DAI-Labor
TU Berlin

Fachgebiet Agententechnologie
in betrieblichen Anwendungen (AOT)
Fakultät IV - Elektrotechnik und Informatik

**Betreuer:** Dr.-Ing. Stefan Fricke
Prof. Dr.-Ing. habil. Sahin Albayrak

# Diploma Thesis

## Simulation of Cognitive Processes for automated Usability Testing

Mathias Runge
204444

July 14, 2008

# Abstract

This diploma thesis deals with the simulation of cognitive processes of a simulated user who is embedded into a workbench for semi-automated usability evaluation. The behavior of the simulated user helps system engineers uncovering several usability problems of their modeled computer interfaces.

A straight approach is impeded by the low output of the still young research fields "cognitive science" and "usability", by missing definitions and by frequently ongoing discussions, especially in the research field psychology. Thus, new approaches are developed by the author derived from existing technologies or the todays understanding of human brain functionality.

Contrary to existing cognitive models, the processes are rather focused on the utilization for uncovering usability problems of interfaces than on solving common problems in cognitive science. Therefore, cognitive science and usability are outlined in the beginning of this thesis. Subsequently, requirements for the processes are defined and their implementation explained. In the end of this thesis, several examples are evaluated by the workbench for an explanation, how the developed cognitive processes support a better usability evaluation.

## Explanation

I declare that I wrote this diploma thesis independently and only with the usage of the given sources and aid. All parts of the text that were taken word-to-word or in content from the sources have been marked. This diploma thesis has not been presented to another board of examiners, whether in an equal nor in a similar version.

Berlin, July 14, 2008

Mathias Runge

# Contents

**Abstract** iii

**Abbreviations** ix

**1 Introduction** 1

**2 Cognitive Science** 3
2.1 Cognitive Processes . . . 3
2.2 Cognitive Systems in Artificial Intelligence . . . 4

**3 Usability** 7
3.1 Usability Evaluation . . . 7
3.2 Automated Usability Evaluation . . . 9

**4 MeMo** 11
4.1 Modeling Phase . . . 12
4.2 Simulation Phase . . . 13
4.3 Reporting Phase . . . 14
4.4 Requirements for User Interaction Model . . . 14

**5 Information Exchange** 17
5.1 Speech Acts . . . 18
5.2 Information Exchange Concept in MeMo . . . 20
5.3 UIM Integration . . . 25

**6 Knowledge Management** 27
6.1 Human Memory Systems . . . 28
6.1.1 Short-Term Memory . . . 29
6.1.2 Long-Term Memory . . . 29
6.2 UIM Integration . . . 31
6.2.1 Knowledge Base . . . 33
6.2.2 Working Memory . . . 38
6.2.3 Mental Model . . . 42

**7 Simulation of Emotions** 45
7.1 UIM Integration . . . 46
7.1.1 Dynamic User Attributes (DUA) . . . 47
7.1.2 Rule Engine . . . 48

**8 Decision Making** 51
8.1 UIM Integration . . . 52

8.1.1 Intention . . . . . 52
8.1.2 Keyword Engine . . . . . 54
8.1.3 Final UIM Version . . . . . 55

**9 Simulations** **57**
9.1 Login . . . . . 58
9.2 Notepad . . . . . 59
9.3 4 Star Cooking Assistant . . . . . 62
9.4 Voice System . . . . . 65

**10 Discussion** **69**

**A Screenshots of MeMo** **73**

**B Additional Material** **77**

**References** **79**

# Abbreviations

| | |
|---|---|
| ACT | **A**daptive **C**ontrol of **T**hought |
| AI | **A**rtificial **I**ntelligence |
| AUE | **A**utomated **U**sability **E**valuation |
| AVP | **A**ttribute **V**alue **P**air |
| CTT | **C**oncur **T**ask **T**ree |
| CWT | **C**ognitive **W**alk**t**rough |
| DUA | **D**ynamic **U**ser **A**ttributes |
| GOMS | **G**oals, **O**perators, **M**ethods, and **S**election Rules |
| HCI | **H**uman **C**omputer **I**nteraction |
| IEC | **I**nformation **E**xchange **C**oncept |
| IPP | **I**nteraction **P**robability **P**air |
| IPU | Information Processing Unit |
| IR | **I**nformation **R**etrieval |
| MBTI | **M**yers-**B**riggs **T**ype **I**ndicator |
| MHP | **M**odal **H**uman **P**rocessor |
| OCC model | emotion model from **O**rtony, **C**ollins and **C**lore |
| SDS | **S**peech **D**ialog **S**ystem |
| SIM | **S**ystem **I**nteraction **M**odel |
| STM | **S**ystem **T**ask **M**odel |
| SUA | **S**tatic **U**ser **A**ttributes |
| UA | **U**ser **A**ttributes |
| UE | **U**sability **E**valuation |
| WIMP | **W**indow, **I**con, **M**enu and **P**ointing device |

# 1 Introduction

Reports of historians about the ancient Greek and Rome include descriptions of automotive machines and androids. Equal tales are known from other cultures, but nobody can say whether these reports are myths or the truth. Even Homer reports in his "Ilias" about Hephaistos[1] who built artificial female servants. Certainly, machines, imitating human anatomy and behavior, are fascinating us since ages, and human intelligence was the subject of thoughts, dreams, novels and serious researches in all times. A surprising huge amount of literature from the middle age or renaissance, and especially modern literature or movies are dealing with humanoid machines. Famous and talented constructors such as Thomas von Aquin (1225-1274), Leonardo da Vinci (1452-1519) or Jacques de Vaucanson (1709-1782) tried to copy human anatomy, intelligence or behavior ([49, 21]). Their machines were adored but also were far from a perfect copy.
Modern fictive androids like "Data"[2] or real robots such as the "Honda P3" ([43, 36]) fire our imagination and hike the yearning for the creation of a perfect human copy. Although a huge amount of philosophers, psychologists or scientists in general, invest enormous time and effort into the research of human intelligence since ages, a complete understanding or a perfect android seems to stay a dream yet, even with todays technological potential.
Admittedly, the technological possibilities simplify our life, but the amount of machines and their growing complexity are creating new problems. Human beings are intelligent enough to serve complex machines, but the different way of thinking between the user and the system engineer leads to reductions of the satisfaction and efficiency with which a person performs her/his task on the machine. As a consequence, a new research field, known as "Usability", was founded and is dealing with the prevention of these problems. The mentioned incomplete understanding of human brain functionality is thereby a big impediment.
The MeMo workbench, for example, follows a methodology of usability evaluation and is based on rules for human behavior prediction which were generated with empirical data sets. These data sets were collected during observations of real users interacting with several systems. In a first version of MeMo, the system engineer was asked to define a prediction of the interaction flow that is needed

[1] the god of handcraft in ancient Greece
[2] famous artificial crew member of the "Enterprise" ("Star Trek"), acted by Brent Spiner

to be performed by a user for a successful task completion. The analysis of this approach discovered a lot of disadvantages. One conclusion was the need for a simulated user, who searches for a solution during his interaction with the system model, but the simulation of a user with realistic behavior includes the simulation of the user's cognitive processes.
This diploma thesis deals with the simulation of the mentioned processes in the context of automated usability testing, and section 2, therefore, gives an introduction and short overview of what cognitive processes are. Usability in general, and usability evaluation methods are described in section 3. Section 4 ("MeMo") explaines a little more detailed because the mentioned module, which represents the simulated user, is embedded within MeMo and communicates with a model of a system. The section describes a usability test with the new approach and defines requirements for the simulation of the user. To be more precisely, the section defines the cognitive processes which are needed for an automated usability evaluation of a system with the support of MeMo.
The ensuing sections derive ideas from existing understanding of human brain functionalities and/or from approaches of other researchers. The first problem that was needed to be solved belongs to the way the simulated user communicates with the system and is explained in section 5 ("Information Exchange"). The found solution for this problem leads to the question how the simulated user could store information which is explained in section 6 ("Knowledge Management"). The stored information, or knowledge, is the basis for processes involved in the decision making phase of the simulated user. Dependencies to this phase are described in section 7 ("Simulation of Emotion"), whereas the process itself is explained in section 8 ("Decision Making").
In each of these sections, the implementation of the simulated user will be explained step by step until the complete functionality is explained. Section 9 ("Simulations") gives a short summary of all processes and their cooperation. The cognitive processes and their requirements are explained with the help of several sample systems, whereas each of them was modeled with MeMo. The simulation results of these examples are given in section 9 as well.
Finally, a discussion in section 10 evaluates these results and gives an outlook for further developments of the simulated user.

