condition.

Figure 7.2.: The first page of the help provided in the HTML condition.
7.1. Method

Steuerung bei Problemen und Warnungen | Control in case of problems and warnings
- This part offered tutorials needed to solve the problems the participants’ were facing during interaction.

Reperaturaufträge | Repair orders - Clicking on one of the options below this header opened a template for a repair order that could be copied and pasted into the corresponding text area to issue a repair order for broken parts of the power plant.

The help system in the interactive condition was the help system described in Section 6.1.3. However, in order to prevent a possible negative bias towards the anthropomorphic agent from having a negative influence of participants’ subjective rating of the help system, we decided to replace the agent with a generic question mark for this experiment (see Figure 7.4).

7.1.2. Questionnaires and Objective Data

For this study the questionnaires were slightly adapted. The term “avatar” was replaced with the term “system” in questions to refer to the help system. Questions not appropriate in the context of a noninteractive help system like “The system offers help at the right time” were discarded. Thus for the self-developed questionnaires only three reduced scales for intrusiveness with one question, help behaviour with three questions, and competence with three questions remained. The items missing in the HTML condition were also excluded computing the scales for the interactive condition for this experiment.

To assess the perceived level of annoyance participants experienced because of the permanent switch between help text and UI we added an annoyance scale to the HTML condition (Cronbach’s $\alpha=.77$). This scale consisted of two questions “The permanent
7. Advanced Help System vs. the Traditional Approach

![Image of Windows](image.png)

Figure 7.5.: “Snapping” windows containing the UI and help pages to occupy half of the screen to have both available at the same time.

switch between help text and actual task editing was time consuming” and “The permanent switch between help text and actual task editing was annoying”. Those statements were rated on a seven point Likert scale expressing disagreement and agreement with the statement.

To evaluate if participants perceived the system used in the interactive condition to be advantageous to more conventional help systems three questions were added to the after interaction questionnaire for this group. The questions were “Compared to help system on separate pages the given help system is advantageous”, “I saved time using the help system”, and “The help system used is more expedient than other help systems”. Those items were summarised to an advantageous scale (Cronbach’s α = .84).

To evaluate the strategies employed to access the help pages and the UI of the power plant in parallel we also logged the number of browser tabs opened and switches between active tabs in the HTML condition. Given the nature of the tasks and tutorials, we expected opening a tutorial in one tab and the UI of the power plant in another tab to be a typical strategy to solving the tasks. Additionally the system installed on the laptops for this experiment was also capable of arranging windows side by side similar to the “Snap” feature in Microsoft Windows 7 (see Figure 7.5).

For us this seemed to be the most advantageous approach in this condition, since it allowed participants to avoid constant switching between tabs or different pages of the UI. As the “Snap” feature of Windows 7 that surely was the predominant OS at the time the study was conducted and thus the OS most participants were accustomed to, seems to encourage this practice, we expected most participants to choose this strategy.
7.2. Hypotheses

We also expected participants in the HTML condition to experience some problems solving the tasks. Thus we hypothesized that there might be a negative effect on participants’ self reported rating of self-efficacy, after interaction with the system took place. Hence, the self-efficacy questionnaire (see Section 6.4.1) was repeated after interaction in this condition. To verify that a potential decrease in self-efficacy was dependent on condition and did not occur in general after interacting with the system, the self-efficacy was also re-evaluated after interaction with the interactive system.

Since this study did not involve spoken in- or output the SASSI-questionnaire (see Section 6.4.2) was not part of the evaluation.

Apart from the differences stated above the questionnaires used for this experiment were the ones described in Section 6.4.

7.1.3. Participants

The HTML condition consisted of 23 subjects. Three subjects were excluded from final analysis as they did not finish the minimum of 4 tasks. The mean age of subjects in this group was 21 years ($SD = 2.7$) and 12 subjects were male and 8 female. While subjects not completing enough tasks were excluded from final analysis they were not excluded from strategy analysis, since we considered the strategy chosen to solve the tasks to be independent of the number of tasks participants completed successfully. This means for example a participant using a browser tab to keep the help page open would do so quite early in the process. We considered this to be the case since we expected users to have developed a certain strategy for the quite common problem of switching attention between a tutorial and the UI of a program and would apply this learned strategy as soon as they were facing this particular problem.

In the interactive condition there were 11 male and 10 female subjects yielding 21 subjects with a mean age of 22.2 years ($SD = 4.41$) overall. Initially this group also consisted of 23 subjects, with 2 subjects excluded due to not completing 4 tasks.

7.2. Hypotheses

Due to the higher extraneous cognitive load and processing during task completion in the HTML condition compared to the interactive condition we expected participants’ performance to be better in the interactive than in the HTML condition. This means we were expecting the participants to be faster in completing the tasks and to enter less erroneous input in the interactive than in the HTML condition. Concerning trial and error attempts and resets, we did not expect any differences between conditions, since we considered those scales not to be dependent on cognitive load, but on participants’ personal properties. Thus we expected personality traits to correlate with objective measures and we expected this correlations to be different between conditions.

We also expected the interactive system to be subjectively preferred by participants over the help pages in the HTML condition. Thus we expected the subjective ratings in the questionnaires to be better in the interactive than in the HTML condition.
7. Advanced Help System vs. the Traditional Approach

7.3. Results

7.3.1. Strategies

Only 6 of the 23 participants in the HTML condition decided to open a second tab to keep the help open while working with the UI of the system. Only one participant opened a second window and split the screen between help pages and UI. The remaining 17 subjects continuously navigated back and forth between controlling UI and help pages. The average number of navigations needed to complete all tasks with this strategy was 24.47 (SD = 7.50).

Two of the participants using a second tab opened it at the first task, one at task three, one at task five, and one at task six.

7.3.2. Objective Scales

Means and standard deviations for objective scales are summarised in Table 7.1.

<table>
<thead>
<tr>
<th></th>
<th>interactive M</th>
<th>interactive SD</th>
<th>HTML M</th>
<th>HTML SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>time login [sec]</td>
<td>112.46</td>
<td>127.38</td>
<td>52.16</td>
<td>55.39</td>
</tr>
<tr>
<td>time task 1 [sec]</td>
<td>217.02</td>
<td>219.69</td>
<td>348.87</td>
<td>257.12</td>
</tr>
<tr>
<td>time task 2 [sec]</td>
<td>123.75</td>
<td>166.08</td>
<td>107.65</td>
<td>43.46</td>
</tr>
<tr>
<td>time task 3 [sec]</td>
<td>336.17</td>
<td>331.94</td>
<td>364.69</td>
<td>230.49</td>
</tr>
<tr>
<td>time task 4 [sec]</td>
<td>88.72</td>
<td>99.08</td>
<td>199.6</td>
<td>106.38</td>
</tr>
<tr>
<td>time task 5 [sec]</td>
<td>160.63</td>
<td>221.26</td>
<td>216.29</td>
<td>166.45</td>
</tr>
<tr>
<td>time task 6 [sec]</td>
<td>107.45</td>
<td>23.48</td>
<td>227.65</td>
<td>140.51</td>
</tr>
<tr>
<td>time task 7 [sec]</td>
<td>41.22</td>
<td>26.74</td>
<td>75.52</td>
<td>61.82</td>
</tr>
<tr>
<td>time overall [sec]</td>
<td>1095.27</td>
<td>477.74</td>
<td>1580.39</td>
<td>443.32</td>
</tr>
<tr>
<td>wrong input overall [#]</td>
<td>4.74</td>
<td>4.07</td>
<td>10.65</td>
<td>4.45</td>
</tr>
<tr>
<td>try and error overall [#]</td>
<td>1.21</td>
<td>1.05</td>
<td>1.25</td>
<td>1.14</td>
</tr>
<tr>
<td>restarts overall [#]</td>
<td>1.25</td>
<td>1.21</td>
<td>1.63</td>
<td>1.77</td>
</tr>
</tbody>
</table>

Table 7.1.: Means and standard deviations for objective scales.

Times

Figure 7.6 displays the average times participants in the HTML and interactive condition needed to complete each of the tasks and the overall time needed for all tasks.

Participants in the interactive condition were significantly faster than participants in the HTML conditions solving task 1 \( t(39) = 1.77, p = .04, d = .55 \), task 4 \( t(39) = 3.46, p < .001, d = 1.08 \), task 6 \( t(37) = 3.77, p < .001, d = 1.21 \), task 7 \( t(37) = 2.27, p = .01, d = .73 \), and to complete all tasks combined overall \( t(37) = 3.28, p = .001, d = 1.05 \). To login participants in the interactive condition participants were significantly slower than participants in the HTML condition \( t(39) = 1.95, p = .03, d = \)
7.3. Results

The times needed for task 2 \((t(22.85) = .43, p = .34)\), task 3 \((t(39) = .32, p = .38)\), and task 5 \((t(38) = .90, p = .19)\) were not significantly different between conditions.

**Erroneous Input**

The average number of wrong input values for each task and all tasks combined is summarised in Figure 7.7. There were significantly errors made by participants in the *HTML* condition than by participants in the *interactive* condition for task 1 \((Z = 1.65, p = .05, r = .34)\), task 2 \((Z = 3.20, p < .001, r = .58)\), task 3 \((Z = 1.96, p = .03, r = .37)\), task 4 \((Z = 2.73, p = .003, r = .48)\), task 6 \((Z = 2.48, p = .007, r = .41)\), and overall \((Z = 3.86, p < .001, r = .60)\). For the remaining items login \((Z = .82, p = .21)\), task 5 \((Z = .20, p = .42)\), and task 7 \((Z = .94, p = .17)\) no significant differences could be found.

**Trial and Error Attempts and Resets**

Figure 7.8 summarises the results for trial and error attempts to solve tasks and the average number of overall task resets dependent on condition. There were no significant differences dependent on condition for the number of trial and error attempts for login \(\chi^2(1, 31) = .30, p = .29\), task 1 \(\chi^2(1, 39) = .27, p = .30\), task 2 \(\chi^2(1, 37) = .12, p = .37\), task 3 \(\chi^2(1, 39) = .21, p = .32\), task 5 \(\chi^2(1, 39) = .17, p = .33\), task 7 \(\chi^2(1, 39) = .29, p = .29\), and overall \(\chi^2(3, 26) = 3.3, p = .17\). For the number of resets there was also no significant difference \((Z = .39, p = .35)\). Only task 4
7. Advanced Help System vs. the Traditional Approach

Figure 7.7.: Average amount of wrong input entered for each task and for all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$

Figure 7.8.: Average number of trial and error attempts and resets per task and all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$
\( \chi^2(1, 38) = 2.07, p = .07 \) and task 6 \( \chi^2(1, 38) = 1.9, p = .08 \) showed marginal significant more try and error attempts in the HTML condition than in the interactive condition.

**Influence of Personality Traits on Objective Scales**

In the HTML condition high values in neuroticism correlated with a high number of trial and error attempts \( r(12) = .66, p = .02 \) and participants with high values in conscientiousness needed less time to complete the tasks \( r(19) = -.48, p = .04 \).

For the interactive condition high values for self-efficacy correlated with a high number of erroneous inputs \( r(21) = .51, p = .02 \) and high values in openness correlated with a low number of trial and error attempts \( r(14) = -.6, p = .02 \) and a high number of erroneous inputs \( r(21) = .48, p = .03 \). In case of agreeableness high values correlated with a low number of restarts \( r(20) = -.52, p = .02 \) and less time needed to complete all tasks \( r(20) = -.51, p = .02 \).

### 7.3.3. Subjective Scales

**Additional questions**

For the annoyance scale that was evaluating to which degree participants were annoyed with having to permanently switch between pages with tutorials and actual UI of the power plant participants reported a mean level of annoyance of 5.42 \( (SD = 1.62) \). Concerning the advantageous scale that was assessing if participants perceived the help system used in the interactive condition to be better than traditional help system, the mean rating was 5.38 \( (SD = 1.79) \).

**Own Questionnaire**

Means and standard deviations for the results attained with the self developed questionnaire are summarised in Table 7.2.

<table>
<thead>
<tr>
<th></th>
<th>interactive</th>
<th></th>
<th></th>
<th>HTML</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
<td></td>
</tr>
<tr>
<td>intrusiveness</td>
<td>1.76</td>
<td>1.51</td>
<td>1.35</td>
<td>.59</td>
<td></td>
</tr>
<tr>
<td>help behaviour</td>
<td>6.08</td>
<td>1.34</td>
<td>6.08</td>
<td>.83</td>
<td></td>
</tr>
<tr>
<td>competence</td>
<td>5.63</td>
<td>1.63</td>
<td>6.23</td>
<td>.78</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2.: Means and standard deviations for the results attained with our own questionnaire.

Figure 7.9 displays the results attained for our own questionnaire for the HTML and interactive condition. The help system in the interactive condition was rated marginally significant worse than the system HTML for the competence scale \( t(29.0) = 1.51, p = .07 \). The intrusiveness scale \( t(39) = 1.14, p = .13 \), and the help behaviour scale
7. Advanced Help System vs. the Traditional Approach

Figure 7.9.: Average rating of the systems for our own questionnaire depending on condition. For *intrusiveness* a low rating means positive evaluation, for *help behaviour* and *competence* high ratings are positive. Error bars are standard errors of the mean. * p < .05

(t(39) = .01, p = .50) did not show significantly better ratings for the *interactive* than for the *HTML* condition.

**CSUQ**

The means and standard deviations for the *HTML* and *interactive* conditions for the CSUQ questionnaire are summarised in Table 7.3

<table>
<thead>
<tr>
<th></th>
<th>interactive</th>
<th></th>
<th>HTML</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>sysuse</td>
<td>5.11</td>
<td>1.59</td>
<td>5.08</td>
<td>1.21</td>
</tr>
<tr>
<td>infoqual</td>
<td>5.93</td>
<td>1.31</td>
<td>5.83</td>
<td>.92</td>
</tr>
<tr>
<td>interqual</td>
<td>4.79</td>
<td>1.48</td>
<td>4.8</td>
<td>1.18</td>
</tr>
<tr>
<td>overall</td>
<td>5.38</td>
<td>1.31</td>
<td>5.31</td>
<td>.99</td>
</tr>
</tbody>
</table>

Table 7.3.: Means and standard deviations for the results attained with the CSUQ questionnaire.

Figure 7.10 gives an overview of the results for the CSUQ questionnaire. The ratings for both systems on the different scales of the CSUQ questionnaire were very similar. Thus neither for *sysuse* (t(39) = .07, p = .47), *infoqual* (t(39) = .29, p = .39), *interqual*
7.3. Results

Figure 7.10.: Average rating of the systems for the CSUQ questionnaire dependent on condition. Error bars are standard errors of the mean.

\( (t(39) = .02, p = .49) \), nor overall \( (t(39) = .19, p = .43) \) there were significant differences.

**AttrakDiff**

Finally the means and standard deviations for the AttrakDiff questionnaire are summarised in Table 7.4 and Figure 7.11.

<table>
<thead>
<tr>
<th></th>
<th>interactive</th>
<th></th>
<th>HTML</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>pragmatic quality</td>
<td>5.21</td>
<td>1.05</td>
<td>5.32</td>
<td>1.04</td>
</tr>
<tr>
<td>hedonic quality - identity</td>
<td>4.71</td>
<td>.92</td>
<td>4.21</td>
<td>.86</td>
</tr>
<tr>
<td>hedonic quality - stimulation</td>
<td>4.39</td>
<td>.75</td>
<td>3.2</td>
<td>.94</td>
</tr>
<tr>
<td>attractiveness</td>
<td>4.83</td>
<td>1.01</td>
<td>4.56</td>
<td>.88</td>
</tr>
</tbody>
</table>

Table 7.4.: Means and standard deviations for the results attained with the AttrakDiff questionnaire.

The system used in the *interactive* condition was rated significantly better than the system in the *HTML* condition for *hedonic quality - identity* \( (t(39) = 1.82, p = .04, d = .57) \) and *hedonic quality - stimulation* \( (t(39) = 4.52, p < .001, d = 1.41) \) and indifferent for *pragmatic quality* \( (t(39) = .34, p = 0.37) \) as well as *attractiveness* \( (t(39) = .92, p = .18) \).

Concerning single items the *interactive* system was rated significantly worse than the
7. Advanced Help System vs. the Traditional Approach

![Graph showing average rating of systems for AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.]

HTML based help system for technical - human \((Z = 1.74, p = .04, r = .27)\) and significantly better for unprofessional - professional \((Z = 2.36, p = .009, r = .37)\), tacky - stylish \((Z = 1.72, p = .04, r = .27)\), conventional - inventive \((Z = 3.61, p < .001, r = .56)\), unimaginative - creative \((Z = 3.32, p < .001, r = .52)\), conservative - innovative \((Z = 2.75, p = .003, r = .43)\), dull - captivating \((Z = 2.51, p = 0.006, r = .39)\), and ordinary - novel \((Z = 2.76, p = .003, r = .43)\)

The better rating for alienating - integrating \((Z = 1.43, p = .08)\) and unpresentable - presentable \((Z = 1.52, p = .06, r = .24)\) was marginal significant better for the interactive than for the HTML condition.

Influence of Personality Traits on Subjective Scales

For the HTML condition high values in self-efficacy correlated with high ratings for sysuse \((r(20) = .55, p = .01)\), infoqual \((r(20) = .48, p = .03)\), and overall \((r(20) = .55, p = .01)\) from the CSUQ questionnaire. Participants with higher values for neuroticism perceived the switch between help content and UI to be more annoying than participants with lower values for neuroticism \((r(20) = .45, p = .05)\). Openness also had an influence on how annoying the context switch was perceived to be and participants with high values for openness considered it to be less annoying \((r(20) = -.46, p = .04)\). Those participants with high openness also rated the help behaviour of the system to be worse \((r(20) = -.5, p = .02)\).

In the interactive condition self-efficacy negatively correlated with help behaviour \((r(21) = -.47, p = .03)\) and competence \((r(21) = -.45, p = .04)\), and positively with
intrusiveness \( r(21) = .5, p = .02 \). For the remaining personality traits there were no significant correlations to subjective scales.

### 7.3.4. Changes in Self-efficacy

In the HTML condition reported self-efficacy was significantly lower after interaction took place than before interaction \( t(19) = 1.9, p = .04, d = .26 \). For the interactive condition there was no effect on self-efficacy \( t(20) = .52, n.s. \).

### 7.4. Discussion

#### 7.4.1. Strategies

The strategies used by participants to have both the UI of the power plant and the help texts available in the HTML condition differs from what we expected. The number of participants using a second browser tab was rather low, indicating that opening multiple tabs in this kind of situation is not a very common strategy. This view is also supported by the finding that only two of the six participants opening a second tab for the help content did so at the first task.

Concerning the strategy of using two separate windows to have tutorial and UI of the power plant side-by-side, this strategy seems to be even less common despite its obvious benefits. However, as the system running on the computers was Ubuntu v12.10 and thus surely different to the Windows 7 predominant at the time of the study, that the participants might have been more accustomed to, most participants might not have been aware of that feature and thus not have tried it. Nevertheless, the demography of participants consisting mostly of younger students suggests that they were from a rather tech-savvy group and if they considered having the windows placed side by side to be advantageous in this particular situation, they would have been able to achieve this. They could for example also have changed the size and arrange the windows manually, without using the “snap” feature. Another important aspect in this context is that the resolution of screens of the laptops in use was only 1280x800 pixels on a 15.4 inch screen. Thus the available space was rather limited. With this limited amount of screen real estate having two windows parallel side by side might not be an approach that directly comes to mind.

#### 7.4.2. Objective Scales

**Times**

As expected the times needed to complete the tasks were higher in the HTML than in the interactive condition. Indicating that the implemented help system is more effective in assisting users to achieve goals than traditional help pages. The only exception is the time for the login task that was significantly higher in the interactive than in the HTML condition. Since the password needed to log in to the system was provided by the help system in the interactive condition users might have needed some time to familiarize
Figure 7.12.: Number of participants finishing the tasks grouped in 5 minute intervals dependent on condition.

with the system in order to get the information needed to log in. In the HTML condition the password was written on the introductory leaflet handed out to participants before starting interaction. Thus there was no need to consult the help system first and thus speeding up the login process.

To further investigate the times needed by participants in the HTML and interactive condition Figure 7.12 shows a summary of the number of participants finishing the tasks in five minute intervals.

This summary clearly shows that while most participants in the interactive condition were already finished after 20 minutes only the fastest of the HTML group were able to finish the tasks between 15 and 20 minutes. This clearly emphasizes the advantageousness of our interactive help system.

**Erroneous Input**

Similar to the results concerning completion times, the results for erroneous input support our hypothesis, stating that due to the higher cognitive load in the HTML condition users make more errors during task completion. The only exceptions where there were no significant differences between conditions were the login task, task 5, and task 7. The reasons for not finding differences for the login task are quite surely the same as for the quicker completion times for this task. Task 7 was quite simple and participants hardly made errors completing this task in either condition, causing a basement effect not allowing to find differences. Not finding a difference for task 5 is rather strange and from our perspective can only be attributed to a coincidence.
7.4. Discussion

Trial and Errors, and Resets

Concerning trial and error attempts and resets the results support our hypothesis that this scales are not dependent on condition, but on participants' personality traits.