# 2 Cognitive Science

Philosophers like Plato or Aristotle sought to understand the nature of human knowledge in ancient Greece. The understanding was expanded by other philosophers and psychologists such as Descartes, Berkeley, Kant or Hobbes in the 17th - 19th century. John Watson ([52]) and Clark Hull ([24]) founded behaviorism as a new point of view, but the term cognitive science was coined out the first time by H. Christopher Longuet-Higgins in 1973.
Cognition (lat.: cognoscere, "to know") refers to the numerous human-like processing of information. Therefore, cognitive science deals with any kind of mental operation or structure that can be studied in precise terms. The enormous complexity and the interdisciplinary nature of these terms lead to a fusion of different research fields as illustrated in figure 1. At this time cognitive science is not yet established as mature science with a clearly agreed upon sense of direction. Furthermore, the community is still growing and new scientific fields might join in future.

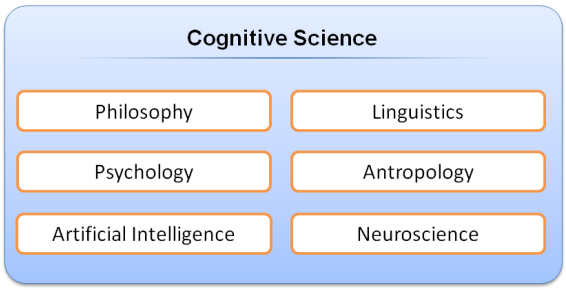



Figure 1: Cognitive science is a fusion of several research fields.

## 2.1 Cognitive Processes

Donald Norman pointed out twelve issues of cognitive science: belief systems, emotion, learning, performance, consciousness, interaction, memory, skill, development, language, perception and thought. All central processes in human mind dealing with these issues are cognitive processes, whereas the question whether peripheral perception and motor controls are included or not is still in discussion. Of course, perception has a big influence on the knowledge extraction from a user

interface, but Higgins based the primary focus on further evaluation comparable to software running on hardware ([29]). A person who is touching, viewing, smelling or tasting an apple with his/her sensor and motor systems does not need these systems for further central processes such as reasoning why an apple falls down. These thoughts are undeniably independent from perception and motor behavior, and above all, they may also be caused by describing the situation of a falling apple in a text.
Newell, on the other hand, included perception and motor behavior ([37]). He mentioned that the human motor and sensor system can not be simply divided into hardware and software. Psychologists are confronted with the problem of a missing, well defined threshold between an electronic signal in the sensor system and a complex information, whereas the information again consists of several electronic signals. Although cognitive science includes research fields dealing with that problem, motor control and peripheral perception are no cognitive processes for the majority of cognitive scientists.

## 2.2 Cognitive Systems in Artificial Intelligence

Contrary to the mentioned problem of psychologists, human computer models mostly do not have the missing threshold. Today researchers of each field in cognitive science using special human-like computer models, which give the field Artificial Intelligence (AI) an important role.
AI research can be categorized by strong AI, applied AI and cognitive simulation. The ultimate ambition of strong AI is to produce a machine whose overall intellectual ability is indistinguishable from that of a human being. Applied AI aims to produce commercially viable "smart" systems, whereas cognitive simulations are used to analyze Human Computer Interaction (HCI) or theories about how the human mind works.
The first successful demonstration of machine learning was published by Oettinger in 1952. He wrote an algorithm called "Shopper" running on a EDSAC[3] machine which learned where to find items in a small world of eight shops. Since 1952 AI became a huge science, and it is also included in a wide range of commercially computer hardware and software. Chess computers, for example, are not working without problem solving, and many computer games would be boring without intelligent opponents. Sensor systems of vehicles or airplanes were developed during researches in AI.
In contrast to applied AI, strong AI lost its importance since the 1980s, and many scientists doubt that a copy of the human brain could ever be built. Most of cognitive simulations are focused on behavior systems and do not need neural networks at all. Dependencies on human neural structures can be calculated by

[3]Electronic Delay Storage Automatic Calculator

special models like the Model Human Processor (MHP) ([37]) and therefore do not need to be implemented.
The Adaptive Control of Thought (ACT) is a psychological theory for the explanation of human cognitive activities, e.g. thinking, memory or language and was presented by John R. Anderson in 1983 ([8, 9, 38]). The implementation oft ACT is a cognitive architecture and belongs to the class of production[4] systems. Since 1983 ACT was extended by numerous scientists and in different directions. The types of expansions are marked as special suffixes of ACT*. The 'R' in the term ACT-R, for example, stands for rational and refers to the rational behavior of the model (cost-benefit analysis). ACT-R was developed by Anderson as well and belongs to the todays state of the art of cognitive process simulators ([9, 38, 31]).
This section outlined the term "cognition" for a further understanding of issues this thesis is dealing with, whereas a definition itself could not be given. Contrary to cognitive science, the science of usability affords definitions which advance the understanding of the next section.

[4] action implications in the form "if A is true do action B"'

# 3 Usability

It is difficult to find a common definition of quality for all kinds of products used by human beings, but almost all existing definitions include the "ease of use" as a big factor of influence. The ease of use, efficiency, comfort, safety and satisfaction with which users perform their tasks on a product are the main issues usability refers to ([23]).
Like cognitive science, usability engineering is also a very young scientific field. Although the usability of human tools or machines were evolutionary increased, the consciousness for usability came up with the appearance of more complex machines in WWII. Industrial psychologists such as John Flanagan discovered that the operator performance could be dramatically improved by reducing the amount of buttons, knobs, switches and control panels in fighter aircrafts. Since the 1980s usability became more important with the broadening of computers in nearly each domain of human life and therefore with the continuous growing amount of human computer interfaces. These interfaces are no more only made for experts like pilots of air fighters who absolved a long training phase and should be usable for a wider variety of users.

## 3.1 Usability Evaluation

The usability of computer systems can be improved by uncovering usability problems with the help of usability evaluation (UE) methods.

> *A usability problem occurs when aspects of a system make a task completion of a user with adequate domain knowledge inefficient, uncomfortable, cumbersome or impossible in the context of use the system was made for.*
>
> (translated from [20], page 22)

These problems can be categorized by their effect on the system and their openness (open or hidden). The effect could be fatal or harmless, whereas fatal problems are not easier to find than harmless ones. Open problems are obvious and can simply be found, e.g. buttons labeled with foreign words. Hidden problems, on the other hand, require more effort. The main classes of UE methods are given in the following.

- **Design Guidelines**
  Design Guidelines are often excluded in such UE method listings because they are more directives for interface designers than evaluation methods for usability experts. These guidelines can be general (compare [6] or [5]) or related to special interface contexts, e.g. design patterns of concerns.

- **Inquiry/Usability Tests**
  Usability tests, or inquiries in general, produce data from user interactions with real systems. These are loggings like the user's menu navigation, the amount of keystrokes or video recording. Famous tests are the Think Aloud Protocol ([20]) where the user is asked to think loud during his task completion in a way the observer is able to understand how the user thinks about the problems he has in accomplishing the task. Therefore, the focus is based on the user himself, whereas other tests are focused on the system. The extracted data of both methodologies is often enhanced by user questionnaires.

- **Inspection**
  Contrary to the design guidelines for designers, inspections are a kind of guidelines for usability evaluators analyzing a real system or a system description. In general, inspections can be divided into heuristic evaluation and Walkthrough processes ([20]). Heuristic evaluations are feature based, whereby usability experts inspect a user interface by applying a set of heuristics, broad guidelines that are generally relevant, e.g visibility or consistency and standards. Walkthroughs are based on user goals. A group of evaluators step through tasks, evaluating at each step how difficult it is for the user to identify and operate the interface element, most relevant to his/her current subgoal and how clearly the system provides feedback to that action. The most representative one is the Cognitive Walkthrough (CWT).

- **Analytical Modeling**
  Analytical Modeling belongs to the most complex UE methods. Usability experts define models of users and interfaces for quantitative prediction of user performance. The user model can be a simple human-like processor (e.g MHP, Keystroke Level Model ([23])) or complex GOMS[5] models. GOMS represents a family of techniques for modeling and representing the knowledge which is necessary for a person for task completions. These techniques can be expanded to NGOMSL[6] or CPM-GOMS[7] ([23]), depending on the focus of the evaluation.

[5] Goals, Operators, Methods, and Selection Rules
[6] Natural GOMS Language
[7] Cognitive Perceptual Motor GOMS

## 3.2 Automated Usability Evaluation

Although UE improves the usability of interfaces, the effort and required time could produce enormous costs, and overall, methods such as inspections or analytical modeling are subjective. The usability evaluator is well informed about the systems functionality and tries to analyze the system from the user's point of view. His conclusions are based on assumptions whereby the output of different evaluators, who are analyzing the same interface, often differs.
An automation of these methods reduces the disadvantages. Automated usability evaluation (AUE) can be categorized by its testing level like capture, analysis or critique and the effort a usability expert has ([20, 23]). The automatic capture of interface usage is one of the simplest AUE methods and are used during inquiries and usability tests. The needed effort of these capture methods is very low and could be performed with tools like UsAGE ([23]) or remotely with KALDI ([7, 23]) or NIST([23]). The collected usage-data can be evaluated with additional analysis tools like DRUM ([35]), the MIKE UIMS ([17, 23]) or AMME ([23, 44]).
Advantages of AUE are undeniable, but Ivory and Hearst pointed out that only 36% of all UE methods are automated ([23]), and almost all of these AE-tools are just analyzing logs of WIMP[8] like system interactions. Furthermore, all UE methods are used by usability experts and not by the interface designers. Therefore, a prototype of the interface, or the system itself, often has to be implemented.
Redesigning interfaces after negative UE results might be very expensive and could be avoided when the evaluation process is placed within the design process of the system engineer. That is the approach of MeMo, a semi-automated usability evaluation workbench, which will be presented in the following section.

[8] Window, Icon, Menu and Pointing device

# 4 MeMo

MeMo is an ongoing research project founded by Deutsche Telekom Laboratories[9] and under development of DAI Labor[10] and DFKI[11]. The workbench brings usability expertise into the design process of system engineers and is simulating a user who interacts with the system model that was developed by the engineer. MeMo is explained more detailed in this section because it gives an important basic overview of the UE process for a further understanding of how the cognitive processes are embedded within the mentioned user model.

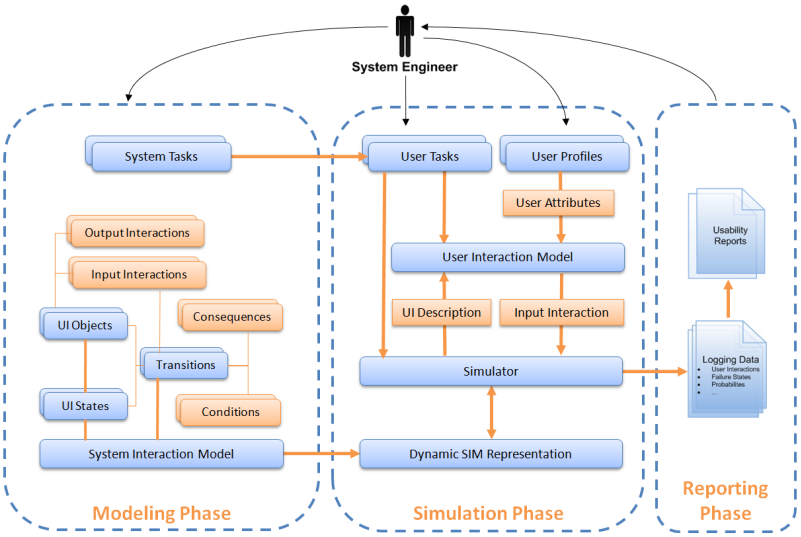



Figure 2: Work flow of a system engineer working with MeMo.

Figure 2 illustrates a raw work flow of a system engineer who is working with MeMo. The flow is divided into a Modeling, a Simulation and a Reporting Phase. In the Modeling Phase the designer defines the System Interaction Model (SIM) and the System Tasks. These are the basis for the evaluation carried out in the

[9] Deutsche Telekom Laboratories, Berlin, Germany
[10] Institute for Distributed Artificial Intelligence, Berlin, Germany
[11] Deutsches Institut für Künstliche Intelligenz (German Research Center for Artificial Intelligence), Saarbrücken, Germany

Simulation Phase in which the designer additionally selects one or more User Profiles and defines specific User Tasks. During the simulation process the Simulator generates logging data for the following evaluation process in the Reporting Phase. These three phases are explained in detail in the following.

## 4.1 Modeling Phase

As presented in figure 2 on page 11, two elements of the Modeling Phase are connected with elements in the Simulation Phase, the System Tasks and the SIM. The SIM represents a State Machine with UI States and Transitions. Several UI States are connected via Transitions, whereas each Transition contains conditions and consequences. A Condition is a term that must be evaluated to *TRUE* for an execution of the Transition, and consequences are responsible for changes to the SIM during the execution. UI States represent everything that is involved in the communication between the user and the system. Consequently, the contained communication objects of a UI State (e.g. dialogs or buttons) are called User Interface Objects (UI Objects). Each UI Object contains several interactions, e.g. a button can be used with a left click as input interaction, whereas the caption of the button would be the output interaction. A change of the SIM with a Transition refers to an input interaction performed by the user. Thus, each Transition contains an input interaction as well, whereas different Transitions might contain the same interaction. In this case, the conditions of the Transitions are needed to be different.
The complete simulation in MeMo is based on an Information Exchange Concept (IEC) and the system engineer, therefore, is asked to define the information flow himself. In general, output interactions transport information from the system to the user, e.g. labels or voice prompts, and input interactions transport information from the user to the system, e.g. text input or left click.
System Tasks define the required conditions for a successful task completion from the system's point of view and can be aggregated with sub tasks to a more complex System Task Model (STM). Basics of the STM are similar to the Concur Task Tree (CTT) notation published by Paternò ([22]), whereas only sub and parent task relations are implemented yet. The main difference between the CTT notation and the STM in MeMo are the annotated goal conditions in each System Task. Goal conditions are necessary for the Simulation Phase because the Simulator needs to know whether a task is completed or not.

## 4.2 Simulation Phase

The Simulation Phase is basically the UE process of MeMo. Before the evaluation process can be started, the engineer needs to define User Tasks. These User Tasks are directly derived from a System Task and define the UI State in which the simulation starts. Additionally, a special task description, readable for the UIM, is annotated, comparable to task descriptions in real user tests. During the simulation, the UIM tries to transport information into the system. A login task, for example, would be successfully completed with the transport of a *username* and the *password*, whereas *username* and *password* are the Information, and *"user1"* and *"pw123"* are related Information Values. The tuple of one Information and a related Information Value is declared as one Assignment. Thus, the mentioned task description contains Assignments for the UIM.
MeMo offers a variety of different User Profiles for the UE process. These profiles can be modified by the system engineer himself and consist of two types of User Attributes (UA). Static UA (SUA) express properties of a user group which can not be influenced by the simulation process such as spoken languages, the age or the technological skill. Dynamic UA (DUA) are flexible properties such as frustration, irritation or time pressure. These properties can also be ranges with a minimum and a maximum value.
As mentioned before, the automated usability evaluation of MeMo is based on the information exchange between the simulated user and the system. To be more precisely, the exchange will be conducted between the UIM and the Simulator (compare figure 2 on page 11). The Simulator is initialized with the selected User Tasks, the selected User Profiles and the amount of evaluation cycles. For each User Task a solution tree is calculated and contains numerous solution paths, whereby the shortest one is declared as optimal. A solution path consists of a sequence of interactions which are needed to be performed for the task completion. The evaluation cycle will be skipped if no solution path was found.
Concrete User Attributes are calculated because some of the attributes contained in the selected User Profiles are ranges (e.g. *age=15-25*). The UIM is initialized with new concrete UAs and a User Task in each simulation cycle. During this cycle the Simulator generates UI Descriptions. A UI Description consists of the current UI State and system Assignments, whereas only those Assignments are presented that will be transfered with the help of output interactions contained in the UI State.
After the initialization process the Simulator presents the first UI Description to the UIM which then selects an input interaction. An assigned transition for the interaction is being searched by the Simulator. In case no Transition can be found, the UIM is being presented the same UI Description again. Subsequently, the Simulator evaluates all conditions of the found Transitions. If none of these

conditions can be evaluated to *TRUE*, the same UI Description is presented to the UIM, otherwise the Simulator executes the Transition. In this case the Simulator:

1. transfers annotated Assignments (e.g. left click on a button was annotated with *choice=true*) into the system
2. transfers Assignments of the UIM (if related interaction supports such a transfer, e.g. text insertions for text fields or voice input) into the system
3. executes the contained consequences
4. switches the current UI State to the target UI State of the transition

All executed transitions, and therefore all interactions which were performed by the UIM, are compared with the calculated solution tree. Thus, the Simulator can evaluate the UIM's interaction with the SIM whether it is optimal, goal driven or wrong. Each single process during the simulation is logged by the Simulator for a final evaluation report. The task is completed by the UIM if all goal conditions of the current task can be evaluated to *TRUE*. If the task was not finished successfully, a new UI Description is generated and committed to the UIM. This process is repeated until the task is completed or the UIM gives up.