Influence of Personality Traits on Objective Scales

We expected to find much more correlations between participants' personality traits and objective measures. Especially for participants' with high self-efficacy we expected higher numbers of trial and error attempts.

The correlation found between neuroticism and a high number of trial and error attempts in the HTML condition seems to be not very logical. One would expect insecure users to rely more heavily on assistance and thus produce less trial and error attempts than self confident users. However, having a closer look at trial and error attempts for single tasks reveals that those only correlate with neuroticism for the login task ($r(15) = .55, p = .03$), task 4 ($r(19) = .58, p = .009$), and task 6 ($r(20) = .56, p = .009$). Except for the login task those are the tasks following unpredictable problems with the power plant. Realising that unaffacted of own actions and even though instructions were followed perfectly, there is still a possibility for problems, might have caused participants with a low degree of emotional stability to no longer care about the instructions and just "let go" by making a random try at the task following a problem. The finding that participants with high values for conscientiousness needed less time to complete all tasks in the HTML condition is not surprising, since these participants are more likely to closely follow the instructions provided and thus save time completing tasks.

The results attained for the interactive condition are also not surprising. Participants with high values for self-efficacy might try to enter more input from what they remember from the last task involving the same input and thus enter more wrong input. Participants with high values for openness are more susceptible for help and produce less trial and error attempts. The high values for wrong input for participants with a high rating for openness might be caused by those users first trying to solve a task on their own, providing some wrong input, but not completing a full trial and error attempt by referring to the help system before the number of wrong inputs exceeded the trial and error attempt limit. For the agreeableness scale that correlated with a low number of restarts and less time needed to complete the tasks, participants with a high level of agreeableness might have wanted to achieve a socially desirable result by finishing the tasks quickly and not mess around with the system. Thus, those participants might have followed the instructions more closely and needed less restarts and less time. The finding that agreeableness had an influence in case of the interactive but not in the HTML condition might be an indicator that this wish to achieve a socially desirable result comes from applying a social script Nass and Moon [2000] to the interaction with the help system and applying such a script might not be deemed appropriate in case of a not interactive HTML based help system.
7. Advanced Help System vs. the Traditional Approach

7.4.3. Subjective Scales

The results attained for subjective scales are mostly not conforming to our initial hypothesis expecting to find many differences concerning subjective ratings between conditions. Especially for scales exclusively measuring the perceived performance of the help system, like the help behaviour scale from our own questionnaire, all CSUQ scales, and the pragmatic quality scale of AttrakDiff there were no differences. Since the material provided by both systems was essentially the same in both conditions this seems not to be surprising. Nevertheless we expected the beneficial aspects of the interactive help system to be reflected in the ratings for these scales. Especially since users stated to have been annoyed by the switch between help content and UI of the power plant in the HTML condition above average and also reported to perceive the interactive help to be advantageous to traditional help systems in the interactive condition. Thus, participants seemed to be well aware of the advantages of the interactive system and drawbacks of the system in the HTML condition, when asked directly. This shows that it is hard to evaluate the performance of help systems relying on self reported measures only.

As soon as the subjective scales included hedonic aspects, especially the two hedonic scales of the AttrakDiff questionnaire, differences in the rating could be found. These differences were also conforming to our hypothesis and the system in the interactive condition was perceived to be superior to the system in the HTML condition. Only for the competence scale the system in the HTML condition was marginally significantly perceived to be more competent than in the interactive condition. This difference might be due to the fact that extraneous material, like the explanations of the functional principle of a nuclear power plant were always presented to the user in the HTML condition. Thus users were aware that the system was able to offer this kind of “irrelevant” material. In case of the interactive system, this kind of information was only presented in case participants actively requested it. Thus participants in the interactive condition might not have been aware of the availability of this kind of information and thus might have perceived the system to be less competent.

That the system in the interactive condition was not perceived to be more intrusive than the system in the HTML condition, even though the system was displayed along with the UI of the power plant, is an interesting finding. It shows that measures to keep the help system from being perceived to be intrusive, like allowing users to relocate the representation of the help system on screen, were effective.

Influence of Personality Traits on Subjective Scales

Similar to the objective scales we also expected personality traits to have a bigger influence on the subjective ratings of the help systems. The correlations found are not surprising and seem to be quite logical. Users with high self reported self-efficacy feel more competent to use complex HTML based help pages and thus rate the performance of this kind of help to be higher in the CSUQ questionnaire than users with lower self-efficacy. Less emotionally stable participants are more likely to be annoyed in general and thus are also more likely to be annoyed with the permanent context switch between
help content and UI of the power plant. In contrast to this, people open to experience do not mind too much about this annoying property of the system.

In the interactive condition participants with high self-efficacy might have felt infantilised by the system and might have expected the system to offer more in extraneous information concerning task completion. This might be the cause for the worse ratings for the scales from our own questionnaire by participants with high values for self-efficacy.

### 7.4.4. Changes in Self-efficacy

The results obtained with our experiment show that help systems can have an influence on perceived self-efficacy of users. Using the interactive help system participants might have perceived the task and problem solving process to be a collaborative process between them and the help system and thus they might have attributed success at least partly to themselves. Even though this problem solving process was quite the same in the HTML condition participants might have felt more like just following orders in this condition. Thus attributing success exclusively to the help system, what finally lead to the decrease of perceived self-efficacy.

### 7.5. Conclusion

Our results for the objective scales clearly show that compared to traditional help systems providing users with HTML based help texts, our system is much more efficient considering task completion times as well as mistakes made during task completion. However, this increase in performance is not reflected in participants’ subjective rating of the performance of the system. If the subjective rating includes hedonic aspects our system is rated significantly better than traditional HTML web pages. Personality traits have a limited influence on objective as well as subjective measures. Indicating that there might not be a need to consider different personality traits implementing help systems and there is a help system applicable to all kinds of users regardless of personal characteristics.

The results also showed that the kind of help system used can have an influence on the perceived self-efficacy of users and might have an influence on the attribution of success and failure.
8. Animated Agent vs. Depiction of a Question Mark

Many users prefer to refer to a third person for help experiencing problems with computer software [Lang et al., 2013a]. Hence, the hypothesis arises that by introducing a virtual anthropomorphic agent and hence creating a virtual peer, especially users that would otherwise ask a third person for help, could be more inclined to consult the anthropomorphic agent than a help system not employing such an agent. Additionally the results using virtual agents reported in Section 2.2 seem in general to be in favour of using virtual agents.

On the other hand there is the negative reception Clippy and the hype surrounding him that might have caused resentments towards virtual agents in general.

In order to validate the influence of introducing an anthropomorphic agent into our help system we performed the following experiment comparing users subjective acceptance of the help system using a generic question mark or an anthropomorphic agent.

8.1. Method

8.1.1. Conditions

The difference in treatment for this experiment was that one group was interacting with a help system represented by an animated, anthropomorphic agent (agent), and a help system represented by a generic question mark (question mark). The different portrayal of the help systems for both conditions is shown in Figure 8.1.

8.1.2. Participants

In the agent condition overall 26 subjects participated in the experiment. The mean age of participants was 23.13 years ($SD = 5.8$) 9 participants were male and 15 female. 2 subjects were excluded from subjective analysis due to incomplete data. Participants in this group received heterogeneous incentives. 5 participants were compensated with 5 € for their efforts and the remaining subjects had the chance to win cinema tickets. To determine if the kind of incentive had an influence on results, we compared the subjective responses of participants in the “money” group to responses of the “lottery ticket” group. For none of the 12 scales evaluated in this experiment there were significant differences between the “money” and the “cinema” ticket group. Since this is consistent with the findings of Göritz [2004] which indicate that the kind of incentive not having an effect on the results of questionnaires, we decided to neglect a possible influence of compensation and subsumed all subjects into one group.
8. Animated Agent vs. Depiction of a Question Mark

Figure 8.1.: Representation of the help system in the *agent* (left) and *question mark* condition (right).

Participants in the *question mark* condition were the participants in the *interactive* condition of Chapter 7.

8.1.3. Questionnaires

The questionnaires used for this experiment were the ones described in Section 6.4. The only scale that had to be discarded from our own questionnaire was the *optimism* scale. Some of the questions for the other scales were also not appropriate in the context of the *question mark* condition. Those were one question in *intrusiveness*, three questions in *dominance*, one question for the *humanity* scale and two questions for the *competence* scale. The items missing in the *question mark* condition were also not considered computing the scales for the *agent* condition for this experiment.

In addition to this, since we were rather interested in the subjective differences of the rating between the systems and not the usability, which was virtually the same for both conditions, we decided to leave out the CSUQ questionnaire. Since primary objective of this questionnaire is assessing a systems usability, we did not expect differences between groups concerning this questionnaire for the given experiment. Especially, since this questionnaire did not reveal any differences for the conditions described for the previous experiment illustrated in Chapter 7 and the difference in usability between conditions in this experiment was much higher than for this experiment and thus much more likely to produce different results than for the experiment described in this section.

As this experiment did not involve spoken in- or output the SASSI questionnaire was also excluded from the post interaction questionnaires.
8.2. Hypotheses

Since the information and assistance provided under either condition was absolutely identical, we did not expect differences concerning the objective scales between the agent and the question mark condition.

For the subjective scales we expected differences between the agent and the question mark. While the persona effect (see Section 2.2) would suggest a positive effect of using an anthropomorphic agent in this context, bad experiences with Clippy (see Section 3.7) might have caused resentments concerning animated agents in the context of help systems. This is why we were not sure what to expect and decided to formulate a two tailed hypothesis.

8.3. Results

8.3.1. Objective Scales

Means and standard deviations for objective scales are summarised in Table 8.1.

<table>
<thead>
<tr>
<th></th>
<th>agent M</th>
<th>SD</th>
<th>question mark M</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>time login [sec]</td>
<td>104.52</td>
<td>62.94</td>
<td>112.46</td>
<td>127.38</td>
</tr>
<tr>
<td>time task 1 [sec]</td>
<td>111.83</td>
<td>88.02</td>
<td>217.02</td>
<td>219.69</td>
</tr>
<tr>
<td>time task 2 [sec]</td>
<td>97.27</td>
<td>151.21</td>
<td>123.75</td>
<td>166.08</td>
</tr>
<tr>
<td>time task 3 [sec]</td>
<td>297.43</td>
<td>307.46</td>
<td>336.17</td>
<td>331.94</td>
</tr>
<tr>
<td>time task 4 [sec]</td>
<td>101.88</td>
<td>74.16</td>
<td>88.72</td>
<td>99.08</td>
</tr>
<tr>
<td>time task 5 [sec]</td>
<td>155.28</td>
<td>116.87</td>
<td>160.63</td>
<td>221.26</td>
</tr>
<tr>
<td>time task 6 [sec]</td>
<td>121.4</td>
<td>47.32</td>
<td>107.45</td>
<td>23.48</td>
</tr>
<tr>
<td>time task 7 [sec]</td>
<td>33.67</td>
<td>13.27</td>
<td>41.22</td>
<td>26.74</td>
</tr>
<tr>
<td>time overall [sec]</td>
<td>998.06</td>
<td>507.81</td>
<td>1095.27</td>
<td>477.74</td>
</tr>
<tr>
<td>wrong input overall [#]</td>
<td>4.17</td>
<td>3.91</td>
<td>4.74</td>
<td>4.07</td>
</tr>
<tr>
<td>try and error overall [#]</td>
<td>.57</td>
<td>.87</td>
<td>1.21</td>
<td>1.05</td>
</tr>
<tr>
<td>restarts overall [#]</td>
<td>.57</td>
<td>.95</td>
<td>1.25</td>
<td>1.21</td>
</tr>
</tbody>
</table>

Table 8.1.: Means and standard deviations for objective scales.

**Times**  The average times needed to completed each task and all tasks combined are visualized in Figure 8.2 The only task with a significant difference in time needed to complete the task is task 1 \( (t(25.37) = 2.06, p = .05, d = .65) \). For the remaining tasks, login \( (t(45) = .28, p = .78) \), task 2 \( (t(45) = .57, p = .57) \), task 3 \( (t(45) = .41, p = .68) \), task 4 \( (t(45) = .52, p = .6) \), task 5 \( (t(41) = .1, p = .92) \), task 6 \( (t(41) = 1.20, p = .24) \), task 7 \( (t(26.95) = 1.15, p = .26) \), and for completion of all tasks combined overall \( (t(40) = .64, p = .53) \) there were no significant differences.
8. Animated Agent vs. Depiction of a Question Mark

Figure 8.2.: Average times needed to complete the given tasks dependent on condition. Error bars are standard errors of the mean. *\( p < .05 \)

Erroneous Input

Figure 8.3 shows the distribution of wrong input for each task and all tasks combined depending on condition. Concerning wrong input only for task 2 the number of wrong inputs provided showed significant differences between groups (\( Z = 2.32, p = .02, r = .38 \)). For the other tasks, login (\( Z = 1.13, p = .26 \)), task 1 (\( Z = 1.17, p = .24 \)), task 3 (\( Z = 1.1, p = .27 \)), task 4 (\( Z = .16, p = .87 \)), task 5 (\( Z = .13, p = .9 \)), task 6 (\( Z = .68, p = .5 \)), task 7 (\( Z = .42, p = .68 \)), and overall (\( Z = .45, p = .65 \)) there were no significant differences.

Trial and Error Attempts and Resets

An overview of trial and error attempts per task and overall, as well as the average number of resets dependent on condition is given in Figure 8.4. There was a significant difference between conditions concerning the trial and error attempts made for task 1 (\( \chi^2(1,45) = 5.83, p = .02, \phi = .36 \)), resets (\( Z = 2.11, p = .03, r = .32 \)) and a marginal significant difference for task 3 (\( \chi^2(1,44) = 2.76, p < .1 \)). For login (\( \chi^2(1,40) = .96, p = .33 \)), task 2 (\( \chi^2(1,44) = .24, p = .63 \)), task 4 (\( \chi^2(1,45) = .05, p = .82 \)), task 5 (\( \chi^2(1,45) = .79, p = .38 \)), task 7 (\( \chi^2(1,43) = .53, p = .47 \)), and overall (\( \chi^2(3,45) = 4.06, p = .26 \)) there were no significant differences. For task 6 there were no try and error attempts in both groups.
8.3. Results

Figure 8.3.: Average amount of wrong input entered for each task and for all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$

Figure 8.4.: Average number of trial and error attempts and resets per task and all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$
8. Animated Agent vs. Depiction of a Question Mark

Influence of Personality Traits on Objective Scales

The results for the question mark condition are reported as results for the interactive condition of Section 7.3.2.

In the agent condition there were no correlations between personality traits and subjective scales at all.

8.3.2. Subjective Scales

Own Questionnaire

Means and standard deviations for the results attained with the self developed questionnaire are summarised in Table 8.2.

<table>
<thead>
<tr>
<th></th>
<th>agent</th>
<th></th>
<th>question mark</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>intrusiveness</td>
<td>1.92 1.02</td>
<td>1.74 1.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>distraction</td>
<td>2.7 1.33</td>
<td>2.91 1.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dominance</td>
<td>1.75 .78</td>
<td>2.23 .9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>humanity</td>
<td>2.55 1.1</td>
<td>3.66 1.27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tolerance</td>
<td>5.54 1.28</td>
<td>5.33 1.33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>help behaviour</td>
<td>6.25 1.1</td>
<td>6.04 1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>competence</td>
<td>5.82 1.18</td>
<td>5.41 1.49</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: Means and standard deviations for the results attained with our own questionnaire.

Figure 8.5 displays the results attained for our own questionnaire for the agent and question mark condition. Intrusiveness \((t(43) = .51, p = .61)\), distraction \((t(42) = .54, p = .59)\), tolerance \((t(42) = .55, p = .59)\), competence \((t(43) = 1.02, p = .31)\), and help behaviour \((t(43) = .58, p = .57)\) did not reveal significant differences between conditions. For dominance there was a marginal significant \((t(42) = 1.88, p = .07)\) and for humanity \((t(42) = 3.11, p = .003, d = .94)\) a significant difference.

AttrakDiff

The means and standard deviation attained with the AttrakDiff questionnaire are summarised in Table 8.3.
Figure 8.5.: Average rating of the system using an anthropomorphic agent (agent) and a generic question mark (question mark) for our own questionnaire. For the scales on the left a low rating means positive evaluation, for the scales on the right high ratings are positive. Error bars are standard errors of the mean. * $p < .05$
8. Animated Agent vs. Depiction of a Question Mark

Figure 8.6.: Average rating of the systems for the AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.

<table>
<thead>
<tr>
<th></th>
<th>agent</th>
<th>question mark</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M</strong></td>
<td>5.28</td>
<td>5.21</td>
</tr>
<tr>
<td><strong>SD</strong></td>
<td>.72</td>
<td>1.05</td>
</tr>
<tr>
<td>pragmatic quality</td>
<td>4.82</td>
<td>4.71</td>
</tr>
<tr>
<td>hedonic quality - identity</td>
<td>.73</td>
<td>.92</td>
</tr>
<tr>
<td>hedonic quality - stimulation</td>
<td>.7</td>
<td>.75</td>
</tr>
<tr>
<td>attractiveness</td>
<td>4.95</td>
<td>4.83</td>
</tr>
<tr>
<td></td>
<td>.66</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Table 8.3.: Means and standard deviations for the results attained with the AttrakDiff questionnaire.

Figure 8.6 shows the results for all scales and items of the AttrakDiff questionnaire. For none of the AttrakDiff questionnaire scales, pragmatic quality ($t(43) = .26, p = .8$), hedonic quality - identity ($t(43) = .44, p = .67$), hedonic quality - stimulation ($t(43) = .31, p = .76$), and attractiveness ($t(43) = .47, p = .64$) could we find significant differences between conditions. The only item showing a marginal significant difference is unprofessional - professional ($Z = 1.89, p = .06$).

Influence of Personality Traits on Subjective Scales

Most of the correlations between subjective scales and personality traits for the question mark condition have already been reported in Section 7.3.3. However, two correlations were not reported, since the corresponding scales have been excluded from the results of Chapter 7 as there were no equivalent counterparts in the HTML condition of that section.
For these remaining scales high values for self reported extraversion correlated with a low rating for tolerance \((r(20) = -0.51, p = .02)\) and humanity \((r(20) = -0.51, p = .02)\) and openness correlated negatively with distraction \((r(20) = -0.46, p = .04)\) from our own questionnaire.

In the agent condition high values of neuroticism correlated with low values for pragmatic quality \((r(23) = -0.49, p = .02)\), hedonic quality - identity \((r(23) = -0.52, p = .01)\), and hedonic quality - stimulation \((r(23) = -0.42, p = .05)\).

High values for extraversion correlated positively with help behaviour \((r(23) = 0.47, p = .02)\), optimism \((r(23) = 0.59, p = .003)\), tolerance \((r(23) = 0.58, p = .004)\), hedonic quality - identity \((r(23) = 0.43, p = .04)\), and attractiveness \((r(23) = 0.45, p = .03)\). For extraversion and intrusiveness there was a negative correlation \((r(23) = -0.53, p = .009)\).

In case of agreeableness high values correlated positively with the rating for attractiveness from the AttrakDiff questionnaire \((r(23) = 0.55, p = .006)\).

The conscientiousness scale correlated positive with the help behaviour \((r(23) = 0.68, p < .001)\), tolerance \((r(23) = 0.65, p = .001)\), humanity \((r(23) = 0.5, p = .02)\), and competence \((r(23) = 0.64, p = .001)\) scales from our own questionnaire and with the hedonic quality - identity \((r(23) = 0.62, p = .002)\), hedonic quality - stimulation \((r(23) = 0.45, p = .03)\), and attractiveness \((r(23) = 0.71, p < .001)\) scales of the AttrakDiff questionnaire. Similar to extraversion there was a negative correlation between conscientiousness and intrusiveness from our own questionnaire \((r(23) = -0.52, p = .01)\).

Self-efficacy only correlated positively with the rating for hedonic quality - identity \((r(23) = 0.44, p = .04)\).

### 8.4. Discussion

#### 8.4.1. Objective Scales

For most of the objective scales attained results confirm our hypothesis. However, there is a very interesting difference concerning the trial and error attempts and time needed to complete the first task between conditions. The time needed to complete this task as well as the number of trial and error attempts in the question mark condition were significantly higher than in the agent condition. We consider this an indicator that after the login task at which participants were explicitly told to consult the help system to get the information needed to log in to the portal, participants were more curious to interact with the anthropomorphic agent than they were in case of the generic question mark. This kind of curiosity might be beneficial, when trying to introduce users to help systems and make them aware of the existence and capabilities of such a help system. Thus, achieving the objective of making users aware of a help system without using a possibly detrimental proactive approach to introducing the help system. The difference in the number of restarts is also supporting this view. Especially since a closer investigation of restarts per task reveal that the differences were for task 1 (1 restart in the agent vs. 7 restarts in the question mark condition) and for task 3 (5 agent, 15 question mark). Thus, the differences were exactly at points were a user could try solving a problem without assistance. The first point is when the actual interaction with the power plant...
8. Animated Agent vs. Depiction of a Question Mark

begins and the second point is after the first warning message indicating a problem was displayed. The finding that users needed less restarts in both situations in the agent condition shows that users were more willing to ask the anthropomorphic agent for help when confronted with a particular kind of problem for the first time and also felt more confident using the agent compared to the question mark and less often had the feeling of having messed up and needing to restart the given task from scratch. The significant difference in trial and error attempts for task 1 and marginal significant difference for task 3 are also conforming to this notion. Thus, our objective data seems to be support the idea of a persona effect (see Section 2.2).

Influence of Personality Traits on Objective Scales

While personality traits had some influence on the objective results attained in the question mark condition (see Chapter 7) personality traits had no influence in case of the agent condition. Thus, using an anthropomorphic agent seems to have a positive influence on compensating for different personal traits when it comes to efficiency and performance of help system usage.

8.4.2. Subjective Scales

Similar to our first experiment we expected to find more differences between conditions concerning subjective scales. Especially for the AttriakDiff questionnaire there were virtually no differences at all. The marginal difference for unprofessional – professional might be caused by the representation of the agent in the agent condition that was being displayed wearing a lab coat. This might have caused the agent to be perceived as more professional than the generic question mark in the question mark condition. Concerning our own questionnaire we found some differences. That the system not using an anthropomorphic agent was perceived to be more human than the system using the question mark seems to be a bit odd. Perhaps participants were rejecting the idea of conceding human characteristics to the rather comical representation of the agent used in the agent condition and thus the lower rating is due to a response of defiance. The tendency to rate the system represented by a question mark to be rated to be more intrusive than the anthropomorphic agent is an interesting finding. It might be an indicator that while users neglect the agent to have human and social characteristics to them they nevertheless appreciate the fact that the agent does not interrupt them and obviously tries not to interfere with them during task completion. This characteristic of the system might be more appreciated if an anthropomorphic, more human representation is used than for a system not represented by such an anthropomorphic agent.

Despite of the differences found it may be stated that results attained with the subjective scales used were rather similar for both conditions. Thus, using an anthropomorphic agent does not have a huge influence on the subjective rating of help systems.
Influence of Personality Traits on Subjective Scales

Concerning the influence of personality traits on subjective rating, the correlations for additional scales not already covered in Chapter 7 indicate that the help system using the question mark is perceived to display lower levels of tolerance and humanity by participants with high self reported extraversion. This might be caused by very outgoing, social, and talkative people expecting more social properties from a computer system to consider them to have social properties like humanity and tolerance.

Similar to participants being less annoyed by the permanent context switch between help content and UI of the HTML condition from Chapter 7 participants with high values for openness felt less distracted by the help system in the question mark condition.

Different to the results for objective scales personality traits seem to have a big influence on the subjective rating of the help system in the agent condition and we found a lot of personality traits correlating with subjective scales of our questionnaires. Overall it seems like users with high levels of extraversion and conscientiousness rate the system to be better in the agent condition for a broad range of scales. Reasons for the better ratings given by participants with high levels of extraversion seem to be rather obvious. More outgoing and social users seem to enjoy interacting with an anthropomorphic agent more than rather solitary and reserved users. Thus, while participants neglected the system in the agent condition to have human properties the finding that social users like interacting with the anthropomorphic system more than less social users is a strong indicator that interaction with the system is perceived to be social nevertheless. The positive influence of conscientiousness on the rating of the system is rather hard to explain. This might also be due to the interaction being perceived to be social in the agent condition and participants not considering themselves very dependable and self-disciplined might feel monitored by the agent and thus perceive the interaction with the system to be less enjoyable. This view might also be supported by neuroticism having in general a negative influence on the rating of the help system using the anthropomorphic agent. Sensitive and nervous users might also feel monitored by the agent and feel uncomfortable in its presence. Compared to the three personality traits mentioned above the remaining traits seem to have limited influence on the subjective rating of the system. This was quite surprising to us since we especially expected agreeableness to have an effect in a pseudo social environment involving an anthropomorphic agent.

8.5. Conclusion

The results attained with the experiment show that using an anthropomorphic agent in a help system has limited influence on the subjective rating of such a help system. Nevertheless our objective results suggest that users tend to consult the help system using the agent at an earlier stage, when problems arise than a system not using an anthropomorphic agent. This might be due to users being curious to interact with the agent. Using a generic question mark seems not to be able to arouse the same kind of curiosity. This makes anthropomorphic agents useful to introduce users to help systems and their features, since they seem to be able to assure that users activate the help system
8. Animated Agent vs. Depiction of a Question Mark

at least once out of curiosity.
9. Reactive vs. Proactive Agent

Another aspect of Clippy that comes to mind considering Clippy’s negative review from users is the proactive behaviour shown. Especially, since most of the times when Clippy tried to proactively help users his advice was perceived to be rather unhelpful.

However, users hardly use the internal help system of software [Lang et al., 2013a] and thus there is a high risk users might not be aware of the existence of the help system in general or might miss to explore the help system’s capabilities. This seems to make a system that is able to proactively assist participants in certain situations a favourable approach. However, the stakes of finding the right time, the situation at which a participant actually needs help and identifying the kind of assistance a user might appreciate are very high. Since, as for the example of Clippy, a system providing inappropriate suggestions at inappropriate times will be perceived to be an annoyance quite quickly.

Xiao et al. [2003] evaluated the effects of proactivity in the context of animated agents in a text editing domain. However, their results show that proactivity for an animated agent using spoken output was appreciated by users. Accordingly they also could not find evidence for a substantial difference in users’ perception of the agent between the proactive and the reactive group. They concluded that proactive behaviour is not problematic in general as long as the information conveyed is actually valuable and presented at the right time. Kim and Baylor [2006] stated that proactive agents can have a positive impact on recall compared to reactive ones. In their opinion this is due to the fact, that not all information theoretically provided by the agent is requested by the user in reality. However, in terms of subjective evaluation no differences between a proactive and a reactive version of the agent could be found. The main differences between Clippy and the two systems used for the evaluations above is that while Clippy uses text based output the other systems rely on speech output. An agent providing textual help might be perceived to be more disturbing than the agent employed by Xiao et al. [2003]; Kim and Baylor [2006] using spoken output as modality. Conforming to Baddeley’s model of working memory (see Section 2.1.1) spoken output seems to be the preferable choice for agents assisting with graphical user interfaces anyway. Nevertheless, computer users are often forced to use systems without auditory output. Especially in open-plan offices auditory output may be distracting. Another aspect of the agent of Xiao et al. [2003] is that its representation is separated from the actual content of the UI and does not interfere with it. This was definitely not true for Clippy that regularly popped up even in the middle of typing and thus breaking the user’s flow of thought [Dix et al., 2010].

In order to verify if users consider a proactive help system using text as output modality and being displayed in a way interfering with the content of the UI was perceived to be inferior by users compared to help system not showing proactivity, we performed
9. Reactive vs. Proactive Agent

an experiment [Lang et al., 2013b] comparing a proactive system and a system not proactively offering help to users.

9.1. Method

9.1.1. Conditions

The difference in treatment for this experiment was that one group was interacting with a help system permanently showing proactive behaviour (proactive) to a second system only reacting on participants' requests for help (reactive).

The system used in the reactive condition was the one described under the agent condition of Chapter 8. Proactive behaviour in the proactive condition was always shown as soon as the user indicated that he had finished reading a task description or warning by clicking the corresponding buttons of the text area containing the task description and warning (see Figure 6.6 and Figure 6.8). After the respective button was clicked, the help system was activated automatically and offered assistance (see Figure 5.8). Thus help was offered at a well defined point in the interaction process. By always offering a step by step guide appropriate to solve the current task as first option in the list of suggested alternatives for assistance it was also guaranteed that the help offered by the system was always highly relevant to task completion. Due to the complexity of the model implemented for the nuclear power plant and the consequential impossibility to solve the tasks without assistance it was also made sure that the assistance offered proactively was not perceived to be superfluous and inappropriate.

9.1.2. Participants

The participants for the reactive group were the participants from Chapter 8. The proactive condition included 23 subjects, with 12 female and 11 male subjects having an average age of 21.9 years ($SD = 2.8$).

9.1.3. Questionnaires

The set of questionnaires used for this experiment consisted of our own questionnaire and the AttrakDiff questionnaire. The CSUQ questionnaire was not used for this experiment since we were rather interested in users acceptance of the help system and not the perceived usability of the system. We also did not expect to find differences concerning this scale, since the assistance provided by the help system was absolutely the same in both conditions. As both help systems used for this experiment did not use spoken input or output the SASSI questionnaire was also not included. Regretfully, we also did not assess cognitive load in this experiment. This would have allowed us to evaluate if the proactive behaviour of the agent increases extraneous cognitive load.
9.2. Hypotheses

For objective data we expected no differences in users’ performance solving the given tasks interacting with the reactive or the proactive agent. Also we expected no differences in erroneous entries and number of restarts of tasks between the two conditions. Since we did not find relations between personality traits and objective measures in the agent condition reported in Chapter 8 we also did not expect personality traits to have an influence on the objective results attained in the proactive condition of the given experiment.

Concerning subjective scales we were on the one hand expecting the reactive agent to be subjectively rated better in terms of intrusiveness, distraction, and dominance than the proactive version. On the other hand we expected to find no differences between the two experimental conditions in terms of help behaviour, optimism, tolerance, humanity and competence. Variables we considered not to be affected by proactivity. Concerning the AttrakDiff scales we expected the agent in the reactive condition to be rated better than the proactive agent for all scales of the questionnaire.

9.3. Results

9.3.1. Objective Scales

Means and standard deviations for the objective scales are summarised in Table 9.1

<table>
<thead>
<tr>
<th>Objective Scales</th>
<th>reactive</th>
<th>proactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>time login [sec]</td>
<td>104.52</td>
<td>94.15</td>
</tr>
<tr>
<td>time task 1 [sec]</td>
<td>111.83</td>
<td>94.83</td>
</tr>
<tr>
<td>time task 2 [sec]</td>
<td>97.27</td>
<td>63.23</td>
</tr>
<tr>
<td>time task 3 [sec]</td>
<td>297.43</td>
<td>170.78</td>
</tr>
<tr>
<td>time task 4 [sec]</td>
<td>101.88</td>
<td>80.53</td>
</tr>
<tr>
<td>time task 5 [sec]</td>
<td>155.28</td>
<td>149.03</td>
</tr>
<tr>
<td>time task 6 [sec]</td>
<td>121.4</td>
<td>126.86</td>
</tr>
<tr>
<td>time task 7 [sec]</td>
<td>33.67</td>
<td>39.74</td>
</tr>
<tr>
<td>time overall [sec]</td>
<td>998.06</td>
<td>751.74</td>
</tr>
<tr>
<td>wrong input overall [#]</td>
<td>4.17</td>
<td>2.57</td>
</tr>
<tr>
<td>try and error overall [#]</td>
<td>.57</td>
<td>.39</td>
</tr>
<tr>
<td>restarts overall [#]</td>
<td>.57</td>
<td>.95</td>
</tr>
</tbody>
</table>

Table 9.1.: Means and standard deviations for objective scales.

**Times** Figure 9.1 shows the times needed to complete each task and all tasks combined for both conditions. For none of the times needed for each task the difference became
9. Reactive vs. Proactive Agent

Figure 9.1.: Average times needed to complete the given tasks dependent on condition. Error bars are standard errors of the mean. * \( p < .05 \)

significant, login (\( t(47) = .60, p = .55 \)), task 1 (\( t(46) = .65, p = .52 \)), task 2 (\( t(47) = 1.05, p = .3 \)), task 4 (\( t(47) = 1.19, p = .24 \)), task 5 (\( t(42) = .14, p = .89 \)), task 6 (\( t(40) = .23, p = .82 \)), and task 7 (\( t(40) = .80, p = .43 \)). However, there was a marginal significant difference for task 3 (\( t(36.86) = 1.87, p = .07 \)) and overall (\( t(39) = 1.84, p = .07 \)).

Erroneous Input

The average numbers of wrong inputs provided by participants for each task and all tasks combined along with the numbers of resets is visualized in Figure 9.2. The differences between conditions were marginal significant for task 5 (\( Z = 1.94, p = .05 \)), task 6 (\( Z = 1.94, p = .05 \)), and overall (\( Z = 1.82, p = .07, r = .26 \)). For login (\( Z = 1.33, p = .18 \)), task 1 (\( Z = 1.01, p = .31 \)), task 2 (\( Z = .95, p = .34 \)), task 3 (\( Z = 1.36, p = .17 \)), task 4 (\( Z = .12, p = .9 \)), and task 7 (\( Z = .46, p = .64 \)) there was no significant difference.

Trial and Error Attempts and Resets

Figure 9.3 gives an overview of the average number of trial and error attempts for each task and all tasks combined along with an average of overall resets dependent on condition. For none of the tasks there was a significant difference of trial and error attempts, login (\( \chi^2(1,46) = .004, p = .95 \)), task 1 (\( \chi^2(1,49) = .24, p = .63 \)), task 2 (\( \chi^2(1,49) = .84, p = .36 \)), task 3 (\( \chi^2(1,49) = .84, p = .36 \)), task 4 (\( \chi^2(1,49) = .9, p = .34 \)), task 5 (\( \chi^2(1,48) = .56, p = .45 \)), task 6 (\( \chi^2(1,42) = 2.54, p = .11 \)), task 7 (\( \chi^2(1,42) = .85, p = .36 \)), and overall (\( \chi^2(3,39) = 1.15, p = .77 \)).
9.3. Results

Figure 9.2.: Average amount of wrong input entered for each task and for all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$

Figure 9.3.: Average number of trial and error attempts and resets per task and all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$
Concerning the number of resets there was also no significant difference between conditions ($Z = .44, p = .66$).

### Influence of Personality Traits on Objective Scales

Similar to the lack of correlations between personality traits and objective measures in the reactive condition (as reported in Chapter 8) there were also no correlations between personality traits and objective scales for the proactive condition.

#### 9.3.2. Subjective Scales

**Own Questionnaire**

Table 9.2 and Figure 9.4 summarise the results attained with our own questionnaire.

<table>
<thead>
<tr>
<th></th>
<th>reactive</th>
<th></th>
<th>proactive</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>intrusiveness</td>
<td>2.35</td>
<td>1.26</td>
<td>2.65</td>
<td>1.14</td>
</tr>
<tr>
<td>distraction</td>
<td>2.7</td>
<td>1.33</td>
<td>3.47</td>
<td>1.45</td>
</tr>
<tr>
<td>dominance</td>
<td>2.54</td>
<td>.93</td>
<td>2.93</td>
<td>.95</td>
</tr>
<tr>
<td>humanity</td>
<td>2.41</td>
<td>.99</td>
<td>2.87</td>
<td>.93</td>
</tr>
<tr>
<td>tolerance</td>
<td>5.54</td>
<td>1.28</td>
<td>5.05</td>
<td>1.01</td>
</tr>
<tr>
<td>optimism</td>
<td>4.83</td>
<td>1.4</td>
<td>4.33</td>
<td>.85</td>
</tr>
<tr>
<td>help behaviour</td>
<td>6.25</td>
<td>1.1</td>
<td>5.61</td>
<td>1.6</td>
</tr>
<tr>
<td>competence</td>
<td>5.67</td>
<td>1.12</td>
<td>5.31</td>
<td>.94</td>
</tr>
</tbody>
</table>

Table 9.2: Means and standard deviations for the results attained with our own questionnaire.

The reactive system was rated significantly better for distraction ($t(45) = 1.90, p = .03, d = .55$) and marginal significantly for dominance ($t(45) = 1.42, p = .08$), tolerance ($t(45) = 1.44, p = .08$), optimism ($t(45) = 1.49, p = .07$), and help behaviour ($t(45) = 1.61, p = .06$). For humanity the reactive system was rated significantly worse than the proactive system ($t(45) = 1.65, p = .05$).

The remaining scales intrusiveness ($t(45) = .87, p = .19$) and competence ($t(45) = 1.18, p = .12$) did not show significantly better ratings for the reactive condition.

**AttrakDiff**

Table 9.3 summarises the means and standard deviations for the AttrakDiff scales dependent on condition.
9.3. Results

Figure 9.4.: Average rating of the system for our own questionnaire. For the scales on the left a low rating means positive evaluation, for the scales on the right high ratings are positive. Error bars are standard errors of the mean. * $p < .05$

<table>
<thead>
<tr>
<th></th>
<th>reactive</th>
<th></th>
<th>proactive</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pragmatic quality</td>
<td>5.28</td>
<td>.72</td>
<td>4.94</td>
<td>.65</td>
</tr>
<tr>
<td>hedonic quality - identity</td>
<td>4.82</td>
<td>.73</td>
<td>4.61</td>
<td>.83</td>
</tr>
<tr>
<td>hedonic quality - stimulation</td>
<td>4.33</td>
<td>.7</td>
<td>3.63</td>
<td>.93</td>
</tr>
<tr>
<td>attractiveness</td>
<td>4.95</td>
<td>.66</td>
<td>4.57</td>
<td>.76</td>
</tr>
</tbody>
</table>

Table 9.3.: Means and standard deviations for the results attained with the AttrakDiff questionnaire.

Figure 9.5 shows the results for the AttrakDiff questionnaire for the reactive and proactive conditions. Statistical analysis revealed a significant differences for three of the four AttrakDiff scales, pragmatic quality ($t(45) = 1.68, p = .05, d = .49$), hedonic quality - stimulation ($t(45) = 2.90, p = .003, d = .85$), and attractiveness ($t(45) = 1.84, p = .04, d = .54$). For hedonic quality - identity there was no statistically significant difference ($t(45) = .94, p = .18$).

Influence of Personality Traits on Subjective Scales

The influence of personality traits on objective scales for the reactive condition are the ones reported for the agent condition in Chapter 8.

For the proactive condition high values for openness positively correlated with a high
9. Reactive vs. Proactive Agent

Figure 9.5.: Average rating of the systems for the AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.

rating for help behaviour \( r(23) = .44, p = .04 \) and optimism \( r(23) = .44, p = .04 \).

For agreeableness high values correlated positively to better ratings for competence \( r(23) = .42, p = .04 \) and pragmatic quality \( r(23) = .45, p = .03 \).

Conscientiousness correlated positively with help behaviour \( r(23) = .52, p = .01 \), tolerance \( r(23) = .49, p = .02 \), and hedonic quality - stimulation \( r(23) = .67, p = .001 \). For intrusiveness there was a negative correlation with conscientiousness \( r(23) = -.47, p = .02 \).

In case of self-efficacy there was only one significant positive correlation with help behaviour \( r(23) = .65, p = .001 \).

9.4. Discussion

9.4.1. Objective Scales

Similar to Xiao et al. [2003] and conforming to our hypothesis we could not find any differences in objective measures. Users amount of time needed for task completion was similar for both groups. Both groups had a similar amount of wrong inputs and restarts. As users were forced to accept help from the help system and tasks were designed not to be solved without assistance the lack of differences is not surprising. Nevertheless, there was a tendency for participants to need less time and provide less erroneous input in the proactive than in the reactive condition. Indicating that a proactive help system could be beneficial and increase software usage efficiency. The differences might not have
been significant due to the design and implementation of the help system in the given experiment.

Influence of Personality Traits on Objective Scales

Again, the lack of correlations between personality traits and objective scales in the proactive condition was quite surprising. However, this finding is conforming to our finding with the help system in the agent condition of Chapter 8 and was thus what we have actually been expecting.

9.4.2. Subjective Scales

Own Questionnaire

Overall there was a clear tendency to rate the proactive agent worse than the reactive one. For most scales of our own questionnaire we could not find significant differences between the two conditions. However, for a lot of scales the difference was marginally significant. For scales where high ratings are positive, means showed that the reactive agent was rated better than the proactive version (except for one scale). Whereas for scales where low ratings are positive, the reactive agent was rated lower, which means better. For help behaviour and competence, subjects rated the help system quite high in the proactive-agent group as well as in the reactive-agent group. As our subjects had to complete predefined tasks, and our agent was always offering help in a semantic way which means he only offered help that was related to the task, we might have provoked a ceiling effect. The scales for optimism and tolerance also revealed only marginal differences, where the agent was rated better in the reactive group than in the proactive one. The text output of the agent was the same for both conditions. It was formulated rather neutral and not reflecting optimism or tolerance. During interaction Dr. Nick was for example not using emotional speech and he was also not trying to be motivating. Thus, these marginal effects might be artifacts of differences for the distraction scale.

While the humanity rating was rather low in both conditions, the proactive version was rated to be significantly more human than the reactive one. This is an indicator that proactive behaviour is perceived to be more human. A similar item appeared in the pragmatic quality scale of the AttrakDiff questionnaire. It contained a bipolar verbal anchor saying “technical – human”. Compared to our scale on humanity the bipolar verbal anchor “technical – human” did not show significant differences between the experimental conditions. Therefore we argue that our scale on humanity is not measuring human behaviour in comparison to technical behaviour. According to the perceived competence of the help system no differences could be found between experimental groups. As the agent did not provide different information in both groups and did not give any wrong information this result is not surprising. For the scales intrusiveness and dominance we expected significant differences between experimental conditions. Conforming to people’s negative responses when they were asked about Clippy [Xiao et al., 2004], the proactive agent was expected to be rated more intrusive, and more dominant than the reactive agent. While the scale on intrusiveness and dominance did not reveal any significant
9. Reactive vs. Proactive Agent

differences, the scale measuring distraction did. A reason for similar ratings for intru-
siveness and dominance in both conditions could be that in the proactive condition the
agent was always activated at a well defined point in time, namely when the user had
finished reading the task or error description. Thus, users were never actually interrupted
by the agent. Even though users did not judge the proactive agent to be more intrusive,
they nevertheless judged it to be more distracting. This could be induced by the proac-
tive agent being activated right after the user had finished reading the task description.
During task description the actual content of the user interface was not displayed and af-
ter clicking the ‘read’ button the help system was activate and required users’ attention,
while new elements became visible. Thus, users might have appreciated having some
time to take a look at the new content, before the agent popped up. Nevertheless, only
in 11.2 percent of cases the agent was closed right after it was proactively opened by the
system, indicating that due to the high task complexity users appreciated the help and
preferred finishing the task quickly over having a closer look at the power plant interface
themselves.