## 4.3 Reporting Phase

Subsequent to the Simulation Phase the logged data can be evaluated interactively by the system engineer in the Reporting Phase. The simulation cycles and results are sorted by User Tasks and User Groups. The engineer has the ability to trace all selected interactions, passed UI States, transfered Assignments, changes of the UIM's DUAs and more. The engineer has the possibility to accumulate these results in a short UE report.

## 4.4 Requirements for User Interaction Model

The described Simulation Phase in MeMo leads to several problems, which are needed to be solved. Figure 3 illustrates the raw working process of the UIM during an interaction selection process, which starts with the presentation of a UI Description by the Simulator. This process can be divided into four activities (compare blue boxes in figure 3).

- **Analysis of the presented UI Description**
  Special cognitive evaluation processes are involved during this analysis. One problem is the extraction of presented information and the dependency to the users skills and weaknesses. The second problem is the identification of interaction possibilities. The UIM needs to know what it can do within the current UI State. The mentioned Information Exchange Concept was

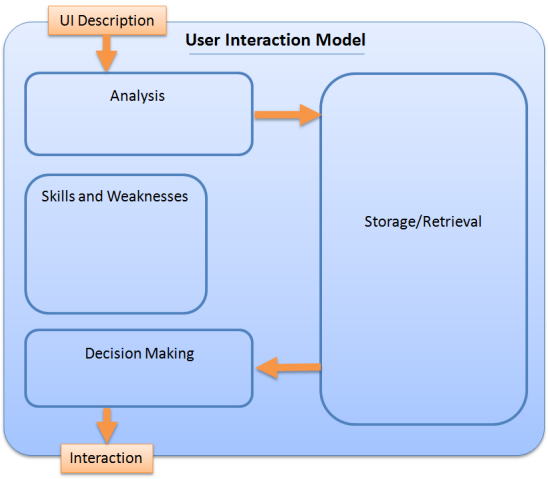



Figure 3: UIM analyzes the presented UI Description, stores the received or generated data and, finally, selects one input interaction.

developed and is, together with the analysis process itself, described in section 5.

- **Storing and retrieval of transfered Information Values**
  Once the problems of analysis are solved, new problems of information storing and retrieval occur. A kind of memory is needed to be developed which is optimized for the storage of task related information. Section 6 introduces to the human memory functionality, analyzes existing memory systems and derives ideas for an implementation.

- **Decision Making**
  Finally, the decision making process itself will be explained in section 8. But several mechanisms are involved into such a process such as intentions and emotions. A realistic simulation, therefore, is needed to simulate these dependencies as well, which is explained in section 7 first.

- **Skills and Weaknesses**
  The analysis process and the decision making process of the UIM are influenced by the users skills and weaknesses. A colorblind user, for example, has problems with the color related contrast of UI Objects. In turn, this user might have a higher probability of mixing these UI Objects. All attributes that have an influence to the UIM's behavior have to be managed by this module.

Each of the following sections explains a concrete problem, presents a solution in general, and finally, explains the implementation in the UIM. Thus, figure 3

will be completed step by step within the specific sections. Finally, the figure is completed in section 9 which gives a short summary of the UIM's working process and presents simulation results of the used examples.

# 5 Information Exchange

In general, dialogs are conducted between humans, between humans and a system or between systems. All three combinations have in common the exchange of information. The human information extraction from its environment is a complex process and many different systems are involved. It starts with the perception of signals with the help of sensor systems. These signals will be prepared by perception processors for further evaluations by the cognitive processor. Subsequently, these processes lead to an activation of relevant motor systems for a reaction to the environment.
The study of the human brain made its largest proceeding in the last two centuries when the researchers equipment made enormous improvements. Although techniques such as the Magnetic Resonance Imaging (MRI) came up first in the end of the last century, the basic understanding of the human brain was established earlier. One method is the categorization of all brain areas by the appearance of different neuron types like Brodman did in 1909 ([9]). He found 52 different areas, responsibility for different functionalities, which are still referenced by todays psychologists. Additionally, relations between these areas were found, but the complete functionality of the brain is still unknown. The main problem is caused by the mentioned missing threshold between a signal and an information (section 2.1). No psychologist can exactly define the moment when a person who sees a dog gets the information of seeing a dog. The same areas are also involved when the user just imagines to see a dog.
Researchers, working with robots, call it the Symbol-Grounding Problem ([47]). The Symbol-Grounding Problem refers to how symbols in the robot's knowledge representation relate to the real world and can be understood as relation between the sensor data and the robots understanding of the world. Robots are closed systems, and the relation to their internal data (symbols) to the outside world is rarely discussed. Especially in design processes of computer interfaces, it is somehow given that designers and potential users know what the symbols mean (e.g. the price of a product) ([47]). The problem is also predominant for linguistics because the symbols or sentences correspond in some way with subjects in the real world. A huge amount of unsolved problems are founded by the incredible complexity of the human brain. Thus, the brain is nearly impossible to model and over all, the usage for simulations of cognitive processes with the help of such

a model would cost huge efforts in modeling related dependencies.
A solution is the division of the problem into smaller parts. Newell, Card and Moran, for example, developed the Model Human Processor (figure 4), a less complex model of the human brain that predicts human reaction times. Several sensor system such as eyes or ears analyzing the environment and forward their data to special perceptual processors. These processors evaluating the data to complex information and store it in related areas of the working memory[12]. The information will be processed by the cognitive processor that generates interaction procedures for the motor processors which, subsequently, will perform these procedures.
Newell, Card and Moran found a solution for the prediction of human reaction times with an adjusted model of the human brain. Similar to this approach, a solution has to be found for the prediction of human interaction. An advantage is the fact that the UIM needs to interact with another computer model. In consequence, the output of one side could be the input for the other side and vise versa. Furthermore, the cognitive processor in the human brain gets prepared information for an evaluation and is not working on the sensor data. These given facts can be used for a model which extract information form its environment abstain from perception systems. The environment in the context of this thesis is a computer interface, modeled by the system engineer and readable by the UIM. Unfortunately, the meaning of the contained interface objects is missing and must be annotated by the engineer. As a consequence, MeMo was extended with a special Information Exchange Concept, derived from the speech act theory. The IEC enables the UIM for an interaction with the environment without explicitly modeled sensor systems.