AttrakDiff

As expected, we found differences between the two conditions for the AttrakDiff- scales.
We found a significant difference in perceived pragmatic quality, which means potential
to support users’ goals. In the reactive condition the help system was especially perceived
to be less complicated under pragmatical aspects. The remaining pragmatically items did
not differ significantly, but means show descriptively that the reactive agent nevertheless
is consistently rated better than the proactive one.

Hedonic quality – identity showed no significant differences for the experimental groups
and none of the items contributing to this scale did either.

The scale on hedonic quality – stimulation also showed significant differences between
the two conditions. Again, only a few items can be seen as reason for the overall sig-
nificant difference, namely unimaginative – creative, dull – captivating, undemanding –
challenging, and ordinary – novel. These findings are very difficult to interpret and we
think there are some further studies needed to explain them in a proper way. Especially
differences in creativity are surprising. Perhaps the proactive agent, that was always
performing the same action (user indicated by clicking a button that he had read the
instruction, proactive agent popped up), was rated to be rather uncreative and therefore
acting very standardized, and dull. This would also make him quite ordinary.

Last but not least we found differences in perceived attractiveness, which measures,
in our case, how attractive the agent appears to the user. Only one item of the whole
scale showed significant differences, but for each item the rating for the reactive agent
was better than for the proactive agent. Therefore a reactive agent is more attractive to
the user than a proactive agent.
Influence of Personality Traits on Subjective Scales

The effects of personality traits on the subjective rating of the system for the proactive condition were quite different to those found for the reactive condition. High values for neuroticism did not reveal to have a negative influence on the subjective rating of the system and high values for extraversion did not have a positive influence. For the reactive condition we argued that these personality traits had an influence due to the social nature of the interaction with the agent based system and that users might feel monitored by the system. For a system that is perceived to be more human like than a system rated less human like, as is the case for the proactive and reactive system, where the proactive system was rated higher on our humanity scale, this influence should even be bigger and the same correlations should be found in the proactive condition. However, to us it seems humanity scale does not actually measure how human the agent is perceived to be and how social interaction with the system is accordingly. To us it seems like the scale actually measures the opposite and is thus inverted. This can either be due to the scale being designed badly, what should be reflected by a bad internal consistency, or that users actually rate the system to be less human the more social aspects interaction with the system actually has. This means the more human like the system is actually perceived to be the more participants refuse to admit that they perceive the system to be human like.

For the proactive system, this would mean while the rating on the humanity scale was higher, it was actually perceived to be less human like. This would not be surprising since permanently asking if participants needed help, every time a button is clicked, is not a behaviour one would expect from a human. Compared to this the agent in the reactive condition was rather well behaved and only acted when assistance was requested. This quietness could also have added to a possible feeling of being monitored by the system. In human-human interaction a person monitoring and judging you usually is not a person constantly talking to you and trying to force his assistance on you. Accordingly, for the proactive system the monitoring aspect might not have been as important and thus the influence of neuroticism and extraversion disappeared. Nevertheless we consider this not a very convincing theory and it can only be sufficiently verified in later research.

9.5. Conclusion

In contrast to previous studies we were able to show that proactive behaviour has an influence on the subjective rating of animated interface agents. The reactive agent was especially perceived to be less distracting. Additionally, the reactive agent performed significantly better in three out of the four different scales of the AttrakDiff questionnaire. Even though the observed effects were rather small, this is a hint that proactivity might have played an important role in building up the resentments towards Clippy. A reason for the effects to be rather small might be that the agent in this study was kind of “well-behaved”. It only offered help at a well defined point in the interaction process (at the beginning of each task) and for tasks, users could not accomplish otherwise; it only offered relevant alternatives and it did not exhibit idle animations. In these four points the help system used definitely differs from Clippy. Hence, the fact that we nevertheless
attained significant results, confirms our point of view.

The key differences between the present study and previously published work, investigating the proactivity of animated interface agents – especially [Xiao et al., 2003] – was the way the agent was presented and the output modality. While the agent in the present study was displayed along with the actual content of the user interface, the agent of Xiao et al. [2003] was presented in a separate area. This might not be a problem, if the agent uses spoken language to communicate with the user, but if text based output is required a proactive agent making suggestions might just be overlooked. Another aspect of separating the agent from the interface is the problem that interface assisting agents should be able to interact with the content in order to provide adequate help at the interface level.
10. Reactive Agent vs. Deactivateable Proactivity

The previous experiment showed that proactivity can have negative consequences on users’ acceptance of help systems. One aspect of the proactive help system from the last section was that proactivity could not be disabled. The same was true for Clippy. At least having a look at the options dialogue available for the first version of Clippy from *Microsoft Word 97 for Windows* as shown in Figure 10.1 does not prominently present such an option.

Apart from this the options available are also quite messy and it is hard to tell which option controls which part of Clippy’s behaviour. Maybe the “Guess help topics” option turns off proactive behaviour, maybe the “Display alerts” option, or maybe both? What are “high priority tips”? Will Clippy help me by offering me wizards if I check the “Help with wizards” box or will he assist me while I use wizards? What are the effects of, and why should I want to “Reset my tips”? Those are just some questions that can arise in the context of this options dialogue. Overall, the presented options are really confusing and an excellent example of how limited screen space forces developers to shorten labels beyond recognition. In order to understand the options offered by the help system, users would need assistance by the help system. In the end this confusing options dialogue might be a contributor to the negative reception of Clippy that should not be underestimated. At least the author felt kind of frustrated and helpless, being confronted with this dialogue for the first time.

To check if giving users control over the proactive behaviour can mitigate the negative consequences of proactivity we performed an experiment involving a system allowing to switch proactivity on and off, using a prominently displayed, visible (in the sense of Norman [2002]), and easily accessible option to disable the proactivity feature of the system.

10.1. Method

10.1.1. Conditions

The system for the *reactive* condition was the system with the anthropomorphic agent – the *agent* condition – from Chapter 8 and the same system as used for the *reactive* condition of Chapter 9.

The system in the second condition (*choice*) was the same as for the *proactive* condition in Chapter 9, except that there was an option added to the UI of the agent, that allowed to disable and reenable proactivity, everytime the agent activated either by proactivity
10. Reactive Agent vs. Deactivateable Proactivity

![Figure 10.1: The options available to control the behaviour of Clippy in Microsoft Word 97 for Windows.](image)

or by clicking the “Assist me” button below the image of the agent. This option read “Do not make suggestions on your own. I prefer asking, when I’m in need of assistance.” and was displayed as last option in the interactive box below the image of the agent (see Figure 10.2 on the left). In case proactivity was deactivated this option was replaced by an option that could be used to reanimate proactivity (see Figure 10.2 on the right).

10.1.2. Participants

The group making up the reactive condition is the same as for the agent condition in Chapter 8.

In the choice condition there were 11 female and 11 male subjects, making 22 subjects overall. The mean age of participants in this group was 21 years ($SD = 2$).

10.1.3. Questionnaires and Objective Data

The set of questionnaires used for this experiment consisted of our own questionnaire and the AttrakDiff questionnaire. The CSUQ questionnaire was not used for this experiment since we were rather interested in users’ acceptance of the help system and not the perceived usability of the system. We also did not expect to find differences concerning this scale, since the assistance provided by the help system was absolutely the same in both conditions. As both help systems used for this experiment did not use spoken input or output, the SASSI questionnaire was also not included.

To evaluate if participants were aware of the possibility to turn off the proactive behaviour of the system, the questionnaire in the choice condition included a binary question stating “I was not aware that it was possible to turn off the unsolicited offering of assistance by Dr. Nick.” Additionally there were two questions added asking participants if they would have turned off proactivity earlier, in case they realized that this possibility
10.2. Hypotheses

Figure 10.2.: The agent with an option to deactivate proactivity (last option on the left) and to reactivate it after deactivation (last option on the right).

existed earlier. This two questions were combined to a realized late scale (Cronbach’s α = .81).

To evaluate participants strategy concerning turning on and off the proactivity we logged when this feature was used during interaction.

10.2. Hypotheses

Similar to the experiment comparing the proactive and reactive help system from Chapter 9 we did not expect to find differences for the objectives scales between the two conditions for this experiment.

For the subjective ratings we expected that while some of the scales might still exhibit worse ratings for the choice condition than for the reactive condition, there will be much less worse ratings than between the proactive condition from Chapter 9 and the reactive condition.

10.3. Results

10.3.1. Proactivity Strategies

From the 22 subjects in the choice condition 12 participants used the feature to disable proactivity. Among these 12 subjects 2 subjects turned off proactivity and immediately
turned it back on again. Another 2 of the 12 subjects turned it off at the login task and task 1 respectively and turned it back on at task 5 and task 3. One subject turned proactivity off and on again more than once. The remaining 7 subjects of the 12 using the proactivity disabling feature turned proactivity off and kept it disabled during the whole interaction process. 1 of these 7 subjects turned it off at login, 2 at task 1, 2 at task 2, and 2 at task 5. There were no correlations between personality traits and proactivity strategies.

10.3.2. Objective Scales

A summary of the means and standard deviations for the scales evaluating participants performance can be found in Table 10.1.

<table>
<thead>
<tr>
<th></th>
<th>reactive</th>
<th>choice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>time login [sec]</td>
<td>104.52</td>
<td>62.94</td>
</tr>
<tr>
<td>time task 1 [sec]</td>
<td>111.83</td>
<td>88.02</td>
</tr>
<tr>
<td>time task 2 [sec]</td>
<td>97.27</td>
<td>151.21</td>
</tr>
<tr>
<td>time task 3 [sec]</td>
<td>297.43</td>
<td>307.46</td>
</tr>
<tr>
<td>time task 4 [sec]</td>
<td>101.88</td>
<td>74.16</td>
</tr>
<tr>
<td>time task 5 [sec]</td>
<td>155.28</td>
<td>116.87</td>
</tr>
<tr>
<td>time task 6 [sec]</td>
<td>121.4</td>
<td>47.32</td>
</tr>
<tr>
<td>time task 7 [sec]</td>
<td>33.67</td>
<td>13.27</td>
</tr>
<tr>
<td>time overall [sec]</td>
<td>998.06</td>
<td>507.81</td>
</tr>
<tr>
<td>wrong input overall [⁻]</td>
<td>4.17</td>
<td>3.91</td>
</tr>
<tr>
<td>try and error overall [⁻]</td>
<td>.57</td>
<td>.87</td>
</tr>
<tr>
<td>restarts overall [⁻]</td>
<td>.57</td>
<td>.95</td>
</tr>
</tbody>
</table>

Table 10.1.: Means and standard deviations for objective scales.

Times

An overview of the times needed by participants of each group to complete each task and all tasks combined is depicted in Figure 10.3. For the times needed to complete task 1 ($t(32.21) = .99, p = .33$), task 2 ($t(46) = .15, p = .88$), task 3 ($t(46) = 1.08, p = .29$), task 4 ($t(46) = 1.05, p = .3$), task 5 ($t(41) = .07, p = .94$), task 6 ($t(41) = .67, p = .50$), task 7 ($t(41) = 1.07, p = .29$), and overall ($t(40) = .59, p = .56$) there were no significant differences. In case of login ($t(46) = 1.77, p = .08$) there was a marginal significant difference.

Erroneous Input

The average amount of wrong input provided for each task and all tasks combined dependent on condition is shown in Figure 10.4. The differences for all tasks, login ($Z = .94, p = .35$), task 1 ($Z = .20, p = .84$), task 2 ($Z = .86, p = .39$), task 3
Figure 10.3.: Average times needed to complete the given tasks dependent on condition. Error bars are standard errors of the mean. * $p < .05$

Figure 10.4.: Average amount of wrong input entered for each task and for all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$
10. Reactive Agent vs. Deactivateable Proactivity

Figure 10.5.: Average number of trial and error attempts and resets per task and all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$

$(Z = .18, p = .85)$, task 4 $(Z = .33, p = .74)$, task 5 $(Z = 1.17, p = .24)$, task 6 $(Z = .72, p = .47)$, task 7 $(Z = .46, p = .64)$, and overall $(Z = .64, p = .52)$ were not significant between conditions.

**Trial and Error Attempts and Resets**

An overview of the average number of trial and error attempts per task along with all tasks combined dependent on condition is given in Figure 10.5. For none of the scales, login $(\chi^2 (1, 42) = 0.04, p = 0.83)$, task 1 $(\chi^2 (1, 48) = 0.45, p = 0.5)$, task 2 $(\chi^2 (1, 48) = 0.08, p = 0.78)$, task 3 $(\chi^2 (1, 48) = 0.08, p = 0.78)$, task 4 $(\chi^2 (1, 48) = 0.86, p = 0.35)$, task 5 $(\chi^2 (1, 48) = 1.5, p = 0.22)$, task 7 $(\chi^2 (1, 43) = 0.01, p = 0.92)$, and overall $(\chi^2 (3, 37) = 1.63, p = 0.65)$, the differences were significant. For task 6 none of the participants made a trial and error attempt.

For resets there was also no significant difference $(Z = 0.85, p = 0.4)$.

**Influence of Personality Traits on Objective Scales**

There were no correlations between objective scales and personality traits for the system in the proactive condition.

In the choice condition conscientiousness correlated negatively with the number of trial and error attempts $(r(16) = -0.59, p = 0.02)$. 

194
10.3. Results

10.3.3. Subjective Scales

Additional Questions

All of the participants in the choice condition stated that they were aware of the possibility to turn off proactivity. The mean for the realized late scale was 2.74 with a standard deviation of 1.61. The realized late scale did not correlate with the number of the task proactivity was turned off for the first time ($r(9) = .21, p = .6$).

Own Questionnaire

The results attained with our own questionnaire are summarised in Table 10.2 and Figure 10.6.

<table>
<thead>
<tr>
<th></th>
<th>reactive M</th>
<th>reactive SD</th>
<th>choice M</th>
<th>choice SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>intrusiveness</td>
<td>2.35</td>
<td>1.26</td>
<td>2.63</td>
<td>1.23</td>
</tr>
<tr>
<td>distraction</td>
<td>2.7</td>
<td>1.33</td>
<td>3.52</td>
<td>1.53</td>
</tr>
<tr>
<td>dominance</td>
<td>2.54</td>
<td>0.93</td>
<td>2.82</td>
<td>1.01</td>
</tr>
<tr>
<td>humanity</td>
<td>2.41</td>
<td>0.99</td>
<td>3.05</td>
<td>0.9</td>
</tr>
<tr>
<td>tolerance</td>
<td>5.54</td>
<td>1.28</td>
<td>5.13</td>
<td>0.98</td>
</tr>
<tr>
<td>optimism</td>
<td>4.83</td>
<td>1.4</td>
<td>4.77</td>
<td>0.86</td>
</tr>
<tr>
<td>help behaviour</td>
<td>6.25</td>
<td>1.1</td>
<td>6.23</td>
<td>0.75</td>
</tr>
<tr>
<td>competence</td>
<td>5.67</td>
<td>1.12</td>
<td>5.72</td>
<td>0.71</td>
</tr>
</tbody>
</table>

Table 10.2: Means and standard deviations for the results attained with our own questionnaire.

Similar to the proactive condition from the experiment in Chapter 9 the system in the choice condition was rated to be significantly more distracting than the system in the reactive condition ($t(43) = 1.94, p = .03, d = .58$) and the system in the choice condition was also rated significantly better with respect to humanity than the system in the reactive condition ($t(43) = 2.28, p = .01, d = .68$).

For the remaining six scales, intrusiveness ($t(43) = .77, p = .22$), dominance ($t(43) = .96, p = .17$), tolerance ($t(43) = 1.19, p = .12$), optimism ($t(43) = .17, p = .43$), help behaviour ($t(43) = .08, p = .47$), and competence ($t(43) = .17, p = .43$) there were no significant differences.

AttrakDiff

Means and standard deviations for the AttrakDiff scales are summarised in Table 10.3 and Figure 10.7
10. Reactive Agent vs. Deactivateable Proactivity

Figure 10.6.: Average rating of the system for our own questionnaire dependent on condition. For the scales on the left a low rating means positive evaluation, for the scales on the right high ratings are positive. Error bars are standard errors of the mean. * $p < .05$

<table>
<thead>
<tr>
<th></th>
<th>reactive</th>
<th></th>
<th>choice</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>pragmatic quality</td>
<td>5.28</td>
<td>.72</td>
<td>4.97</td>
<td>.78</td>
</tr>
<tr>
<td>hedonic quality – identity</td>
<td>4.82</td>
<td>.73</td>
<td>4.84</td>
<td>.83</td>
</tr>
<tr>
<td>hedonic quality – stimulation</td>
<td>4.33</td>
<td>.7</td>
<td>4.24</td>
<td>.94</td>
</tr>
<tr>
<td>attractiveness</td>
<td>4.95</td>
<td>.66</td>
<td>4.84</td>
<td>.89</td>
</tr>
</tbody>
</table>

Table 10.3.: Means and standard deviations for the results attained with the AttrakDiff questionnaire.

Statistical analysis of the results for the AttrakDiff data showed a marginal significant worse rating for the choice than for the reactive condition for pragmatic quality ($t(43) = 1.38, p = .09$) and no significant difference for the remaining scales, hedonic quality – identity ($t(43) = .07, p = .47$), hedonic quality – stimulation ($t(43) = .34, p = .37$), and attractiveness ($t(43) = .48, p = .32$).

A closer look at single items reveals that the only items with a significantly better rating for reactive than for choice are complicated – simple ($Z = 2.15, p = .02, r = .32$), and ugly – attractive ($Z = 2.48, p = .007, r = .37$). For the technical – human scale the rating for reactive is marginal significantly worse than for choice ($Z = 1.27, p = .1$).
10.4. Discussion

10.4.1. Proactivity Strategies

From the 22 subjects only 7 subjects turned off proactiveness during interaction and left it turned off, while all 22 subjects stated that they were aware of the possibility to turn proactivity off. This shows that the majority of participants appreciated the help system offering help proactively. 5 participants even turned proactiveness back on after turning it off. Not finding any relations between personality traits and proactivity strategies was rather surprising to us. We would for example have expected less open subjects or subjects with high self-efficacy to switch proactiveness off earlier. However, that we did not find relations is consistent with personality traits hardly having an influence on objective results at all.

Influence of Personality Traits on Subjective Scales

The correlations between subjective scales and personality traits in the proactive condition are the ones reported for the proactive condition in Chapter 9.

For the choice condition there was a significant positive correlation between neuroticism and humanity ($r(21) = .45, p = .04$) and hedonic quality – stimulation ($r(21) = .49, p = .03$).

Conscientiousness correlated positively with the rating for distraction ($r(21) = .49, p = .02$), help behaviour ($r(21) = .51, p = .02$), and competence ($r(21) = .52, p = .02$).

Figure 10.7.: Average rating of the systems for the AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.
10. Reactive Agent vs. Deactivatable Proactivity

10.4.2. Objective Scales

Similar to the proactive system and conforming to our hypotheses there were no significant differences between the choice condition and the reactive condition. However, as for the proactive condition there was a tendency to need less time for task completion in the choice condition than in the reactive condition. There was especially a marginal significant difference for the login task. This is indicating that if there is a possible positive effect on efficiency caused by proactivity, this positive effect can be retained with voluntary proactivity. Thus there is no need to force proactive behaviour on users and it is safe to offer a feature to disable proactivity without putting possible positive effects on efficiency at stake.

Influence of Personality Traits on Objective Scales

Similar to all other conditions we could hardly find relations between objective scales and personality traits. The only relation found in the choice condition was participants with a high value of conscientiousness produced less trial and error attempts. While such a relation is quite logical, a dutiful and organized person will be more willing to follow instructions by the system and will try to follow these instructions more thoroughly than a less dutiful person, it is unclear why we did not find such a relation for the other conditions.

10.4.3. Subjective Scales

Concerning the scales with significant differences between conditions for our own questionnaire the same scales that were significantly different comparing the proactive and the reactive system from Chapter 9 also revealed significant differences comparing the system in the reactive and in the choice condition. Those scales were distraction and humanity. This is indicating that despite of the possibility to disable proactivity the system in the choice condition was still perceived to be more distracting than the system in the reactive condition not showing proactive behaviour. While there was an overall tendency to rate the system worse in the proactive condition from Chapter 9 than in the reactive condition for the remaining scales of our own questionnaire, with four out of the remaining six scales showing a marginally significant better rating for the system in the reactive than in the proactive condition, this tendency could not be observed for the choice condition. For the remaining six scales none revealed even marginally significant differences between the rating of the system in the choice condition and the rating of the system in the reactive condition and ratings were quite similar for the systems in both conditions. We consider this a hint that giving users control over proactivity can actually help to mitigate the negative consequences of proactive behaviour in help systems. The results attained with the AttrakDiff questionnaire strongly support this view. None of the major scales of AttrakDiff showed significant differences between the choice and the reactive condition, while there were three scales with significant differences between the proactive and the reactive condition of Chapter 9. The only scale with a marginal significant difference for the choice and the reactive condition was pragmatic quality.
a closer look at single items for this scale it becomes clear that this difference was due to the system in the choice condition to be perceived to be more complicated than the system in the reactive condition. This difference might be caused by the additional option that was introduced to control proactive behaviour. Adding this option to the UI of the help system made the help system more complex and more complicated accordingly than the system in the reactive condition not having this option. The only other item of AttrakDiff revealing a significant difference between conditions was ugly - attractive. This difference might also have been caused by the additional option added to the list of suggestions made by the help system after activation (see Figure 10.2). For this option we decided to prefer clarity over beauty. Hence, the text describing this option is very descriptive and rather lengthy to clearly describe the effects of activating this option. Thus it rendering the text into the box below the depiction of the agent required two lines. This multiline rendering was not very pretty and thus might have caused the system in the choice condition to be less attractive than the system in the reactive condition not showing such a multiline option every time the help system was activated. It should also be mentioned that the items of AttrakDiff with significant differences between the choice and the reactive condition were all items also showing significant differences between the proactive and the reactive condition.

Influence of Personality Traits on Subjective Scales

Only two personality traits, namely neuroticism and conscientiousness significantly correlated with five scales of the subjective questionnaires. Hence, compared to the five personality traits we found having an influence on the subjective rating of overall eighteen scales for the reactive condition (see Chapter 8) the number of correlations found for the choice condition was rather low.

Similar to the proactive condition of Chapter 9 participants’ extraversion did not show to have significant influence on the rating of the system in the choice condition. Given the higher rating for humanity in the choice condition compared to reactive condition seems to verify our reasoning from Section 9.4.2. There, we stated that the humanity scale, might rather reflect participants’ objection to acknowledge interaction with the system to have human-human like characteristics and thus be higher the less human like the system is actually perceived to be. Explicit binary options like the one offering to stop or reactivate proactive behaviour are typically not part of human-human interaction. While requesting a well intended real person to stop making suggestions for assistance would be considered rather rude, it is absolutely normal to do so interacting with an uncaring computer system. In this context, by giving users further control to influence the behaviour of the help system, the feeling of being monitored by the system might also have been further reduced leading to the rather positive influence of neuroticism on the rating of the help system in the choice condition, compared to the quite negative influence in the reactive condition.
10. Reactive Agent vs. Deactivateable Proactivity

10.5. Conclusion

The results of the performed experiment strongly suggest that giving users control over proactive behaviour can help to mitigate negative consequences concerning the acceptance of a help system proactively offering help, when compared to a help system not showing proactive behaviour. Thus, the subjective negative rating of Clippy might to a certain degree have been prevented offering users a clear and easy to understand option to disable proactivity.

While adding such an option improves acceptance of the help system, our data suggests that possible positive effects on efficiency remain untouched by adding such an option.

Overall it seems like proactivity might be a beneficial option in case of help systems to introduce users to the existence and capabilities of a help system. Negative consequences on the subjective acceptance of a proactive help system can be mitigated by giving users control of the proactive behaviour of the help system.
11. Spoken Out- and Input vs. Text-based Interaction

Baddeley’s model of WM (see Section 2.1.1) suggests that it might be preferable to use spoken output to communicate the instructions given by a help system, since it reduces cognitive load on the visual part of WM. Findings reported by Karl et al. (1993); Shneiderman (2000) state that users find it difficult to speak and think at the same time. Hence, they conclude that speaking interferes with the problem solving capacity of WM. At first this seems to contradict the idea of using spoken language in a help system. However, in case of help systems, speech is not part of the problem but part of the solution. Users use the help system in order to sort out a problem collaboratively with the help system. That way speech is not used concurrent to problem solving but as part of the problem solving process in case of help systems. To keep the problem solving process completely in the auditory part of WM such a help system should also be able to accept spoken input. In case of help systems theoretical advantages are nice but will be in vain in case users avoid or do not enjoy using the system. For help systems subjective user appreciation is of utmost importance. While every user will inevitably need help working with computers, the source the user turns to in order to get this help depends a lot on the users preferences. If a user does not like the help system integrated into the software he will just start to consult an Internet search engine for help or ask a colleague or friend for assistance. Taking this into account our primary goal for the present study was to evaluate users’ subjective acceptance of a speech enabled help system and compare it to a text based system.

Thus initially the experiment was planned to have two conditions. One condition with subjects using the text based help system and one condition with subjects interacting with the system capable of speech input and output. Since it was possible to control the speech enabled system using mouse and keyboard as well, employing speech input was voluntary for users interacting with the speech enabled system. We were expecting some users to be unwilling to try the speech input feature and expected the majority of participants to try the voice control at least once. Spoken output could be disabled in the speech enabled system too. However, since the control to disable and reenable spoken output was not displayed very prominently and the label indicating the functionality of the corresponding control was not very clear (see Section 5.2.5 and Figure 5.27) we did not expect many participants to try this option. Accordingly, we do not consider participants using of this function to be very meaningful concerning participants preference for spoken or textual output.

To encourage usage of spoken input the handout provided to introduce participants to the experiment explicitly pointed out the novelty of the system’s voice control feature and
the parts concerning speech recognition in the handout were stressed using a highlighter. By not making the usage of speech compulsory, we hoped to find differences in the personality traits between users not using speech recognition and those using it. We became aware quite early in the experiment that our assumption was wrong and hardly any of the participants would use speech input. We nevertheless decided to finish the experiment as planned. However, in the end none of the participants in the group with voluntary speech input had used spoken input to interact with the system.

Thus, we spontaneously decided to add a third group to the experiment. This time social pressure on participants in favor of using speech input was raised by the conductor verbally asking the subjects to try the speech recognition feature at least once. And for this group all participants really ended up keeping their promise and tried the speech input modality at least once.

11.1. Method

11.1.1. Conditions

As already mentioned there were three different conditions in this experiment. The text condition was the same as used for the agent condition in Chapter 8.

In the speech input condition participants interacted with a help system using spoken output as output modality and spoken input as input modality. This system was the one described in Section 5.2.5. Participants in this condition were asked to try the spoken input feature of the system at least once.

The system used in the speech output condition was the same as for the speech input condition. The only difference to the speech input condition was that participants were not explicitly asked to try the speech input feature.

11.1.2. Participants

The group for the text condition is the same as the group for the agent condition in Chapter 8.

In the speech output condition there were 21 subjects with a mean age of 21.3 years (SD = 3.5) and 11 subjects being female and 10 subjects male.

For the speech input condition there were 8 female and 13 male subjects, making 21 overall. Mean age of participants in this group was 21.3 years (SD = 2.8).

11.1.3. Procedure

Similar to all other experiments, this study was conducted in a typical seminar room for about 20 to 30 people. For the speech output and speech input conditions computers were equipped with a stereo headset. Due to the social aspect in the two speech conditions, supervisors made sure that there was at least one other participant present, when a subject started interaction with the help system. This means none of the participants started interaction with the system alone in the seminar room.
11.1.4. Questionnaires and Objective Data

Since the data for the reactive condition was originally collected for the experiment of Chapter 8 there are no results for the CSUQ questionnaire available for this condition. Since this condition also did not involve spoken in- our output there is also no SASSI data for this condition.

The questionnaires for the speech output and speech input included the whole set of questionnaires as described in Section 6.4. Hence, we also assessed cognitive load for these two conditions.

In addition to the task completion times, number of erroneous inputs, trial and error attempts and resets, we also logged how often and when the speech input feature of the system was used in order to evaluate participants strategies of using spoken input for the help system.

11.2. Hypotheses

We hypothesized that the in- and output modalities have an impact on the subjective rating of the systems and there are differences between the system using text based output compared to the systems using spoken output. We also expected differences between the condition where users were confronted with spoken output but did not use spoken input and the condition where participants used spoken input as well.

Concerning objective scales we were not sure what to expect, since the results for such comparisons are highly task dependent and studies reported in literature are quite inconsistent [Christian et al., 2000]. Comparing our task to the tasks reported in literature we came to the conclusion that there will not be a significant difference for objective scales between the three conditions.

11.3. Results

11.3.1. Cognitive Load

The average reported overall cognitive load for the speech output condition ($M = 3.14, SD = 1.22$) was not significantly different to the cognitive load reported in the speech input condition ($M = 2.85, SD = .95, t(40) = .87, n. s.$). Only the extraneous cognitive load item revealed a significantly better rating for the speech input condition ($Z = 17.74, p < .001$). Participants in the speech output condition had a mean rank of 50.26, while participants in the speech input condition had an average rank of 44.74. Concerning the remaining scales assessing different aspects of cognitive load there were no significant differences.

11.3.2. Objective Scales

Table 11.1 shows an overview of the means and standard deviations of the objective scales for both conditions.
11. Spoken Out- and Input vs. Text-based Interaction

![Figure 11.1.](image)

Figure 11.1.: Average times needed to complete the given tasks dependent on condition. Error bars are standard errors of the mean. * $p < .05$

<table>
<thead>
<tr>
<th>task</th>
<th>text M</th>
<th>text SD</th>
<th>speech output M</th>
<th>speech output SD</th>
<th>speech input M</th>
<th>speech input SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>time login [sec]</td>
<td>104.52</td>
<td>62.94</td>
<td>120.56</td>
<td>55.6</td>
<td>162.11</td>
<td>109.71</td>
</tr>
<tr>
<td>time task 1 [sec]</td>
<td>111.83</td>
<td>88.02</td>
<td>149.29</td>
<td>142.14</td>
<td>175.68</td>
<td>119.07</td>
</tr>
<tr>
<td>time task 2 [sec]</td>
<td>97.27</td>
<td>151.21</td>
<td>121.06</td>
<td>137.54</td>
<td>74.5</td>
<td>72.41</td>
</tr>
<tr>
<td>time task 3 [sec]</td>
<td>297.43</td>
<td>307.46</td>
<td>269.92</td>
<td>222.26</td>
<td>288.27</td>
<td>291.12</td>
</tr>
<tr>
<td>time task 4 [sec]</td>
<td>101.88</td>
<td>74.16</td>
<td>85.44</td>
<td>37.57</td>
<td>95.27</td>
<td>73.77</td>
</tr>
<tr>
<td>time task 5 [sec]</td>
<td>155.28</td>
<td>116.87</td>
<td>138.97</td>
<td>112.12</td>
<td>91.28</td>
<td>73.77</td>
</tr>
<tr>
<td>time task 6 [sec]</td>
<td>121.4</td>
<td>47.32</td>
<td>112.84</td>
<td>28.11</td>
<td>150.49</td>
<td>111.52</td>
</tr>
<tr>
<td>time task 7 [sec]</td>
<td>33.67</td>
<td>13.27</td>
<td>36.23</td>
<td>31.29</td>
<td>50.82</td>
<td>72.66</td>
</tr>
<tr>
<td>time overall [sec]</td>
<td>998.06</td>
<td>507.81</td>
<td>1034.3</td>
<td>366.64</td>
<td>1088.41</td>
<td>362.63</td>
</tr>
<tr>
<td>wrong input overall [#]</td>
<td>4.17</td>
<td>3.91</td>
<td>4.98</td>
<td>2.97</td>
<td>3.58</td>
<td>4.53</td>
</tr>
<tr>
<td>try and error overall [#]</td>
<td>.57</td>
<td>.87</td>
<td>.92</td>
<td>1.0</td>
<td>.5</td>
<td>.63</td>
</tr>
<tr>
<td>restarts overall [#]</td>
<td>.57</td>
<td>.95</td>
<td>1.14</td>
<td>1.59</td>
<td>1.67</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 11.1.: Means and standard deviations for objective scales.

The time needed by participants to complete each task and all tasks combined is shown in Figure 11.1.

Analysis of differences in the times needed to complete the tasks showed a significant difference for login ($\chi^2(2, 68) = 7.84, p = .02$) and a marginal significant difference for
11.3. Results

Figure 11.2.: Average amount of wrong input entered for each task and for all tasks combined dependent on condition. Error bars are standard errors of the mean. * \( p < .05 \)

task 5 \( (\chi^2(2,65) = 5.45, p = .07) \) between the three conditions. For the remaining measures, task 1 \( (F(2,64) = 1.74, p = .18) \), task 2 \( (F(2,65) = .70, p = .5) \), task 3 \( (F(2,65) = .06, p = .94) \), task 4 \( (F(2,65) = .37, p = .69) \), task 6 \( (\chi^2(2,65) = .79, p = .67) \), task 7 \( (F(2,62) = .88, p = .42) \), and overall \( (F(2,61) = .25, p = .78) \) there were no significant differences.

Between the two speech conditions \( (speech \ output, \ speech \ input) \) and the text condition there was a significant difference for the time needed for login \( (t(54.64) = 2.02, p = .05, r_{\text{contrast}} = .26) \) and a marginal significant difference for task 1 \( (t(64) = 1.72, p = .09, r_{\text{contrast}} = .21) \). Between both speech conditions \( (speech \ output, \ speech \ input) \) there was a marginal significant difference for task 5 \( (t(23.37) = 1.87, p = .07) \).

Erroneous Input

Average number of wrong inputs provided for each of the tasks and for all tasks combined depending on condition is shown in Figure 11.2. Analyzing the data for differences between conditions yielded no significant results for login \( (\chi^2(2,60) = 1.33, p = .52) \), task 1 \( (\chi^2(2,51) = .31, p = .86) \), task 2 \( (\chi^2(2,50) = 4.52, p = .1) \), task 3 \( (\chi^2(2,49) = .58, p = .75) \), task 4 \( (\chi^2(2,54) = 1.10, p = .58) \), task 5 \( (\chi^2(2,55) = 4.24, p = .12) \), task 6 \( (\chi^2(2,54) = .50, p = .78) \), task 7 \( (\chi^2(2,62) = 1.80, p = .41) \), and overall \( (\chi^2(2,68) = 4.47, p = .11) \).
11. Spoken Out- and Input vs. Text-based Interaction

Figure 11.3.: Average number of trial and error attempts and resets per task and all tasks combined dependent on condition. Error bars are standard errors of the mean. * $p < .05$

**Trial and Error Attempts and Resets**

An overview of the average number of trial and error attempts along with the resets made for each condition is shown in Figure 11.3.

None of the differences between conditions for the tasks login ($\chi^2(2, 64) = .59, p = .74$), task 1 ($\chi^2(2, 57) = .84, p = .66$), task 2 ($\chi^2(2, 57) = 2.68, p = .26$), task 3 ($\chi^2(2, 58) = 3.68, p = .16$), task 4 ($\chi^2(2, 57) = .33, p = .85$), task 5 ($\chi^2(2, 57) = 1.53, p = .47$), task 7 ($\chi^2(2, 64) = 1.02, p = .6$), and overall ($\chi^2(6, 49) = 3.12, p = .79$) were significant concerning trial and error attempts. For task 6 there were no trial and error attempts for all three conditions.

With respect to the resets there was a significant difference between the three conditions ($\chi^2(2, 65) = 7.07, p = .03$) and post-hoc testing showed that there is a difference between the text condition and speech input condition ($Z = 2.76, p = .003, r = .42$).

**Influence of Personality Traits on Objective Scales**

In the speech output condition self-efficacy negatively correlated with the time needed to complete all tasks ($r(21) = -.43, p = .05$) and in the speech input condition high values for self-efficacy correlated with a high number of trial and error attempts ($r(16) = .54, p = .03$). Additionally, in the speech input condition neuroticism correlated negatively with a high number of trial and error attempts ($r(16) = -.75, p = .001$) and resets ($r(21) = -.45, p = .04$).
11.3.3. Speech Input Strategies

Evaluating the strategies used for spoken input in the speech input condition revealed three remarkably distinctive usage patterns:

**marginal users** (n=2) Subjects in this group used the speech recognition during one task and stopped using it afterwards. Hence, the percentage of using speech for interaction in this group was below 10%. One of the two participants used it for task 3 and the second participant for the login task.

**casual users** (n=9) Participants in this group used speech to communicate with the system whenever it was convenient. The percentage of speech usage within this group was more than 45% and less than 80%.

**power users** (n=10) Subjects falling into this group virtually always used speech input to interact with the system. The percentage of spoken input compared to mouse based input was more than 95%.

The average values for personality traits separated for members of each group are listed in Table 11.2.

<table>
<thead>
<tr>
<th></th>
<th>marginal users</th>
<th>casual users</th>
<th>power users</th>
</tr>
</thead>
<tbody>
<tr>
<td>neuroticism</td>
<td>3.11 ± .31</td>
<td>2.94 ± 1.26</td>
<td>3.79 ± 1.01</td>
</tr>
<tr>
<td>extraversion</td>
<td>4.78 ± .63</td>
<td>5.06 ± .49</td>
<td>4.48 ± .5</td>
</tr>
<tr>
<td>openness</td>
<td>4.06 ± 1.49</td>
<td>5.25 ± .74</td>
<td>4.62 ± .72</td>
</tr>
<tr>
<td>agreeableness</td>
<td>5.56 ± .63</td>
<td>5.21 ± 1.04</td>
<td>4.92 ± .8</td>
</tr>
<tr>
<td>conscientiousness</td>
<td>5.72 ± .55</td>
<td>5.19 ± .86</td>
<td>4.92 ± .69</td>
</tr>
<tr>
<td>self-efficacy</td>
<td>30.0 ± 0.0</td>
<td>32.22 ± 2.05</td>
<td>29.0 ± 2.94</td>
</tr>
</tbody>
</table>

Table 11.2.: Means and standard deviations for the big five personality traits and self-efficacy for the three different groups of speech input users.

Due to the small number of samples in the marginal users group this group was excluded from further analysis and only the casual users and power users groups are compared in Figure 11.4.

For extraversion ($t(17) = 2.56, p = .02, d = .86$) and self-efficacy ($t(17) = 2.74, p = .01, d = .39$) there were significant differences and for openness ($t(17) = 1.86, p = .08$) a marginal significant difference between subjects falling into the casual users and power users group. For neuroticism ($t(17) = 1.63, n. s.$), agreeableness ($t(17) = .68, n. s.$) and conscientiousness ($t(17) = .74, n. s.$) there were no significant differences.

11.3.4. Subjective Scales

 Own Questionnaire

Means and standard deviations for the results attained with our own questionnaire are summarised in Table 11.3.
11. Spoken Out- and Input vs. Text-based Interaction

Figure 11.4.: Comparison of average values for personality traits for subjects from the casual users and power users groups. Error bars are standard errors of the mean. * $p < .05$

<table>
<thead>
<tr>
<th>Trait</th>
<th>Text M</th>
<th>Text SD</th>
<th>Speech output M</th>
<th>Speech output SD</th>
<th>Speech input M</th>
<th>Speech input SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrusiveness</td>
<td>2.35</td>
<td>1.26</td>
<td>2.03</td>
<td>1.07</td>
<td>2.3</td>
<td>1.14</td>
</tr>
<tr>
<td>Distraction</td>
<td>2.7</td>
<td>1.33</td>
<td>3.39</td>
<td>1.7</td>
<td>2.75</td>
<td>1.44</td>
</tr>
<tr>
<td>Dominance</td>
<td>2.54</td>
<td>.93</td>
<td>2.52</td>
<td>1.09</td>
<td>3.01</td>
<td>1.23</td>
</tr>
<tr>
<td>Humanity</td>
<td>2.41</td>
<td>.99</td>
<td>2.7</td>
<td>1.23</td>
<td>2.77</td>
<td>1.0</td>
</tr>
<tr>
<td>Tolerance</td>
<td>5.54</td>
<td>1.28</td>
<td>5.21</td>
<td>1.06</td>
<td>4.88</td>
<td>1.23</td>
</tr>
<tr>
<td>Optimism</td>
<td>4.83</td>
<td>1.4</td>
<td>4.01</td>
<td>1.71</td>
<td>4.14</td>
<td>1.6</td>
</tr>
<tr>
<td>Help behaviour</td>
<td>6.25</td>
<td>1.1</td>
<td>6.11</td>
<td>.77</td>
<td>6.23</td>
<td>.87</td>
</tr>
<tr>
<td>Competence</td>
<td>5.67</td>
<td>1.12</td>
<td>5.42</td>
<td>.77</td>
<td>5.26</td>
<td>1.29</td>
</tr>
</tbody>
</table>

Table 11.3.: Means and standard deviations for the results attained with our own questionnaire.

Figure 11.5 visualizes the results attained with our own questionnaire dependent on condition.

For none of the scales, intrusiveness ($F(2, 63) = .46, p = .63$), distraction ($F(2, 63) = 1.46, p = .24$), dominance ($F(2, 63) = 1.39, p = .26$), humanity ($F(2, 63) = .75, p = .48$), tolerance ($F(2, 63) = 1.70, p = .19$), optimism ($F(2, 63) = 1.81, p = .17$), help behaviour ($F(2, 63) = .15, p = .86$), and competence ($F(2, 63) = .83, p = .44$) the differences were significant.