## 5.1 Speech Acts

For thousands of years human beings have been using their voice for communication. These speak abilities were evolutionary increased from simple utterances to complex languages and are jointly responsible for the human irresistible advancement. Since today voice became our preferred communication medium. Two or more persons can share knowledge or coordinate their doings with each other, which makes a division of work much more efficient. An example gives a person $A$ with great mathematic abilities who is calculating formulas for another person $B$ who is short in time or does not have the needed skills. The transfer of the operation parameters for this calculation will be done with a speech act. Speech acts belong to the research field of linguists ([48, 33]) and can be divided into:

[12] figure 4 illustrates only the Auditory and Visual Image Store, but there exist more storages, nearly one for each sensor system

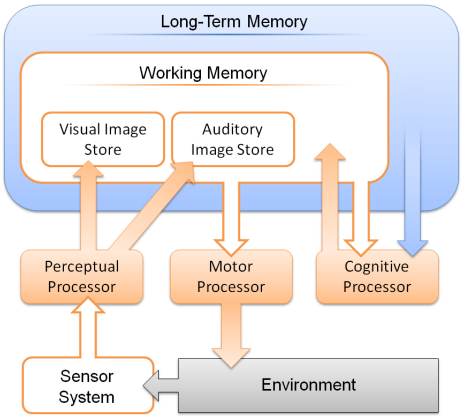



Figure 4: Model Human Processor (adapted from Card, Moran & Newell, [37])

- **locutionary act**: simple utterances, e.g. an "aw!" during wonderful fireworks
- **illocutionary act**: utterance with a communicative intention, e.g. question, request, warning or recommendation
- **perlocutionary act**: speech act for getting someone to do or realize something, e.g. persuading, convincing, scaring, enlightening or inspiring

Nowadays linguistic research is focused on the illocutionary act and often refers to the same thing "speech acts" stands for. Searle ([48]) has set up the following classification of illocutionary speech acts:

- **assertive**: speech acts committing a speaker to the truth of the expressed proposition
- **directive**: speech acts causing the hearer to take a particular action, e.g. requests, commands or advices
- **commissive**: speech acts that commit a speaker to some future action, e.g. promises or oaths
- **expressive**: speech acts that express the speaker's attitudes and emotions towards the proposition, e.g. congratulations, excuses or thanks
- **declarative**: speech acts that change the reality in accord with the proposition of the declaration, e.g. baptisms, pronouncing someone guilty or pronouncing someone husband and wife.

The speech act in the example is illocutionary because $B$ communicates parameters to $A$ who, in turn, gives back his results.

In the century of computational revolution $A$ would be a computer and $B$ transfers his parameters with the help of a computer interface. The first computers had very simple interfaces, but the communication was comparable to the conducted speech act between the two persons $A$ and $B$. A user, who wanted the system to do something, typed his wish on a keyboard, whereas he used short commands instead of long phrases. Although the interfaces have been changed, and most of the commands are done with a mouse instead of a keyboard, the principle did not change.
The similarity between natural speech acts and todays computer interactions is utilized for the IEC in MeMo. Additionally, the speech act theory is based on a concrete information exchange which is independent from sensor systems and therefore supports the avoidance of sensor system implementations. Similar approaches are used, among other things, by several information communication languages such as KQML ([50]).
Assuming a user tries to fulfill a task with the help of a computer, most interactions with the system can be understood as directive, illucutionary act. A formalization of the speech act performed in the formula example could be the following:

| | |
|---|---|
| User | *request(result)* |
| System | *request(parameters)* |
| User | *inform(parameters)* |
| System | *inform(result)* |

The information flow of parameters and results is known by a system engineer before he creates the interface. Actually a system engineer creates the interface after he has a well-defined imagination of the flow. Each UI Object he places during the Modeling Phase supports the flow with its input and output interactions. In general, the engineer does not need to annotate the flow to the UI Object interactions because a user often captures the meaning during his/her interaction with the interface. With the extension of MeMo by the IEC, the engineer is now told to annotate the information flow to the interactions of the related UI Objects.

## 5.2 Information Exchange Concept in MeMo

A communication between a user and a system can simply be described with an example of a login dialog asking for a *username* and a *password* as presented in figure 5. The user analyzes the window with his visual sensor and processor systems. The text outputs "Username" and "Password" in combination with the text input fields to their right are identified as requests. Users who know what the system means by "Username" and "Password" and also associate values with these information are able to pass the dialog. Generally a user types its associated

value for *username* in the text field right to the text "Username" and its value for *password* in the text field right of the text "Password". The knowledge where to type the values belongs to the user's experience with ordinary situations, e.g. filling a form or the reading from left to right. During the user's input, the system gives a feedback with each typed letter, and the user can control whether the value the system received is correct or not. If the user realizes a mistake (wrong typed letters) he is able to correct it. After *username* and *password* are typed into the text fields, the user clicks with the left mouse button on the button labeled with "Login". Although the user never used this dialog before, he is able to respond to the systems requests as long as he has the requested knowledge.

The term speech act refers to real spoken communications, but the dialog ex-

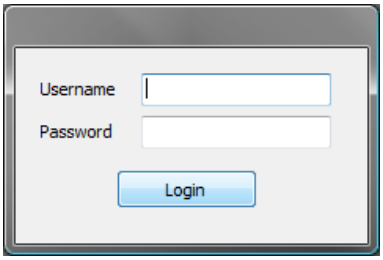



Figure 5: Login Dialog

ample can also be seen as a speech act. The system requests the user's *username* and *password* and the user informs the system by typing the values into the text fields. If this communication would be verbally accomplished, it would be a speech act as it is meant by linguists. Of course, there is a big difference between saying something and clicking somewhere, but the transfered information during a communication is still the same and the amount of questions and answers as well. Furthermore, the description of the information flow is independent from any modalities. An example is given by numerous automated telephonic supports where the user has the choice in typing a number with the keys of his telephone or by saying the number. In this case, the flow can be described as follows:

| | |
|---|---|
| System | *request(number)* |
| User | *inform(number)* |

After the engineer got an imagination of the flow, he chooses UI Objects supporting the systems part of the information exchange. As a consequence of the missing perceptual abilities of the UIM, the engineer is asked to annotate the meanings of the related interactions. The *request(number)* in the described information flow above, for example, could be annotated to the output interaction of a voice prompt *"Please type your number with the keys on your phone or just say it!"*. The input interactions of the phone keys and the microphone will be annotated

with *inform(number)*. The UIM is able to capture the meaning of the prompt and the meaning of the possible input interactions by evaluating the annotated information flow.

The login interface gives an example for the explanation of the MeMo Informa-

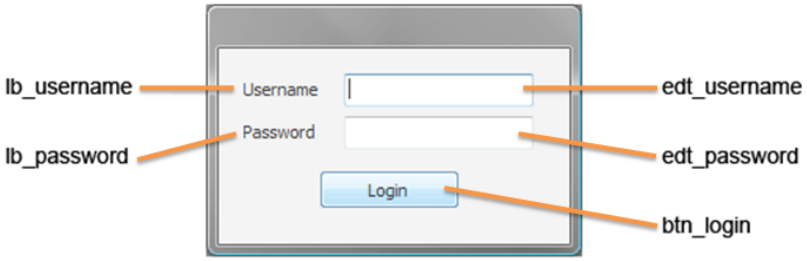


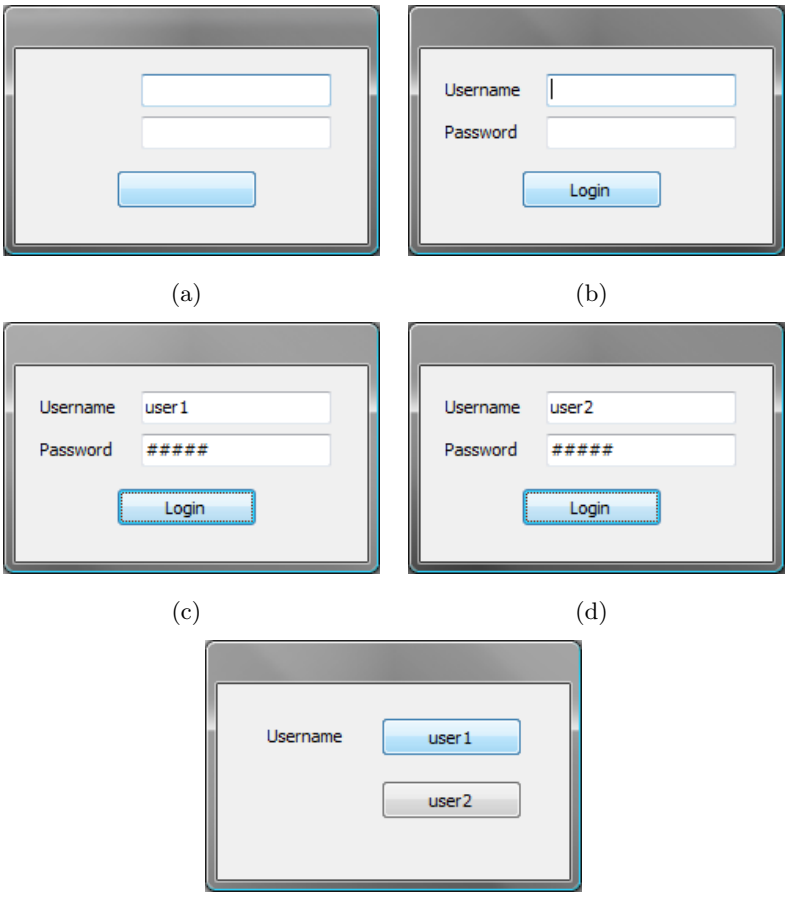

Figure 6: Login Dialog with names of the contained UI Objects

tion Exchange Concept. For further understanding, the names of each UI Object within the login dialog are given in figure 6. As mentioned before, each input interaction supports a transfer from the user to the system and each output interaction from the system to the user. Figure 7(a) on page 23 gives an imagination of a login dialog, if the system does not communicate its request for authorization. All information the user receives during the analysis of the dialog are the possibilities for typing texts into the text fields or clicking on the button. This situation can be compared with the results of the analysis process of the UIM without any annotated information flows.