The only planned contrast showing a marginal significant difference between the text
11.3. Results

Figure 11.5.: Average rating of the system for our own questionnaire dependent on condition. For the scales on the left a low rating means positive evaluation, for the scales on the right high ratings are positive. Error bars are standard errors of the mean. * $p < .05$

condition and the two speech conditions (speech output, speech input) is for the optimism scale ($t(63) = 1.88, p = .06$).

CSUQ

The results attained with the CSUQ questionnaire dependent on condition are summarised in Table 11.4 and Figure 11.6

<table>
<thead>
<tr>
<th></th>
<th>speech output</th>
<th>speech input</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
</tr>
<tr>
<td>sysuse</td>
<td>5.57</td>
<td>1.23</td>
</tr>
<tr>
<td>infoqual</td>
<td>5.88</td>
<td>.98</td>
</tr>
<tr>
<td>interqual</td>
<td>5.27</td>
<td>1.01</td>
</tr>
<tr>
<td>overall</td>
<td>5.64</td>
<td>.99</td>
</tr>
</tbody>
</table>

Table 11.4: Means and standard deviations for the results attained with the CSUQ questionnaire.

For none of the CSUQ scales, sysuse ($t(40) = .87, p = .39$), infoqual ($t(40) = .82, p = .42$), interqual ($t(40) = .22, p = .82$), nor overall ($t(40) = .8, p = .43$) there were significant differences.
11. Spoken Out- and Input vs. Text-based Interaction

Figure 11.6.: Average rating of the systems for the CSUQ questionnaire dependent on condition. Error bars are standard errors of the mean.

SASSI

Table 11.5 and Figure 11.7 show a summary of the results attained for the speech output and speech input condition for the SASSI questionnaire.

<table>
<thead>
<tr>
<th></th>
<th>speech output</th>
<th>speech input</th>
</tr>
</thead>
<tbody>
<tr>
<td>response accuracy</td>
<td>5.24</td>
<td>5.44</td>
</tr>
<tr>
<td>habitability</td>
<td>4.6</td>
<td>5.28</td>
</tr>
<tr>
<td>speed</td>
<td>5.42</td>
<td>5.47</td>
</tr>
</tbody>
</table>

Table 11.5.: Means and standard deviations for the results attained with the SASSI questionnaire.

There were no statistically significant differences for the evaluated scales, response accuracy \((t(38) = .72, p = .48)\), habitability \((t(38) = 1.64, p = .10)\), and speed \((t(38) = .11, p = .9)\) of the SASSI questionnaire dependent on condition.

AttrakDiff

Table 11.6 shows a summary of the means and standard deviations for the four AttrakDiff scales and Figure 11.8 visualizes the results for all word pairs.
11.3. Results

![Graph showing average rating of systems for SASSI questionnaire dependent on condition](image)

Figure 11.7.: Average rating of the systems for the SASSI questionnaire dependent on condition. Error bars are standard errors of the mean.

<table>
<thead>
<tr>
<th></th>
<th>text M</th>
<th>text SD</th>
<th>speech output M</th>
<th>speech output SD</th>
<th>speech input M</th>
<th>speech input SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>pragmatic quality</td>
<td>5.28</td>
<td>.72</td>
<td>5.19</td>
<td>.57</td>
<td>5.01</td>
<td>.87</td>
</tr>
<tr>
<td>hedonic quality – identity</td>
<td>4.82</td>
<td>.73</td>
<td>4.62</td>
<td>.82</td>
<td>5.0</td>
<td>.77</td>
</tr>
<tr>
<td>hedonic quality – stimulation</td>
<td>4.33</td>
<td>.7</td>
<td>3.66</td>
<td>.86</td>
<td>4.06</td>
<td>.91</td>
</tr>
<tr>
<td>attractiveness</td>
<td>4.95</td>
<td>.66</td>
<td>4.61</td>
<td>.88</td>
<td>4.63</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 11.6.: Means and standard deviations for the results attained with the AttrakDiff questionnaire.

Only one of the four AttrakDiff scales, hedonic quality – stimulation ($F(2, 63) = 3.74, p = .03, \eta^2 = .11$) revealed a significant difference between conditions. For the remaining scales, pragmatic quality ($F(2, 63) = .76, p = .47$), hedonic quality – identity ($F(2, 63) = 1.28, p = .28$), and attractiveness there were no significant differences.

Accordingly, for the planned contrasts there was only one scale with a different rating between the speech conditions (speech output, speech input) and the text condition, this scale was hedonic quality – stimulation ($t(63) = 2.23, p = .03, r_{\text{contrast}} = .27$).

For the second planned contrast verifying the expected difference between speech output and speech input, none of the scales pragmatic quality ($t(63) = .78, p = .44$), hedonic quality – identity ($t(63) = 1.6, p = .11$), hedonic quality – stimulation ($t(63) = 1.59, p = .12$), and attractiveness ($t(63) = .08, p = .94$) revealed significant differences between conditions.
11. Spoken Out- and Input vs. Text-based Interaction

![Graph showing average rating of systems for AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.](image)

**Figure 11.8.:** Average rating of the systems for the AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.

**Influence of Personality Traits on Subjective Scales**

For the *speech output* condition there was a negative correlation between *openness* and *humanity* from our own questionnaire ($r(21) = -0.45, p = 0.04$). *Agreeableness* correlated positively with *hedonic quality - stimulation* ($r(21) = 0.45, p = 0.04$).

In the *speech input* condition *neuroticism* correlated positively with *help behaviour* ($r(21) = 0.5, p = 0.02$) and *competence* ($r(21) = 0.46, p = 0.04$). *Conscientiousness* correlated negatively with *distraction* ($r(21) = -0.53, p = 0.01$), and positively with the *speed* scale from the SASSI questionnaire ($r(20) = 0.58, p = 0.007$).

11.4. Discussion

**11.4.1. Cognitive Load**

While we did not find a significant difference in the overall cognitive load reported for the *speech output* and *speech input* conditions, we found a difference for extraneous cognitive load between the two conditions. Even though this scale consisted of one item only, we consider this an indicator that using spoken out- and input compared to exclusively using spoken output can reduce cognitive load on users and hence support learning in the context of help systems.
11.4. Discussion

11.4.2. Objective Scales

Conforming to our hypothesis the in- and output modality did not have a significant influence on the overall task performance for the particular kind of task used in the experiment.

There were no differences for the completion times of separate tasks except for the initial login task. There are two possible reasons for this difference. Firstly, the unconventional spoken system might have taken more time to accommodate to. Secondly, the rather complicated password used for login was also provided by the help system. In the text based scenario this password was written as text in the speech bubble. This password could easily be read and even copied and pasted from the bubble into the password field. In the speech conditions the password was spelled letter by letter. Thus, the spoken output version was definitely more time consuming than the text based version. This circumstance contributed to the difference. Concerning the amount of wrong input entered by participants and number of trial error attempts there were no significant differences between conditions at all. Only for the number of resets there was a significant difference between the speech input and the text condition, with the speech output condition lying between the other two conditions. This difference might be caused by two outliers in the speech input condition that were responsible for 12 of the 35 resets of the whole group. Resets are an indicator that participants had problems finishing certain tasks. Those problems usually arose when participants selected a wrong step by step tutorial for the given task. Due to the design of the experiment this rarely happened, since the appropriate step by step guide assisting to accomplish the current task was always the first one offered by the help system, it occasionally happened. While in the text condition participants experiencing problems finishing tasks were likely to abort interaction – 3 of the 26 participants in the text condition did not complete all tasks – in the speech output and speech input conditions all 21 subjects finished all tasks. The number of resets for participants not finishing all tasks was excluded from analysis, since we wanted to compare the average number of resets to complete all tasks between conditions. Hence, the difference between the conditions might rather reflect a difference in involvement between conditions and participants in the speech input condition did not abort interaction even if they experienced problems. Removal of the outliers from both the speech output and speech input causes the significant difference to disappear.

11.4.3. Speech Input Strategies

An interesting finding of the performed experiment are the three different strategies for using spoken input in the speech input condition. The three identified groups are remarkably well distinguishable. The smallest group consisting of only 2 participants refused to use the spoken input beyond giving it a try as asked to do by the supervisor. The second group consisting of 9 subjects used both spoken as well as mouse input to control the help system and the third group consisting of 10 participants more or less exclusively used spoken input to interact with the help system. Since participants were explicitly asked by the supervisor to try the speech input system at least once, this difference in usage
patterns can either be attributed to participants’ actual preferences or subjects’ desire to act socially desirable by fulfilling the supervisor’s request. If participants in the power users group were using the speech input feature so extensively because they considered this behaviour to be socially desirable – as indicated by the supervisors’ request – there should be differences for the agreeableness scale between the power users and marginal users groups. However, with respect to agreeableness there were no significant differences between the two groups ($t(10) = 1, p = .32$) and agreeableness was even higher in the marginal users group. Of course the marginal users group was quite small consisting of two participants only, what makes it hard to find significant differences between the two groups. However, if agreeableness had an influence on the usage pattern of spoken input for our experiment, than there should also be a difference between the casual users and power users groups. Since there was also no difference between those two groups concerning agreeableness, we argue that the difference in usage patterns was actually due to participants preferences using spoken input. Accordingly, our data suggests that there are three groups of users when it comes to using spoken input to interact with computer systems. One group rejecting the idea of using spoken input, another group using spoken input whenever it seems convenient to them and a third group exclusively relying on spoken input. For the power users and casual users groups we found differences in personality traits related to these usage patterns. Participants in the casual users group had higher values for extraversion and self-efficacy. Additionally, there was also a marginally significant difference between the two groups concerning openness with openness in the casual users group being higher than in the power users group. Thus, energetic users that are open to new things and are quite sure of their own skills are more likely to use speech as a new, alternative input modality in a way that seems most appropriate to them and integrate it into their normal workflow. Users that are more introvert, and comparably less certain of their own skills tend to exclusively rely on one input modality and exclusively use the spoken input feature of the system.

11.4.4. Subjective Scales

Contradicting our initial hypothesis there were hardly differences between the subjective scales evaluated for this experiment.

Own Questionnaire

Our own questionnaire only revealed one marginally significant difference between the two conditions involving spoken output and the text based system. This difference was for the optimism scale. This difference might be caused by the voice and prosody of the spoken output. The output was rendered in a typical uninflected TTS voice and thus did not exhibit personal features like optimism.

AttrakDiff

Contradicting our initial hypothesis there were no differences for three out of the four AttrakDiff scales. Indicating that the in- and output modalities might not have such
11.5. Conclusion

a big effect on the overall evaluation of the system. The slight preference for the text-based system might be due to aspects specific to the help system used in the given experiment and might not be applicable in general. The bad rating for the technical – human category for example might be caused by combining a very human-like soft voice with the comical representation of the agent. This odd combination might have pushed the agent into the uncanny valley (see Section 2.3).

The different rating for "unruly-manageable" might be due to the fact that in the speech conditions users could not easily control, when the agent spoke. Unlike to human-human interaction where communication partners use subtle hints and backchannels to direct the dialogue this obviously was not possible during interaction with the agent. If users for example wanted to pause the speech output they had to explicitly click one of the controls. The fact that the audio playback of the agent could only be controlled by visual components was understandably more serious for participants using spoken input with the agent. Since, those users were less used to switching their visual focus of attention from the problem at hand to interaction with the agent. Hence participants in the speech input group rated the agent even worse with respect to the unruly-manageable item. This difference between verbal human-human and human-computer interaction might also have been the reason, why interaction with the speech output modality system was considered to be more challenging. And, since people that ignore subtle hints during conversations and keep on talking are perceived to be less likeable, this might also be the reason for the perceived lower likeability of the agent in the speech enabled conditions.

CSUQ and SASSI

The results for the comparison between the group using speech output and mouse control for the agent and the group using speech input is quite clear. While there was a tendency in favor of the group using spoken input, none of the CSUQ or SASSI scales showed significant differences between the two conditions. Thus, the initial hypothesis of the input modality having influence on the overall subjective evaluation system is to be rejected, at least when the user is given the decision on the input modality of choice.

11.5. Conclusion

We were able to show that users are still quite reluctant to use voice control features of software, especially in public. As reasons for their unwillingness to use the speech feature participants stated that it would feel weird to speak to the computer in front of others.

We could also show that when participants were mildly pushed into using speech recognition most of them tend to give up their initial resistance and use the feature quite frequently. This gives rise to the hope that users' reluctance to use voice controlled software might be a temporary phenomenon and change over time.

Different to our hypothesis the data obtained reveals that the output- and input-modality chosen for help systems has limited influence on the overall user satisfaction with help systems. While there was a tendency to prefer the text-based over spoken interaction there is reasonable doubt that this preference might be due to some limitations specific
to the help system used for the given experiment and not a problem of speech based help systems in general. Taking this into account implementing a speech enabled help system that will be accepted by users seems feasible. Considering the theoretical benefits of such a help system this also seems to be desirable.

Furthermore our data suggests that there might be three distinguished groups of users exhibiting different patterns of employing speech input when confronted with a system capable to handle spoken as well as traditional mouse input. One group avoids using spoken input and prefers using mouse based interaction, a second group uses both input modalities and a third group almost exclusively uses spoken input. Our data also indicates that personality traits have an influence on the pattern exhibited by particular users.
12. Animated Agents as Social Actors

The ratings for humanity from our own questionnaire and the technical - human item from AttrakDiff consistently got the worst ratings within their respective questionnaires as soon as the help system involved an anthropomorphic agent. Thus users were strongly neglecting the system to have any human like characteristics. Accordingly users typically neglect that they perceive interaction with computers to be social or similar to interaction with humans. [Nass and Moon, 2000]. Nevertheless, for our first experiments we found that for the two options offered by the agent when a step by step guide was finished in 93% of cases the positive option was chosen by participants. Thus, users did not use the derogatory “Pff, I could have done that on my own!” message to close the help system at the end of a step by step tutorial, but preferred using the positive “Thanks!” option (see Figure 12.1).

We considered this to be a clear indicator that participants behaved socially towards the help system. However, since the positive option was always presented first for our initial experiments and it was also the option with less text, there was still the possibility that participants were just clicking the shorter first option, without caring about the semantic meaning of this option. To compensate for a possible bias towards the first or the shorter option, we decided to randomize the position of the options as well as the text message used for the options. We introduced a longer text message saying “Thank you for your help!” for the positive closing option and a shorter version containing only the “Pff!” part of the negative message to the system. Short and long version of each option were chosen randomly at the end of every guide and the sequence of options was also randomized. The results attained with this configuration were the same as for the non-randomized option and for all experiments combined participants chose the positive option in 1066 of cases and the negative option for 66 cases. This means the positive option was chosen for an impressive 94% of cases and participants doubtlessly showed social behaviour towards the help system.

The finding that the interaction between humans and computers is influenced by social behavioural patterns is not new. Ground breaking studies in this area were performed by Nass et al. [1994]. Among other things they were able to show that “users asked by a computer about its own performance will feel compelled to be more positive than will users asked about the computer by an independent source.” [Nass et al., 1994] Overall they stated that computers should be regarded as social actors. The proposed framework has been termed as Computers As Social Actors (CASA) and could be verified by various studies (see Mishra [2006] for an overview).

Following Langer [1992], Nass and Moon [2000] attributed this inappropriate social behaviour towards computers to be caused by the mindless application of simplistic social scripts learned from human-human interaction to human-computer interaction.
12. Animated Agents as Social Actors

Figure 12.1.: The help system at the end of step by step guides with the positive and negative option to close the system.

To evaluate the influence of anthropomorphic agents on the findings of the CASA framework we performed an experiment [Lang et al., 2013c] based on the findings of Nass et al. [1994]. We were especially interested in the effects on the evaluation of the agent, when the feedback was assessed directly by the agent, indirectly but on the same computer by means of a web-based form or indirectly from an independent source, namely a pen and paper questionnaire.

12.1. Method

12.1.1. Conditions

In the agent condition evaluating questions were asked by an online system implemented specifically for this purpose. This system used the representation of the agent to ask the questions of the questionnaires. Figure 12.2 shows an example for the way questions were presented to participants in this condition.

This system was accessed using the same computer and browser as was used for the interaction with the system.

The second group in the online condition used an online questionnaire created with LimeSurvey [Schmitz, 2012]. The questions were presented to participants on the same computer and the same browser as used during the interaction with the system. Figure 12.3 shows an example for how the questions were presented to participants in this condition.
Figure 12.2: Example questions asked directly by the agent in the agent condition.
12. Animated Agents as Social Actors

Figure 12.3.: Typical page of a questionnaire created with LimeSurvey [Schmitz, 2012] as used in the online condition

The last group of participants in the pen and paper condition completed a printed version of the questionnaires after interaction. To assess the personal data before interaction, the same online questionnaires as for all other experiments were used.

12.1.2. Participants

The agent condition consisted of 21 participants having an average of 22.2 (SD = 2.1) and 10 subjects being female and 11 male.

Participants in the online condition were the same as participants in the proactive condition of Chapter 9.

In the pen and paper condition there were 15 participants, 8 female and 7 male. The mean age of participants in this group was 23.7 (SD = 2.7).

Questionnaires and Objective Data

As all participants interacted with the same system, we decided to exclude the CSUQ questionnaire from the questionnaires assessed after interaction, since this questionnaire is rather unemotional and focuses on the usability of the system. Hence, we did not expect it to yield different results between conditions.

The system used in this experiment also did not involve spoken in- or output and thus the SASSI questionnaire was also not part of the questionnaires after interaction.
Since all participants interacted with exactly the same system, evaluating for differences concerning the performance of subjects between conditions would not have yielded meaningful results. Thus, we did not gather objective data for this experiment.

12.2. Hypotheses

Adhering to the CASA hypothesis [Nass et al., 1994, 1999; Nass and Moon, 2000] we expected the modality of the questionnaire to have an influence on the subjective rating of the help system. Furthermore we expected the system to be rated best in the agent condition and worst in the pen and paper condition. The online condition was expected to lie somewhere between the agent and pen and paper condition.

As we consider online forms to be the standard means to gather subjective evaluation data in computer related research we were especially interested in differences between the agent and the online condition and our second planned contrast was dedicated to evaluate these differences. In this context, we expected the system in the agent condition to be rated significantly better than in the online condition.

12.3. Results

Mean and standard deviations for the results attained with our own questionnaire are summarised in Table 12.1.

12.3.1. Own Questionnaire

<table>
<thead>
<tr>
<th></th>
<th>pen and paper</th>
<th>online</th>
<th>agent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M</td>
<td>SD</td>
<td>M</td>
</tr>
<tr>
<td>intrusiveness</td>
<td>3.0</td>
<td>1.17</td>
<td>2.65</td>
</tr>
<tr>
<td>distraction</td>
<td>4.56</td>
<td>1.1</td>
<td>3.47</td>
</tr>
<tr>
<td>dominance</td>
<td>3.11</td>
<td>1.07</td>
<td>2.93</td>
</tr>
<tr>
<td>humanity</td>
<td>3.28 .72</td>
<td>2.87</td>
<td>.93</td>
</tr>
<tr>
<td>tolerance</td>
<td>5.7</td>
<td>1.04</td>
<td>5.05</td>
</tr>
<tr>
<td>optimism</td>
<td>4.93</td>
<td>.83</td>
<td>4.33</td>
</tr>
<tr>
<td>help behaviour</td>
<td>5.48</td>
<td>1.36</td>
<td>5.61</td>
</tr>
<tr>
<td>competence</td>
<td>5.44</td>
<td>1.17</td>
<td>5.31</td>
</tr>
</tbody>
</table>

Table 12.1.: Means and standard deviations for the results attained with our own questionnaire.

Figure 12.4 shows the average rating for our system dependent on condition.

There was a significant difference between the three conditions for intrusiveness \((F(2,56) = 5.34, p = .008, \eta^2 = .16)\), distraction \((F(2,56) = 6.35, p = .003, \eta^2 = .18)\), and optimism \((F(2,56) = 3.34, p = .04, \eta^2 = .11)\). For tolerance \((F(2,56) = 3.10, p = .05)\) the difference was marginal significant and for the remaining four scales dominance \((F(2,56) = 221\).
12. Animated Agents as Social Actors

Figure 12.4.: Average rating of the system dependent on condition for our own questionnaire. For the scales on the left a low rating means positive evaluation, for the scales on the right high ratings are positive. Error bars are standard errors of the mean. * \( p < .05 \)

There was a linear trend to rate the system better dependent on the level of “indirectness” of the questionnaire modality, from \textit{agent} (worst), over \textit{online} (better) to \textit{pen and paper} (best) for \textit{tolerance} (\( t(56) = 2.49, p = .008, r_{\text{contrast}} = .32 \)) and \textit{optimism} (\( t(56) = 2.58, p = .006, r_{\text{contrast}} = .33 \)).

Concerning the differences in rating between the \textit{agent} and the \textit{online} condition the system was rated significantly better in the \textit{online} than in the \textit{agent} condition for \textit{intrusiveness} (\( t(56) = 3.21, p = .001, r_{\text{contrast}} = .39 \)), and \textit{distraction} (\( t(56) = 3.39, p < .001, r_{\text{contrast}} = .41 \)). We consider the rating to be better, since for these scales high values represent a negative evaluation of the system.

12.3.2. AttrakDiff

Means and standard deviations for the scales of the AttrakDiff questionnaire are summarised in Table 12.2.
12.3. Results

Figure 12.5.: Average rating of the systems for the AttrakDiff questionnaire dependent on condition. Error bars are standard errors of the mean.