A user understands the meaning of the dialog by adding the labels *lb_username* and *lb_password*. The UIM understands it by adding the labels and the meaning of their output interactions to the dialog:

| UI Object | Interaction | Annotation |
|---|---|---|
| lb_username | TEXT_OUT | request(username) |
| lb_password | TEXT_OUT | request(password) |

In this way the UIM evaluates the output interactions of the presented dialog, and it will also evaluate the annotated requests. Consequently, it searches for input interactions which support the transfer of the requested Information Value. *Btn_login*, *edt_username* and *edt_password* contain input interactions, but none of them has an annotated information flow. Thus, the UIM has no reason to select the text input interactions. The dialog and accordingly the input interactions of the text fields could be extended by following annotations:




(a) (b) (c) (d) (e)

Figure 7: Login examples

| UI Object | Interaction | Annotation |
|---|---|---|
| edt_username | TEXT_IN | inform(username) |
| edt_password | TEXT_IN | inform(password) |

Although the User Task does not contain the commitment of the user data, the UIM now tries to transfer the Information Values by performing the interactions *TEXT_IN* of the text fields sequentially.
After the UIM performed both text input interactions, the login dialog might look like illustrated in figure 7(c). Contrary to a real user, the UIM does not get the feedback from the system about the typed values because of the missing information flow related to the output interaction of the text fields. Assuming the UIM has the Assignment *username="user1"* and will not be informed by the system with the system's Assignment *username="user2"* as illustrated in figure 7(d), it has a low chance for an error correction. Thus, the related output interactions will be extended by:

| UI Object | Interaction | Annotation |
|---|---|---|
| edt_username | TEXT_OUT | inform(username) |
| edt_password | TEXT_OUT | - |

The output interaction of *edt_password* has no annotation because the text is masked with the sign '#'. Neither a real user nor the UIM can be sure that the password was typed correctly. Of course, a real user could count the '#'-signs and compare it with the amount of signs in his password, but such a special method is not implemented for the UIM yet.
Figure 7(e) presents another type of a login dialog. The user has to click on one of the two buttons without committing a password. A real user interprets the text "Username" in combination with the buttons labeled with "user1" (*btn_user1*) and "user2" (*btn_user2*) as choice for a login. For an equal interpretation by the UIM, the information flow of the interactions is modeled like this:

| UI Object | Interaction | Annotation |
|---|---|---|
| lb_username | TEXT_OUT | request(username) |
| btn_user1 | TEXT_OUT | - |
| btn_user1 | LEFT_CLICK | inform(username=user1) |
| btn_user2 | TEXT_OUT | - |
| btn_user2 | LEFT_CLICK | inform(username=user2) |

In this situation, the UIM is not able to inform the system about the requested Information Value directly, but if the presented Assignments are equal to *username="user1"* or *username="user2"*, the UIM will prefer the input interaction of the related buttons.
With the help of the mentioned annotations the UIM is able to communicate with the system. The information flow can be used by the UIM as long as the interaction refers to the possible skills of the simulated user. A blind user, for example, can not use output interactions from the system which refer to the visual sensor system. If the UIM simulates a user with matching skills, it is now able to understand what it can do with the dialog. The simulation of further cognitive processes is based on the captured meaning of each relevant UI Object.

## 5.3 UIM Integration

Section 4.4 defined four requirements for the UIM and presents additionally an abstract figure of the UIM. The first requirement was the extraction of relevant data from the UI Description, illustrated in figure 3 as empty area labeled with "Analysis". As it is presented in figure 8 on page 26, the area turned to a concrete analysis module named Information Processing Unit (IPU). The IPU evaluates all output interactions of the current UI Description. These interactions might refer to abilities which the simulated user does not have and the evaluation of these interactions, therefore, depends on the Static UAs. As it is mentioned before, the SUAs refer to attributes of a user which will not change during the interaction with the system such as color blindness or deafness. For example: in case the UIM simulates a user with deafness, it will ignore the output interactions of voice prompts. As consequence the second blue area in figure 3 on page 15, labeled with "Skills and Weaknesses" turned to the module User Attributes which includes the submodule Static UA. The blue dashed arrow represents the influence of the User Attributes to the Perception of the IPU.

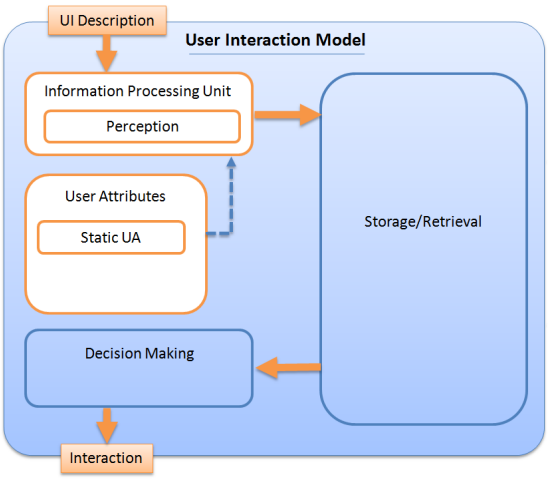



Figure 8: The UIM is extended by the IPU with a Perception module which is influenced by the User Attributes.

During the evaluation of a presented UI Description, several retrieved Assignments or requests for Information Values have to be stored in a memory module labeled with "Storage/Retrieval". This knowledge management will be explained in the next section.

# 6 Knowledge Management

Anthropologists analyzed human skulls dated to ages before the common era and found out that the human brain did not changed in size or form significantly. A conclusion that human beings in the ancient world were intelligent like modern ones seems to be plausible. Thus, the human technological advancement is rather based on knowledge than on intelligence, what makes knowledge to one of humans most important goods. It is not surprising that human beings try to preserve it since ages. The earliest known form were cuneiform writings on clay tablets followed by ink on papyrus and todays 0s and 1s on a CD. With the help of these medias, information can be transfered from one person to another, from one culture to another and even from one century into another one.
Most literature dealing with knowledge or memory do not differ between knowledge and information. The question whether "knowledge" is a more encompassing term than "information" or if both are equal concepts with different words is changing by time and usage. A common definition for knowledge is still in discussion by several cognitive scientists, but knowledge in the context of this thesis will be understood as it is defined by Karl M. Wiig[13]:

> *"Knowledge consists of facts, truths, and beliefs, perspectives and concepts, judgments and expectations, methodologies and know-how. [...] We use knowledge to determine what a specific situations means. Knowledge is applied to interpret information about the situation and to decide how to handle it."*
>
> (taken from KM-Forum "knowledge vs. information", [2])

One interpretation of this definition is that information itself can not create new information. A simple example is given in the following:
$\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ should be binary information which can be assigned with the values *TRUE* or *FALSE*, and the terms $\boldsymbol{"A \rightarrow B"}$ and $\boldsymbol{"B \rightarrow C"}$ are information containing implications which can be evaluated to *TRUE* or *FALSE*. Contrary to these terms, knowledge, additionally, contains the "Know-How" of using these information for the creation of a new implication such as the term $\boldsymbol{"A \rightarrow C"}$.
The terms used in this example are called production rules and systems, which

[13] Chairman of Knowledge Research Institute, Inc. ([1])

are using these rules, belong to the class of production systems such as ACT-R ([38, 37]) or SOAR ([3, 37]). Unfortunately, the binary evaluation to *TRUE* or *FALSE* constitutes disadvantages. Human beings often do not know exactly how to use a computer interface and their interaction might caused by beliefs or assumptions. Relations between several symbols in human mind can not formalized by productions at all time, especially when they are not based on facts. Thus, production systems are not useful for an UIM implementation. The needed modules for the UIM can be identified with a closer look to the human memory system.

## 6.1 Human Memory Systems

The basis for modern memory understanding was established by early studies of Henry M., the single most famous case study in the history of neuropsychology. The patient has a severe case of amnesia and has been observed by more than 100 researchers in more than 50 years ([15]). Today the studies with him are cornerstones in memory research. Henry M. became amnesic in 1953 after a radical and experimental operation performed by William Beecher Scoville for alleviating heavy symptoms of his epilepsy. During this operation, his amygdala, his entorhinal and perirhinal cortices and about two-thirds of his hippocampi were removed. The outcome of this treatment was twofold: his seizures decreased markedly in frequency, but Henry M. showed a severe anterograde amnesia[14] that has been persisted ([15]).
Todays psychologists are agreed that the human memory does not comprise a single unitary system and is rather an array of interacting systems, whereas each system is capable of encoding or registering information, storing it or making it available by retrieval. The human memory system can be divided into:

- **sensory memory**
  Memory that refers to the role of temporary, perceptual information storage such as iconic memory referring to visual sensory memory or echoic memory referring to its auditory equivalent.

- **short-term / working memory**
  Memory that refers to the temporary storage of knowledge necessary for performing a range of complex tasks such as long-term learning, reasoning or comprehension.

- **long-term memory**
  Memory that refers to more durable storage systems than the short-term memory. Distinctions within long-term memory include the episodic and

[14] anterograde amnesia refers to an impaired capacity for new semantic learning contrary to reterograde amnesia, the missing capacity for the retrieval of learned knowledge

semantic memory, the capacity to recollect experience or the storage of world knowledge.

Short-term and long-term memory playing an important role in human knowledge storing. The case of Henry M. also showed that both memory systems belong to different areas, whereas the hippocampus plays an important role as interface between both ([15, 13, 19]).

### 6.1.1 Short-Term Memory

In 1959 Peterson & Peterson performed psychological studies to determine the duration, or volatility, of short-term memory ([13]). Their studies were based on recall experiments where probands try to recall a triplet of consonants over a delay of a few seconds during which they are required to count backwards. The duration of forgetting the triplet depended on the amount of tests. Sequences that were presented in the beginning of a test lead to little or no forgetting, whereas the probands had problems in remembering the sequences in the end of the tests. The mean duration until the neglect began was rarely more than about 30 seconds.
The capacity span of the short-term memory is also astonishing low and can be measured with the digit span, the longest sequence of numbers that can be immediately repeated back in the correct order ([13, 9]). Usually the sequence encompasses seven digits or five random letters but can be increased if the letters becomes more like natural language. The reason is that it can be broken down into memorable, typically pronounceable chunks[15] which indicates that the short-term memory interacts with other systems such as the long-term memory.

### 6.1.2 Long-Term Memory

The long-term memory is capable of storing huge amounts of information. The rapid and accurate retrieval of stored information is remarkable and dwarfs the information retrieval algorithms of computer systems, which often consist of a scan over all possible storage locations. Generally, the human memory retrieval is relatively automatic, but the recollecting of half-remembered events has elements in common with problem solving. Different persons might use different strategies in completing their memories. An example gives a person who tries to remember where his/her keys might gone lost. The person could try to remember all positions she/he was or could try to remember where she/he found the keys the last time. Both strategies might be successful.
Although Henry M. was unable to remember things for more than the duration

[15] atomic information

of his short-term memory, the motor system still had a capacity of new learning. He was told to draw a star within two lines trough a mirror. The first tries lead to bad results but the quality of his task completion increased with the amount of tries. As expected, he was unable to remember the tests after a while, but his abilities in drawing the star were still present for a long duration ([15]).
Thus, psychologists divide the long-term memory into a declarative and a procedural part. The mentioned abilities in the star drawing experiment belong to the procedural long-term memory and all memories that are consciously available belong to the declarative one which consists of an episodic and a semantic part ([13, 9]).
The episodic memory brings events in relation to the context of their happening, whereas the semantic memory refers to the system which is responsible for storing the knowledge of the world. It comprises not only the meanings of words, rather than a large amount of non verbal information with semantic relations. These relations span a huge semantic network that brings every information in a relation to each other like it is demonstrated in figure 9.
The relations between single entities of the human memory system can refer to anything, even emotions the person had during his/her learning phase. A related phenomenon is that of mood-congruent memory, whereby a happy mood will make it easier to recall happy than sad memories and vice versa ([13]).
Similar to the neglect of the short-term memory, knowledge stored in the long-term memory also fade away but with a much longer delay. Studies showed that in remembering a foreign language, good retention occurred as much as 50 years after the initial learning but with differences in the level of quality.
It is undeniable that the human memory is the most important system for HCI.

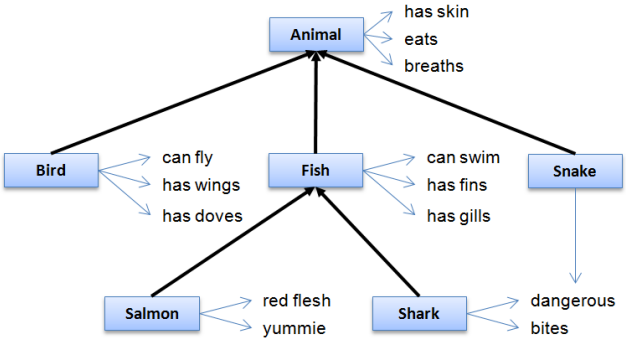



Figure 9: Example of a little semantic network in semantic memory.

A goal driven interaction without storage of temporally needed chunks is not possible. An implementation of memory systems comparable to the mentioned theories above was therefore indispensable. The development of the IEC reduced

the problem space by the procedural long-term and sensory short-term memory and only the concrete knowledge management itself has to be implemented.

## 6.2 UIM Integration

The current state of the art of knowledge management in computer systems are ontologies ([40, 54, 42, 11]). Ontologies might consist of simple sensor data or complex semantic networks similar to the presented one in figure 9, whereby each relation in the network between two or more items is well defined. Relations, based on assumptions are unsure and can be stored in a belief system like it is done in several software agent technologies. These are advantages, and over all, the UIM handles directly with Information Values during its interaction with the SIM. Thus, the values could be brought into a relation for a storage in the semantic network without any transformation.
On the other hand, the usage of ontologies is problematic for the generation of new knowledge. The relation between Information Values can not be defined by the UIM itself during its learning phase unless the system presents these relations. This would be unrealistic because either the system engineer models a semantic network for the UIM or the needed relations are stored somewhere with access for the UIM. Additionally, the UIM should simulate different users with different knowledge which means that different networks have to be created. Furthermore, human beings might associate symbols with each other without a well defined relation. "Child" and "Ball", for example, can be brought into relation without any additional information (e.g. the association is "playing").
If the relation between to entities in a semantic network is not well defined, how could these relation be stored in a semantic network such as the presented one in figure 9? The problem is similar to those of pattern recognition in Neuroscience ([18, 47]). Different input signals on one side of a black box are needed to be brought into relation with defined output signals on the other side of the box. In general, the black box is a neuronal network and the relation between input and output signals are trained connections which are not directly pronounceable. Approaches with these networks for an interaction with the environment lead to the mentioned Symbol-Grounding Problem and are not useful for the UIM implementation as well.
A combination of both technologies is needed but also unknown. One reason is the different point of view for the usage of storage systems. Knowledge stored in neuronal networks, generally, do not have to be communicated. These networks often exist in closed system such as robots or software for pattern recognition. Represented knowledge in semantic networks, on the other hand, is used to be shared by different software agents or human beings. It would be problematic to communicate chunks without well defined relations and a common concept of

storage. A combination of both technologies are neither adequate for one nor for the other side.
Anyway, the combination of the mentioned technologies is required but not difficult to implement because the values for the storage system are still present (e.g. Information Values, User Task or the labels of Navigation Objects). Additionally, fuzzy associations can be generated during successful task completions (e.g. all labels of used Navigation Objects can be brought into a relation to the User Task). Learning caused by negative task completions is disregarded. On one hand, personal experiments of the author with such an approach failed in past and on the second hand, a task completion includes an array of interactions. The UIM needs to know, which interaction was the wrong one that leads to failures, but the fuzzy way of data storing impedes a concrete analysis. Additionally, usability evaluation is based on the assumption, that a system is unusable in case that a task could not be completed by a user. The UIM would learn to use an unusable system which, in turn, adulterates the UE process.
In general, Information in MeMo are a kind of slots for Information Values comparable with variables in the system, e.g. if the system asks for a value for a variable it can be understood as a request for a specific Information Value. For a respond to the request, the UIM is needed to have the requested value contained in its Data Storage, like it is explained in section 5.2. Similar to the human memory system, the Data Storage is divided into:

- **Knowledge Base**
  As it might be expected by the name, the Knowledge Base offers the basis for the UIM's knowledge management and consists of a fuzzy semantic network. Each included fact, or information, can be related to anything that is included within the network as well, comparable to the human semantic long-term memory. The Knowledge Base includes three knowledge types: Task Knowledge, Domain Knowledge and General Knowledge. These types are only meta references for specific contents of the Knowledge Base needed for the usability evaluation.