<table>
<thead>
<tr>
<th></th>
<th>pen and paper</th>
<th>online</th>
<th>agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>pragmatic quality</td>
<td>4.13 1.06</td>
<td>4.94 .65</td>
<td>4.82 .71</td>
</tr>
<tr>
<td>hedonic quality - identity</td>
<td>4.88 1.17</td>
<td>4.61 .83</td>
<td>4.32 .9</td>
</tr>
<tr>
<td>hedonic quality - stimulation</td>
<td>4.81 1.27</td>
<td>3.63 .93</td>
<td>3.58 .95</td>
</tr>
<tr>
<td>attractiveness</td>
<td>4.82 1.22</td>
<td>4.57 .76</td>
<td>4.1 .91</td>
</tr>
</tbody>
</table>

Table 12.2.: Means and standard deviations for the results attained with the AttrakDiff questionnaire.

The results for each item dependent on condition is shown in Figure 12.5.

Concerning **pragmatic quality** \( F(2, 56) = 5.16, p = .009, \eta^2 = .16 \) and **hedonic quality - stimulation** \( F(2, 56) = 7.60, p = .001, \eta^2 = .21 \) there were significant differences in the rating of the system dependent on condition. For **attractiveness** the difference was marginal significant \( F(2, 56) = 2.80, p = .07 \) and for **hedonic quality - identity** there was no significant difference \( F(2, 56) = 1.53, p = .23 \).

Similar to the results for our own questionnaire there was a trend to rate the system better depending on the level of “indirectness” of questionnaire modality for **hedonic quality - stimulation** \( t(56) = 3.86, p < .001, r_{\text{contrast}} = .46 \), and **attractiveness** \( t(56) = 1.97, p = .03, r_{\text{contrast}} = .26 \). With respect to **pragmatic quality** \( t(56) = 3.01, p = .002, r_{\text{contrast}} = .37 \) the relation was in the other direction with ratings being better if the modality of the questionnaire was more direct.

The planned analysis of better ratings for the **agent** than for the **online** condition
12. Animated Agents as Social Actors

did not show any significant differences for pragmatic quality \( (t(56) = 0.53, p = 0.3) \), hedonic quality - identity \( (t(56) = 1.01, p = 0.16) \), hedonic quality - stimulation \( (t(56) = 0.18, p = 0.43) \), and a significantly worse rating for attractiveness \( (t(39, 29) = 1.85, p = 0.04, r_{\text{contrast}} = 0.28) \)

12.4. Discussion

The strong preference of choosing the positive option to close the agent after successfully finishing a step-by-step guide reported in the introduction to this experiment perfectly fits the idea of users mindlessly applying social scripts to the interaction with computers [Nass and Moon, 2000].

This seems to make the fact that we did not find the expected differences for the different types of questionnaires even more peculiar.

We found differences for five out of twelve scales. For two out of the five scales with differences rating was worst for the agent condition, better for the online condition and best for the pen and paper condition. The same is true for one scale only showing a marginal significant difference between conditions. For the remaining scales there was no homogeneous pattern. Most times a contrast analysis showed that either the pen and paper condition differed from the two other conditions or the two online conditions differed.

Those results are not conforming to the findings of Nass et al. [1994]. What are possible explanations for this contradictory findings? Within the recent years, users' attitude towards computers might have changed. While computers might have been perceived as a monolithic entity in the past, computers nowadays are usually connected to the Internet. More and more applications are running in a user’s web-browser, as was the application used for this study. This implies that applications are no longer restricted to a single computer and your data and applications are available at various computers or even smart-phones. Thus, the perimeter between “my computer” and the Internet – or cloud if you want to call it that way – is blurring. Computers might be perceived by users as a kind of window to the Internet. And if there is no well defined physical partner, applying social rules becomes less appropriate. Additionally, the findings of Joinson [1998] suggests that in the context of the perceived anonymity of the Internet users tend to show different behaviour than they do in “real life” situations. In a study reported by Joinson [1999] users especially reported lower social anxiety and social desirability when they were using the Internet than when they were using paper-based methods.

Thus, we argue that applying social scripts to human-computer interaction becomes less appropriate and that the usage of the Internet among with its implications on users’ social behaviour becomes more and more dominant. The mindlessly applied social script towards computers as postulated by Nass and Moon [2000] could also become more and more replaced by the social behaviour users exhibit in an Internet context.
12.4.1. Conclusion

First of our experiment showed that the modality of questionnaires has an influence on the results in the context of user interface evaluation. Thus, it is problematic to combine results attained with different modalities (e.g. an online or a pen and paper questionnaire) and preferable to keep the modality of the questionnaire consistent. Additionally we presented first indicators that the paradigm of users treating computers as social actors might become obsolete in an increasingly network-based and dislocated computing environment.
Part IV.

Conclusion and Future Directions
In this thesis we have reviewed and evaluated help systems and sources for assistance available in case users experience problems using computer software. While users frequently experience problems working with computers and these problems can cause serious frustration and decrease efficiency, the internal help systems of software are hardly consulted in case of frustrating situations [Lang et al., 2013a]. Based on this finding we reviewed and evaluated existing help systems of software with respect to their usefulness and issues concerning their impact on users' cognitive load that might inhibit learning and restrict utility. None of the existing help systems in productive use is capable to quickly introduce users to the complex UI of software fostering a paretro efficient usage. The concept of paretro efficiency implies that the majority of tasks can be completed using only a subset of the features available for a given program. By reducing the number of features offered, this problem could be avoided. However, we do not consider the reduction of features to be a beneficial approach due to various reasons. The most important of these reasons is that a software lacking a required feature might not be usable at all for certain users, forcing those users to switch to another program offering the functionality needed. Forcing users to switch to other applications causes frustration and additional workload on behalf of users and is definitely not desirable from a software vendor's perspective. Hence, we consider software offering more features than a user needs to complete his particular tasks to be unavoidable. In order to enable users to use software in an efficient way nevertheless, help systems could be used to point out essential elements of a UI and stress important aspects and peculiarities involved in the interaction with particular software. We consider guides implementing worked examples to be an excellent approach to achieve this. Completing a worked example users can be introduced to and learn about essential elements of the UI. Additionally, users quickly receive first results completing a worked example tutorial and can refine these first results according to their needs. This refinement process usually involves less essential or prominent elements of the UI. Thus, in addition to introducing novice users to the software, a help system should also be able to provide assistance in the context of these less important elements. Using software also involves extraneous or “nice to know” information concerning underlying concepts and paradigms. While this information might not be essential to complete tasks, interested users might want to learn more about these concepts to actually understand what they are doing and why things are done the way they are done. While this kind of information should be excluded from the guides used for the worked examples in order to increase coherence and decrease cognitive load on users, from our perspective this information should nevertheless be provided by the help system. To distinguish between extraneous and essential material provided by the help system appropriate signaling strategies have to be implemented.

Based on these preliminary considerations we implemented a help system for a grid computing web portal [Lang et al., 2011; Lang and Minker, 2012; Lang and Nothdurft, 2012]. The focus of the help system implemented was on reducing extraneous processing, by increasing coherence, spatial, and temporal contiguity. The help system was capable of:

• Assisting novice users to familiarize with the UI of the web portal and to complete
tasks using a worked example based approach.

- Providing assistance for elements of the UI that were not essential to complete basic tasks, but that were required to refine initial results.

- Introducing users to concepts related to the portal, grid computing and the applications accessible through the UI of the portal.

Information was provided in direct proximity of the explained content and the help system was capable to interact with the UI of the portal by pointing at arbitrary elements and providing example input for input fields. It was also capable to use spoken out- and input to interact with users and to offer assistance proactively.

Another aspect we focused on during the implementation of the help system was its portability and seamless as well as straightforward integration into the existing portal framework. To allow developers to easily and efficiently add functionalities of the help system to their applications various components have been developed. Those components have been well documented and tutorials on how to use the components have been provided for developers. As well as the step by step tutorial feature of the help system itself, these tutorials followed a worked example based approach.

While the implemented components offered extensive capabilities for customization, we considered an intelligent default behaviour of components to be of major importance. This way developers could quickly achieve acceptable results embedding the help system into their applications. The intelligent default behaviour of the system maximized user benefit for a minimal effort required from application developers. This benefit could have been increased by developers by applying further customization.

The tight integration of the help system could actually help developers to save time they would otherwise have to spend on creating separate help content. Especially, changes to the UI of applications usually also require the help content to be updated accordingly. In case of the implemented help system those changes were reflected instantly by the help system and did not require further adaptations. This was also the case if the layout of the portal was changed. If help content was provided to users employing an help page based approach those changes would also have made adaptations of the corresponding help content necessary.

Due to access restrictions the productive web portal the help system was initially developed for could not be used to evaluation purposes. Hence, we implemented a different scenario consisting of a nuclear power plant simulation to evaluate our system.

The first experiment conducted compared our advanced help system to a traditional HTML help page based approach of user assistance. To avoid a possible influence of using a virtual agent on user acceptance of the system we decided to not use an anthropomorphic representation of the help system for this experiment. The results of the first experiment clearly showed the superiority of our system concerning objective measures. Compared to help pages task completion times as well as mistakes made during interaction were significantly lower for our system. This superiority was not reflected in subjective scales evaluating the performance of the help systems. This is an indicator that users hardly appreciate the increased efficiency induced by using the advanced help system. As soon
as subjective scales involved hedonic aspects of the interaction process the advanced help system was rated better than the help page based system. This shows that even though users might not appreciate the advanced help system from a performance perspective, they nevertheless prefer interacting with the advanced system. This is an encouraging finding, since — as the example of Clippy shows — the success of help systems is highly dependent on user acceptance. While from our perspective some features of the help system involving Clippy could actually have been beneficial to users, it was nevertheless considered to be annoying by most users and consequently its usage had to be abandoned. Thus, the focus of the remaining experiments conducted in the context of this thesis was on user acceptance of our help system.

Accordingly, the second experiment compared a help system using an anthropomorphic agent to a system represented by a generic question mark symbol. Objective results showed that for the first task of the nuclear power plant simulation, participants using the system represented by an anthropomorphic agent needed significantly less time to complete the tasks and were significantly less often trying to solve the task without assistance using a trial and error based approach. We consider this an indicator that users were more curious to interact with the anthropomorphic agent and thus tended to consult him earlier in the task solving process. This curiosity to interact with the anthropomorphic agent might be a nice way to draw attention towards a help system. Especially, since subjective measures practically revealed no differences in user acceptance between the two systems.

Another way to draw attention to help systems is proactively offering assistance to users. This feature was extensively used in the context of Clippy and is often held responsible for it as being perceived to be annoying. To evaluate the influence of proactivity on users’ acceptance of help systems, we performed another experiment [Lang et al., 2013b]. This experiment compared the system using the anthropomorphic agent from the previous experiment to a system using the same anthropomorphic agent that proactively offered help to users. Our results for this experiment showed that while there were no differences between the systems concerning objective measures, there was a strong tendency to rate proactive system worth then the reactive one. We consider this an indicator that users in general do not appreciate assistance being proactively offered to them.

An important aspect of the proactivity feature in the previous experiment was that users had no control over proactivity. This means they were not able to activate or deactivate it. To investigate the influence of giving users an option to control the proactive behaviour of the system we performed our next experiment. We compared the reactive system using an anthropomorphic agent to a system using the same anthropomorphic agent, proactively offering assistance and offering users an option to deactivate and activate proactivity. Similar to the previous experiment objective results did not reveal significant differences between both conditions. However, concerning subjective scales the negative influence of proactivity could be mitigated by offering users the option to deactivate proactivity. This is a strong indicator that users accept proactive behaviour as long as they are kept in control and are able to deactivate it.

Nevertheless, taking into account that proactivity did not increase efficiency of users
for both experiments, we argue that proactivity should be avoided in the context of help systems. We believe that users are usually well aware, when they are in need of assistance and will consult the help system in those situations anyway. Assuming of course that users are aware of the existence of the help system and that they have learned that this help system can be a valuable resource for assistance in case of frustrating experiences. Thus, the exclusive benefit of proactively offering assistance to users is to announce the existence of the help system and to introduce users to its features and capabilities. Proactively offering information to users on how to work with a computer program more efficiently seems rather detrimental to us. Especially considering the impact of this kind of interruption on users workflow and cognitive load by introducing extraneous material that is not guaranteed to be useful.

The modality principle suggests that it might be beneficial for help systems to provide information on the visual UI of computer programs using auditory spoken output and input in order to reduce cognitive load. Accordingly, the help system implemented this feature. However, since the productive system was supposed to be employed for teaching purposes in public computer rooms, this feature was not used in the help system for the productive portal. We expected users not to appreciate and make use of the auditory feedback and speech input modality in public places. These concerns were validated by our experiment involving spoken output and input. While it is easy to make sure that the spoken output modality is not disturbing other users by using a headset, there is no way to unobtrusively use spoken input. In this context using spoken input to interact with computers in public still seems to feel weird to users. Accordingly, none of the users of our help system providing a spoken input modality used this feature if usage was optional and there were other persons present. However, as soon as social pressure on participants was increased by explicitly asking them to try the feature at least once, all participants made use of it. For participants using the feature we identified three remarkably different patterns of usage. There was a small group of participants trying the system only once — as asked by the supervisor — and two equally sized groups with participants using spoken input occasionally but frequently and another group rather exclusively using spoken input. For the last two groups we found significant differences in personality traits indicating that the usage pattern might be dependent on personal characteristics.

The last experiment conducted evaluated social aspects of interacting with an anthropomorphic agent [Lang et al., 2013c]. While participants of our experiments tended to neglect the help system having human like characteristics they nevertheless behaved socially towards the system by — in the vast majority of cases — choosing the positive option to stop interaction with the system after completing step by step guides. Following the findings of Nass et al. [1994] we implemented an experiment evaluating the help system using three different modalities. One group evaluated the system using pen and paper questionnaires, another group was using typical online questionnaires, and for the third group evaluating questions were presented directly by the anthropomorphic agent representing the help system. We expected participants to behave social towards the agent and rate the help system better, when questions were asked directly by the agent. However, results were not conforming to our hypothesis and the findings of Nass et al. [1994]. We argue that this might be due to the fact that users' perspective on computers might
have changed. While computers were detached entities in the past, today computers might rather be perceived as an appliance to access the Internet. Applying social scripts to such an appliance, especially in the case of answering questionnaires, might no longer be considered as appropriate as it was when interacting with a rather monolithic entity.

Future Directions

The help system implemented in the context of this thesis took extensive care in minimizing extraneous processing and cognitive load on users in order to maximize learning efficiency. This was achieved following the principles reported in literature. The experiments conducted in the context of this thesis did not evaluate the help system's performance with respect to cognitive load and learning. We especially did not evaluate learning success and thus the objective performance of our help system. To us it seemed more important to evaluate users’ acceptance of different help systems, since we consider the success of a help system to be rather dependent on user acceptance than on its objective performance. The failure of Clippy and the finding that users did subjectively not appreciate the increased performance using the advanced help system instead of the help page based system, seem to support this view. Nevertheless, the effects of using the advanced help system on learning should be evaluated to verify the assumptions made throughout this thesis. Especially, the concept of fostering exploratory learning by completing a worked example seems very promising in this context. The design of the evaluation scenario did not cover this aspect at all and exploratory learning was not an applicable strategy for any of the implemented tasks.

Concerning the spoken input modality, we found that users tend not to use spoken input when other persons are present. Participants stated that it would feel weird to talk to a computer with other people around. To find out if the reluctance of using spoken input was actually caused by the presence of people, the experiment should be repeated under private conditions. If users did not use the spoken input modality in private this would indicate that users are currently reluctant to use spoken input to interact with computers in general. In case they used the feature this would allow to verify our findings concerning the relation between personality traits and spoken input usage patterns. While our results suggest that personal characteristics actually have an influence on the way spoken input is used, this finding could still be a side effect of participants with diverging personal properties just reacting different to the social pressure implied by asking them to try the spoken input modality.

On a final note, we consider the field of computer help systems to be a very complex area that has not sufficiently been investigated by research. The current state of help systems does by no means reflect or make use of well established findings from learning and cognitive load theory. Hence, it is not surprising that users hardly use existing help systems if they experience problems with computers. Especially, since software companies have to dedicate time and money into creating help content for their programs this situation seems to be unsustainable for all parties involved and should definitely be improved. We believe that putting the findings of cognitive load theory into action.
as was done for the help system introduced in the context of this thesis - can be a straightforward approach to achieve such an improvement. Additionally, further research is required to increase usability and usefulness and to identify factors influencing users’ subjective acceptance of help systems.
A. Questionnaires

Participants of the different experiments had to complete various questionnaires. In the first phase of each experiment participants had to fill out questionnaires concerning personal data and personality traits. After completing those forms the interaction was performed and finally different questionnaires evaluating system performance and user satisfaction were filled in.

All questionnaires, except for the pen and paper-condition of Chapter 12, were online forms that were completed on the same computer used for the interaction.

A.1. Before Interaction

The data assessed before the interaction with the different systems took place, was the same for all treatment groups. It consisted of several parts. With the first part concentrating on demographic data the remaining two parts used standardized questionnaires to measure self-efficacy along with the big five personality traits.

A.1.1. Personal Data

As can be seen in figure A.1 the first part of the questionnaire requested some general demographic data. As it was important to be able to clearly allocate the questionnaires filled out before and after the interaction to one person while protecting participants’ anonymity a sophisticated algorithm was used to generate an unambiguous code word. This code word consisted of:

1. First letter of your mother’s given name ⇒ one letter
2. First letter of your father’s given name ⇒ one letter
3. Second letter of your own given name ⇒ one letter
4. Day of month of your birthday in two digit (dd) format ⇒ two digits

A pen and a dedicated line on the sheet of paper that was handed out to each participant was provided to allow subjects to note down their code word, in order to relieve them from remembering their word or having to recompute it. From the overall 204 participants, there was only one swapping two letters in his code word and two participants not adhering to the instructions at all using arbitrary pseudonyms. All these issues could have been resolved for final analysis and it may be stated that the given coding system worked exceptionally well.
A. Questionnaires

- Bitte geben Sie hier Ihr persönliches Codewort ein.

- Bitte geben Sie Ihr Geschlecht an.
  - weiblich
  - männlich

- Wie alt sind Sie?
  In dieses Feld dürfen nur Ziffern eingetragen werden.

- Was ist Ihr höchster Schulabschluss?
  - Allgemeines Abitur
  - Fachgebundene Hochschulreife (FOS/BOS)
  - Hauptschulabschluss
  - Realschulabschluss
  - Kein Schulabschluss

- Welches Fach studieren Sie aktuell oder haben Sie studiert?

- Falls Sie sich nicht mehr in der Ausbildung befinden, was ist Ihre aktuelle berufliche Tätigkeit?

Figure A.1.: The questions asked to assess demographic information.

A.2. After Interaction

Different to the questionnaires used before the interaction the questionnaires used after the interaction were not consistent throughout experiments. All questions asked for all experiments will be listed in the next section.

A.2.1. Self Developed Subjective Evaluation Questionnaire

For all items of this questionnaire participants could rate their attitude towards the help system on a 7 point Likert scale with poles labeled “strongly agree” and “strongly disagree” (see Figure A.2.1).
The following lists contain all items used for data collection. Due to bad internal consistency some items have been eliminated before final analysis. Those items are indicated by a “-” replacing the “•” symbol in the corresponding list. Additionally the alpha-values for the scale if the particular item was not eliminated is specified in brackets. The rating for some scales was for high numbers representing unfavorable ratings. Those scales have been reversed for evaluation, to have higher numbers reflect better rating. Inverted scales are indicated by a “*” symbol in front of the scale name.

For the agent condition as described in Section 12 the statements of this questionnaire have been changed to be in first person. For example the statement “The avatar’s proposals for solutions are useful” was altered to “Are my proposals for solutions useful?”

<table>
<thead>
<tr>
<th>scale</th>
<th>item</th>
<th>⇔</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help Behaviour</td>
<td>Die Lösungsvorschläge des Avatars sind hilfreich</td>
<td></td>
<td>.8</td>
</tr>
<tr>
<td></td>
<td>Der Avatar gibt die Hilfe stets zum richtigen Zeitpunkt</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar erledigt seine Hilfe ordentlich</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ohne die Hilfe des Avatars könnte ich die Aufgaben alleine nicht lösen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intrusiveness</td>
<td>Ich hätte gerne mehr Zeit, um selber auf die Lösungen für die Probleme zu kommen</td>
<td></td>
<td>.71</td>
</tr>
<tr>
<td></td>
<td>Das Hilfsangebot des Avatars ist überflüssig</td>
<td></td>
<td>.63</td>
</tr>
<tr>
<td></td>
<td>Der Avatar ist aufdringlich</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar steht gerne im Zentrum des Geschehens</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distraction</td>
<td>Der Avatar versperrt mir die Sicht auf die Szene</td>
<td></td>
<td>.87</td>
</tr>
<tr>
<td></td>
<td>Die Anwesenheit des Avatars lenkt mich von den Aufgaben ab</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar zieht meine Aufmerksamkeit ständig auf sich</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar irritiert mich bei der Bearbeitung der Aufgaben</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impact</td>
<td>Das Verhalten des Avatars macht mich nervös</td>
<td></td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td>Die Anwesenheit des Avatars beruhigt mich</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure A.2: Exemplary illustration of the Likert scale used for the self developed questionnaires.
### A. Questionnaires

<table>
<thead>
<tr>
<th>scale</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Der Avatar wirkt in seinem Auftreten präsent</td>
</tr>
<tr>
<td></td>
<td>Der Avatar wirkt auf mich wie ein Streber</td>
</tr>
<tr>
<td>Dominance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar ist arrogant und überheblich</td>
</tr>
<tr>
<td></td>
<td>Ich habe eine Abneigung gegen weiße Kittel</td>
</tr>
<tr>
<td></td>
<td>Der Avatar ist kühl und distanziert</td>
</tr>
<tr>
<td></td>
<td>Ich fühle mich durch den Avatar eingeschüchtert</td>
</tr>
<tr>
<td></td>
<td>Ich denke der Avatar fühlt sich anderen selten unterlegen</td>
</tr>
<tr>
<td></td>
<td>Ich denke nicht, dass der Avatar entspannt ist und aufgeben will, wenn etwas schief geht</td>
</tr>
<tr>
<td>Optimism</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ich denke der Avatar zieht es gewöhnlich vor, Dinge allein zu tun</td>
</tr>
<tr>
<td></td>
<td>Der Avatar wirkt auf mich wie ein gut gelaunter Optimist</td>
</tr>
<tr>
<td></td>
<td>Der Avatar verliert nicht den Mut, auch wenn Probleme auftauchen</td>
</tr>
<tr>
<td></td>
<td>Selbst in einer stressigen Situation, kann der Avatar einem Problem noch etwas Gutes abgewinnen.</td>
</tr>
<tr>
<td>Tolerance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar ist freundlich</td>
</tr>
<tr>
<td></td>
<td>Der Avatar macht mir Mut, die Aufgaben bewältigen zu können</td>
</tr>
<tr>
<td></td>
<td>Ich finde den Avatar sympathisch</td>
</tr>
<tr>
<td></td>
<td>Ich finde den Avatar zynisch und skeptisch</td>
</tr>
<tr>
<td>Humanity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Die Mimik des Avatar lenkt mich ab</td>
</tr>
<tr>
<td></td>
<td>Der Avatar wirkt lebendig auf mich</td>
</tr>
<tr>
<td></td>
<td>Der Avatar zieht meine Aufmerksamkeit auf sich</td>
</tr>
<tr>
<td></td>
<td>Der Avatar wirkt warm und persönlich</td>
</tr>
<tr>
<td></td>
<td>Der Avatar beeinflusst meine Stimmung</td>
</tr>
<tr>
<td></td>
<td>Ich kann mich mit dem Avatar identifizieren</td>
</tr>
<tr>
<td></td>
<td>Ich kann viel Spaß mit dem Avatar haben</td>
</tr>
<tr>
<td></td>
<td>Ich finde den Avatar interessant</td>
</tr>
<tr>
<td>Competence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Der Avatar wirkt sehr wissend und erfahren auf dem Gebiet der Atomphysik</td>
</tr>
<tr>
<td></td>
<td>Ich denke der Avatar versucht seine Aufgaben sehr gewissenhaft zu erledigen</td>
</tr>
<tr>
<td></td>
<td>Der Avatar weiß immer was zu tun ist</td>
</tr>
<tr>
<td></td>
<td>Ich kann mich auf den Avatar verlassen</td>
</tr>
</tbody>
</table>

238
A.2. After Interaction

Due to its low internal consistency that could not be improved by eliminating certain items, the Impact scale was excluded from analysis. The items used are reported for the sake of completeness.

Despite of its borderline internal consistency (even with respect to the rather lax limits of Nunnally et al. [1967]) the Dominance scale was used for analysis.

<table>
<thead>
<tr>
<th>scale</th>
<th>item</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Der Avatar drückt sich klar und präzise aus</td>
<td>Der Avatar gibt nur relevante und wichtige Informationen</td>
<td>Der Avatar wirkt kompetent auf mich</td>
</tr>
<tr>
<td>Der Avatar wirkt auf mich nicht sehr systematisch</td>
<td></td>
<td>vorgehend</td>
</tr>
<tr>
<td>Der Avatar wirkt nicht sehr wissbegierig</td>
<td></td>
<td>⇔</td>
</tr>
</tbody>
</table>

Table A.1.: Scales, items, and Cronbach’s alpha values for the self developed questionnaire. (⊙=Scale/Item was eliminated; ⇔=Scale/Item was inverted).
Bibliography


Bibliography


243


IBM Corp. IBM SPSS Statistics for Linux, 2012.


H. B. Mann and D. R. Whitney. On a Test of whether One of Two Random Variables is Stochastically Larger than the Other. The annals of mathematical statistics, 18(1):50–60, 1947.


G. A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological review, 63(2):81, 1956.
Bibliography


Bibliography


List of Figures

1.1. Scotty the chief engineer of the Enterprise talking to a mouse. Used with permission from Columbia Broadcasting System. .................................................. 5
1.2. Model of human memory according to Atkinson and Shiffrin [1968] ...................... 16
1.3. Baddeley’s model of working memory Baddeley [2012] ........................................ 17
1.4. The action cycle from Norman [2002] adapted to computer usage. ......................... 21
1.5. The uncanny valley according to Mori [1970] .................................................... 25

2.1. UI of the help pages of version 3.8.1 of Eclipse. Used with permission of the Eclipse Foundation. The Eclipse IDE and Help System are open source software, distributed under the Eclipse Public License. .................................................. 29
2.2. UI of the help pages of version 21 of SPSS. Reprint Courtesy of International Business Machines Corporation, ©International Business Machines Corporation. SPSS Inc. was acquired by IBM in October, 2009. ............................................ 29
2.3. UI of the help pages of version 2.8 of GIMP. The software shown on the screenshot is entirely licensed under the (L)GPL and thus Free Software. ...................... 30
2.4. UI of the help pages of version 4.2.4.2 of LibreOffice. The software shown on the screenshot is entirely licensed under the (L)GPL and thus Free Software. ...................... 30
2.5. UI of the help pages integrated into Microsoft Office for Windows 2010 in offline mode (top) and online mode (bottom). Used with permission from Microsoft. .................. 32
2.6. UI of the help pages integrated into the Synology Disk Station Manager. Used with permission of Synology Inc. ................................................................. 33
2.7. Online help pages of Google Chrome. Google and the Google logo are registered trademarks of Google Inc., used with permission. ................................. 35
2.8. Online help pages of Mozilla Firefox. Firefox is open source software, distributed under the Mozilla Public License. .................................................. 35
2.9. Tooltips in macOS Sierra, Windows 7, and Ubuntu 14.04. Screen shot(s) reprinted with permission from Apple Inc. (top). Used with permission from Microsoft (middle). Used with permission from Canonical Ltd. (bottom). ................................................. 37
2.10. The “What’s this” feature in Microsoft Word 97. Used with permission from Microsoft. .................. 38
2.11. The button for the “What’s this” feature in a KDE application. The software shown on the screenshot is entirely licensed under the (L)GPL and thus Free Software. .................. 38
2.12. The “What’s this” feature in KDE. The software shown on the screenshot is entirely licensed under the (L)GPL and thus Free Software. .................. 39
2.13. Extended tooltip for the “Format Painter” UI control in Microsoft Word 2013 for Windows. Used with permission from Microsoft. .................................. 39
2.14. Interface of the HUD feature in recent versions of Ubuntu. Used with permission from Canonical Ltd. ................................................................. 40
2.15. Interface of the Help Menu Search feature of macOS Sierra. Screen shot(s) reprinted with permission from Apple Inc. .................................................. 41
2.16. Interface of the offline help in Microsoft Word 2013 for Windows. Used with permission from Microsoft. ................................................................. 42
List of Figures

3.17. Help dialogue presented to users after booting *Windows 95*. Used with permission from Microsoft. .......................................................... 43
3.18. Selection of introductory screens presented to users during the *Ubuntu 13.04* installation process. Used with permission from Canonical Ltd. .......................................................... 43
3.19. First screens of the tour presented to users of *Microsoft PowerPoint 2013*. Used with permission from Microsoft. .......................................................... 45
3.20. Typical notification messages as displayed after updates of Apps on mobile phones and tablet computers introducing new features or changes to the UI. Used with permission from Microsoft (left). Used with permission from Runtastic GmbH (right). .......................................................... 45
3.21. Representation of Clippit the *Microsoft Office Assistant* in *Microsoft Office for Windows 97* and *Microsoft Office for Windows 2003*. Used with permission from Microsoft. .......................................................... 47
3.22. Clippy answering user queries. Used with permission from Microsoft. .......................................................... 47
3.23. Clippy offering advice to users. Used with permission from Microsoft. .......................................................... 47
3.24. Clippy suggesting to use the letter writing wizard, some dialogues of the wizard and the resulting letter. Used with permission from Microsoft. .......................................................... 48
3.25. *Nina Virtual Assistant* as displayed on the *Nuance* (left), *Jetstar* (middle), and *Coca Cola* (right) websites. Used with permission from Nuance Communications Inc. (left). Used with permission from Jetstar Airways Pty Ltd. (middle). Used with permission from The Coca Cola Company (right). .......................................................... 50
3.26. *IKEA Anna* as presented on the web page of *IKEA*. Used with permission of Inter IKEA Systems B.V. .......................................................... 51
3.27. *IKEA Anna* telling users to click the non existent ‘Read more’ option to get further information on specific items. Used with permission of Inter IKEA Systems B.V. .......................................................... 52
3.28. Screen shots of the Network Connection Assistant of *macOS Sierra*. Screen shot(s) reprinted with permission from Apple Inc. .......................................................... 53
3.29. Questions concerning *Microsoft Word* in the *superuser* community of *StackExchange*. Used with permission from Stack Exchange Inc. .......................................................... 56

5.1. Classification of grid computing according to Foster et al. [2008] .......................................................... 74
5.2. Flowchart of a servlet-based web application. .......................................................... 77
5.3. Flowchart of a portlet-based web application. .......................................................... 81
5.4. Page of *GridSphere* based web portal containing two portlets. .......................................................... 82
5.5. Representation of the agent when inactive .......................................................... 82
5.6. Representation of the agent during mouse hover. Superimposed image of arrows to indicate movability .......................................................... 83
5.7. Representation of the agent after activation .......................................................... 84
5.8. The agent with an adapted list of proposed options .......................................................... 84
5.9. Results for a search query .......................................................... 85
5.10. The first page of the UI of the *bwGRID Science Portal*. .......................................................... 86
5.11. The ‘NWChem/Gaussian’ subtab of the ‘Chem’ section of the *bwGRID Science Portal*. .......................................................... 87
5.12. The options for assistance offered by the agent after activation in the ‘NWChem-/Gaussian’ subtab of the ‘Chem’ section of the *bwGRID Science Portal*. .......................................................... 88
5.13. The agent explaining all elements of the interface that need to be edited to achieve a given objective. .......................................................... 89
5.14. The agent proceeding to explain the input fields needed to achieve the given objective, after the UI of the ‘Gaussian/NWChem’ portlet became available. .......................................................... 90
5.15. The agent proposing to paste a working example for a *Gaussian* input file that was required for the input field. .......................................................... 91
List of Figures

5.16. The help system allowing users to enter "Maximum Wall Time" by days, hours, and minutes instead of seconds. ............................................ 91
5.17. The help system helping users to navigate to portlets involved in a step by step tutorial. ................................................................. 92
5.18. The help system, when the mouse pointer hovers over the representation of the agent. Arrows to indicate movability and "minimize" button in the top right corner. ......................................................... 93
5.19. Markers brought to front using the Ctrl key. ........................................ 93
5.20. Representation of the agent when inactive with added option to resume an interrupted step by step tutorial ........................................ 94
5.21. Representation of the agent after the "Help me with an element of this page!" function was activated. .......................................................... 95
5.22. The agent explaining the meaning of a term used in the text of the UI. ........ 96
5.23. The help system answering a user question in the speech balloon of the agent. ................................................................. 97
5.24. The help system explaining the concept of certificates and MyProxy credentials to the user. .................................................... 98
5.25. The help system in complex answer mode with sequential markers activated. 99
5.26. The option to resume a presentation added to the UI of the help system in idle mode. ................................................................. 99
5.27. The UI of the help system with spoken out- and input enabled. ............... 100
5.28. Visualisation of the MVC pattern. ....................................................... 103
5.29. Interaction of components from the perspective of the help system. .......... 103
5.30. Representation of the help system, when waiting for a response from the server. ................................................................. 116
5.31. Overview of a spoken language dialogue system. .................................. 116
5.32. The introductory page of the documentation for the help system. ............ 119
5.33. An overview of files that need to be edited to add specific help system functionality to a portlet. .................................................... 119
5.34. A listing from a tutorial showing how to use the tag library of the help system to add specific features to an existing portlet. ................. 120
5.35. Excerpt from the documentation of the tag library explaining attributes of a tag along with their usage. ........................................ 120
5.36. Excerpt from the documentation of the client side implementation of the help system. .................................................... 121

6.1. The banner and navigational area of the nuclear power plant GUI. .......... 126
6.2. The section of the GUI labeled "Kontrolle" ........................................... 128
6.3. The section of the GUI labeled "Reaktor". ............................................ 128
6.4. The section of the GUI labeled "Primärkreislauf" .................................. 129
6.5. The section of the GUI labeled "Krisenreaktion" .................................... 129
6.6. The description of the first task, before clicking on "read". ......................... 130
6.7. The UI after clicking the "read" button. .............................................. 130
6.8. The UI after clicking the "read" button. .............................................. 131
6.9. The agent explaining the different functional units of the plant. ............... 132
6.10. The agent showing how the performance of the power plant is computed. .. 133
6.11. The agent pointing at an input field and providing the correct value to enter in order to solve the current task. ........................................ 135
6.12. The ANVIL workbench during annotation. The ANVIL software version 5.1.2 shown on the screenshot is free for noncommercial research and education. ©2000–2012 Michael Kipp. ................................................ 144

7.1. The first page of the help provided in the HTML condition. ..................... 148
7.2. The first page of the help provided in the HTML condition. ..................... 148
7.3. The tutorial to solve the first task in the HTML condition. ...................... 149
7.4. Visualisation of the agent as generic question mark in the interactive condition. ................................................................. 149
List of Figures

7.5. "Snapping" windows ........................................ 150
7.6. Average task completion times HTML and question mark conditions .......... 153
7.7. Wrong input HTML and question mark conditions .......................... 154
7.8. Trial and error attempts and resets HTML and question mark conditions ...... 154
7.9. Average rating own questionnaire HTML and question mark conditions ... 156
7.10. Average rating CSUQ HTML and question mark conditions ............... 157
7.11. Average rating AttrakDiff HTML and question mark conditions ......... 158
7.12. Number of participants finishing tasks in 5 minute intervals HTML and question mark conditions ........................................ 160

8.1. Representation of the help system in the agent (left) and question mark condition (right) ........................................................... 166
8.2. Average task completion times question mark and agent conditions .......... 168
8.3. Wrong input question mark and agent conditions ............................. 169
8.4. Trial and error attempts and resets question mark and agent conditions..... 169
8.5. Average rating own questionnaire question mark and agent conditions ... 171
8.6. Average rating AttrakDiff question mark and agent conditions .......... 172

9.1. Average completion times proactive and reactive conditions .............. 180
9.2. Wrong input proactive and reactive conditions ............................. 181
9.3. Trial error attempts and resets proactive and reactive conditions .......... 181
9.4. Average rating own questionnaire proactive and reactive conditions ..... 183
9.5. Average rating AttrakDiff proactive and reactive conditions ............ 184

10.1. The options available to control the behaviour of Clippy in Microsoft Word 97 for Windows. Used with permission from Microsoft ......................... 190
10.2. The agent with an option to deactivate proactivity (last option on the left) and to reactivate it after deactivation (last option on the right) .... 191
10.3. Average completion times choice and reactive conditions ............... 193
10.4. Wrong input choice and reactive conditions ..................................... 193
10.5. Trial and error attempts and resets choice and reactive conditions ...... 194
10.6. Average rating own questionnaire choice and reactive conditions ....... 196
10.7. Average rating AttrakDiff choice and reactive conditions ............... 197

11.1. Average completion times text, speech output, and speech input conditions .... 204
11.2. Wrong input text, speech output, and speech input conditions ............. 205
11.3. Trial and error attempts and resets text, speech output, and speech input conditions ......................................................... 206
11.4. Personality traits for casual and power users .................................. 208
11.5. Average rating own questionnaire text, speech output, and speech input conditions ........ 209
11.6. Average rating CSUQ text, speech output, and speech input conditions .... 210
11.7. Average rating SASSI text, speech output, and speech input conditions .. 211
11.8. Average rating AttrakDiff text, speech output, and speech input conditions .... 212

12.1. The help system at the end of step by step guides with the positive and negative option to close the system ................................. 218
12.2. Example questions asked directly by the agent in the agent condition .... 219
12.3. Typical page of a questionnaire created with LimeSurvey [Schmitz, 2012] as used in the online condition. The software shown on the screenshot is entirely licensed under the GPL and thus Free Software .................. 220
12.4. Average rating own questionnaire pen and paper, online, and agent conditions ................ 222
List of Figures

12.5. Average rating AttrakDiff pen and paper, online, and agent conditions . . . . . . . . . 223
A.1. The questions asked to assess demographic information. . . . . . . . . . . . . . . . . . . . . . . 236
A.2. Exemplary illustration of the Likert scale used for the self developed questionnaires. . 237
List of Tables

1.1. Recompiled list of specific causes for frustration as reported by Ceaparu et al. [2004]; Lazar et al. [2006] ................................................................. 8
1.2. Combined list of solutions to frustrating experiences from Ceaparu et al. [2004]; Lazar et al. [2006] ................................................................. 9
6.1. Cronbach's alpha for the condensed version of the NEO-FFI scales ................................................................. 136
6.2. Varimax rotated component matrix for a factor analysis of the items to measure different manifestations of cognitive load. (Coefficients < .4 omitted for clarity, unofficial translation of items from German to English by the author) ................................................................. 138
6.3. Cronbach's alpha for the scales with several remaining items for the differentiated cognitive load questionnaire by Klepsch et al. [2014] ................................................................. 138
6.4. Scales, number of items, sample item and Cronbach's alpha for our own questionnaire ................................................................. 140
6.5. Scales, number of items per scale, sample item and Cronbach's alpha for CSUQ ................................................................. 141
6.6. Scales, number of items per scale, and Cronbach's alpha for SASSI ................................................................. 141
7.1. Means and standard deviations objective scales HTML and question mark conditions ................................................................. 152
7.2. Means and standard deviations own questionnaire HTML and question mark conditions ................................................................. 155
7.3. Means and standard deviations CSUQ HTML and question mark conditions ................................................................. 156
7.4. Means and standard deviations AttrakDiff HTML and question mark conditions ................................................................. 157
8.1. Means and standard deviations objective scales question mark and agent conditions ................................................................. 167
8.2. Means and standard deviations own questionnaire question mark and agent conditions ................................................................. 170
8.3. Means and standard deviations AttrakDiff question mark and agent conditions ................................................................. 172
9.1. Means and standard deviations objective scales proactive and reactive conditions ................................................................. 179
9.2. Means and standard deviations own questionnaire proactive and reactive conditions ................................................................. 182
9.3. Means and standard deviations AttrakDiff proactive and reactive conditions ................................................................. 183
10.1. Means and standard deviations objective scales choice and reactive conditions ................................................................. 192
10.2. Means and standard deviations own questionnaire choice and reactive conditions ................................................................. 195
10.3. Means and standard deviations AttrakDiff choice and reactive conditions ................................................................. 196
11.1. Means and standard deviations objective scales text, speech output, and speech input conditions ................................................................. 204
11.2. Means and standard deviations big five and self-efficacy marginal, casual, and power users ................................................................. 207
11.3. Means and standard deviations own questionnaire text, speech output, and speech input conditions ................................................................. 208
11.4. Means and standard deviations CSUQ text, speech output, and speech input conditions ................................................................. 209
11.5. Means and standard deviations SASSI speech output and speech input conditions ................................................................. 210
11.6. Means and standard deviations Attrakdiff text, speech output, and speech input conditions ................................................................. 211
List of Tables

12.1. Means and standard deviations own questionnaire pen and paper, online, and agent conditions ........................................... 221
12.2. Means and standard deviations AttrakDiff pen and paper, online, and agent conditions 223
A.1. Scales, items, Cronbach’s alpha own questionnaire ........................................ 239
List of Own Publications


List of Own Publications

Parts of this thesis have previously been published in the publications listed in the List of Own Publications.
Curriculum Vitae

Professional Experience

since 05/2017 **Software Developer**, Digital Customer Experience and Smart Factory Group, Daimler TSS, Ulm.


03/2011–01/2016 **Research Assistant**, Institute of Communications Engineering, University of Ulm.

03/2009–03/2012 **Project Team Member bwGRiD**, Communication and Information Center, University of Ulm.

10/2005–02/2008 **Student Assistant**, Programming Workshop of the Computer Science Service Group, University of Ulm.

Education


Other Activities

09/2001–05/2002 **Basic Military Service**.