- **Working Memory**
  The function of the Working Memory is comparable with those of the human short-term memory. It temporary stores knowledge necessary for performing a range of complex tasks such as long-term learning or reasoning.

- **Mental Model**
  A mental model represents a discrete mental representation of how a system might react and is, therefore, a kind of belief system. It is contained in a single system of the human memory and is rather distributed over several systems. In the context of usability evaluation, the system engineer needs

an overview of the UIM's imagination of the system it interacts with. These imaginations are stored in the Mental Model.

Figure 10 presents the extension to the UIM model. The IPU stores chunks in the Data Storage which are used, in turn, by the module responsible for decision making. The Knowledge Base and the Working Memory are implemented, whereas the Mental Model is still in development.

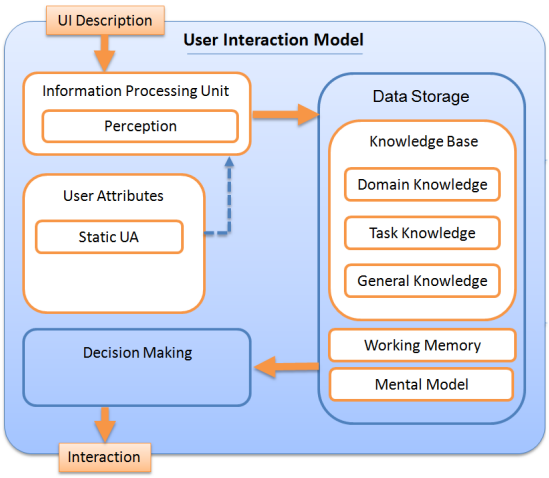



Figure 10: The UIM is extended by a Data Storage that consists of a Knowledge Base, a Working Memory and a Mental Model of the SIM.

All memory systems of the UIM will be described in detail in the following.

### 6.2.1 Knowledge Base

A usability evaluator, who is testing a system, tells his proband what he/she has to do during the test. In other words, the proband gets some information he/she needs for fulfilling the task. A simulated user also needs some information about what he has to do. Therefore, the engineer defines User Tasks for the simulation which provide these information to the UIM. The functionality of the User Tasks, and the knowledge management of the UIM in general, will be explained with the help of the program Notepad[16], which is presented by figure 11.

Assuming an usability evaluator analyzes the program and wants an user to print the typed text. He will give the user a command such as: *"Please print your text on PrinterA"*. Naturally, the UIM does not understand such a sentence, but it is

[16]Notepad is a program distributed with Microsoft Windows OS and supports several tasks such as writing, editing or printing texts.

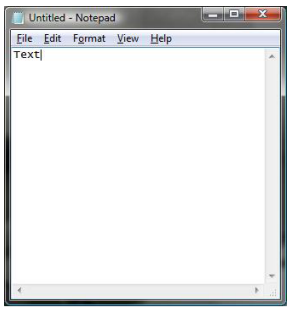
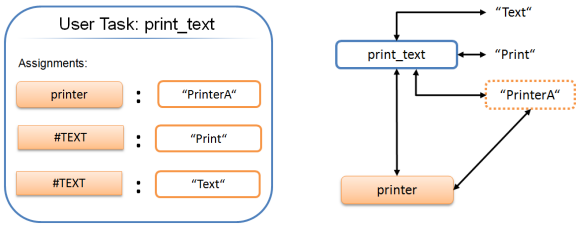



Figure 11: Screenshot of Notepad

able to communicate with the system model using speech acts. As mentioned before, the IEC supports the transfer of Information Values and the system engineer is told to encode the sentence into Assignments as it is presented by figure 12(a). Before the simulation starts, all Assignments of the User Task ("print_text") will be stored as Task Knowledge in the Knowledge Base of the UIM (fig. 12(b)). The colors of the chunks in figure 12(b) symbolize the relations to the items from figure 12(a). The blue bordered chunk represents the User Task *"print_text"*, the orange chunk represents the Information *"printer"* and the last three chunks represent texts, whereas *"PrinterA"* is an Information Value for *"printer"*.
The task *"print_text"* can not be completed by one single user interaction (compare figure 11). All the user can do is typing a text or clicking on one of the five Navigation Objects *"File"*, *"Edit"*, *"Format"*, *"View"* and *"Help"*. Contrary to the UIM, a real user got more hints than the printers name for this choice. He/she compares each label with keywords such as *"Print"* and *"Text"*, which are contained in the textual command he/she got. These keywords are needed by the UIM as well, because it has the same problem in choosing one of the five Navigation Objects. The Assignments (*#TEXT*, *"Print"*) and (*#TEXT*, *"Text"*) are textual descriptions, comparable to the task description for the user. Therefore, the keywords are stored in the Knowledge Base as textual items with unitary relations to the User Tasks *"print_text"*. *"PrinterA"* in figure 12(b) is stored as Information Value but can also be interpreted as textual item with relations to the User Task. Additionally, *"PrinterA"* has a relation to the stored Information *printer*.

Figure 13(a) and 13(b) demonstrate the interaction flow for a successful task completion (steps 1-4). It is assumed that the UIM performs a *LEFT_CLICK* on the menu item *"File"* (1). An answer to the question, how the UIM knows that it has to use this menu item, will be given in section 8. The current section only describes the functionality of the knowledge management.
After a *LEFT_CLICK* on the menu item *"File"*, new Navigation Objects appear.




(a) (b)

Figure 12: (a) Assignments defined by engineer. (b) Representation in Knowledge Base

Neither a possibility for a transfer of the value *"PrinterA"* nor an inform by the system about the current value of *printer* can be identified, and the UIM has to navigate again.

One of the new Navigation Objects is labeled with "Print" (2), which matches well with the keyword *"Print"* in the Knowledge Base. The relation between *"Print"* and the User Task (and indirectly the Information *printer*) causes the UIM to perform the related input interaction of the menu item "Print". As a consequence, a print dialog appears (13(b)), which, among other things, informs the UIM about *"PrinterB"* as current value for *printer* (figure 13(b)). Because of the mismatch between the system value and the value in the Knowledge Base, the UIM transfers the value *"PrinterA"* with the help of a list interaction (by selecting the list item labeled with "PrinterA" (3)) and remembers the transfer in its Working Memory.

Whether the task is completed, by setting up the printer, or not, depends on the task goal conditions defined by the engineer. Normally a printing task is completed by clicking the button *"Print"* (4). This final step is no problem for the UIM. The Information Value *"PrinterA"* is transfered, and one of the presented Navigation Objects is labeled with "Print". Therefore, the UIM performs the related interaction *LEFT_CLICK* on the print button, and the task is completed. During its interaction in the given example, the UIM learned a new text item *"File"* that is related, somehow, to the Information *printer*, or the printing task in general (compare figure 13(c)). The relation between the keywords *"Print"* and *"File"* is higher scored than others because these keywords were important for a successful task completion. Next time the UIM tries to print texts with a model of Notepad, or another program, it prefers rather Navigation Objects labeled with "Print" or "File" than others.

The knowledge representation in figure 12(b) was generated with the Assignments of the User Task and belong to the Task Knowledge. Because of the absence of other knowledge items in the Knowledge Base after the initialization, the Knowl-

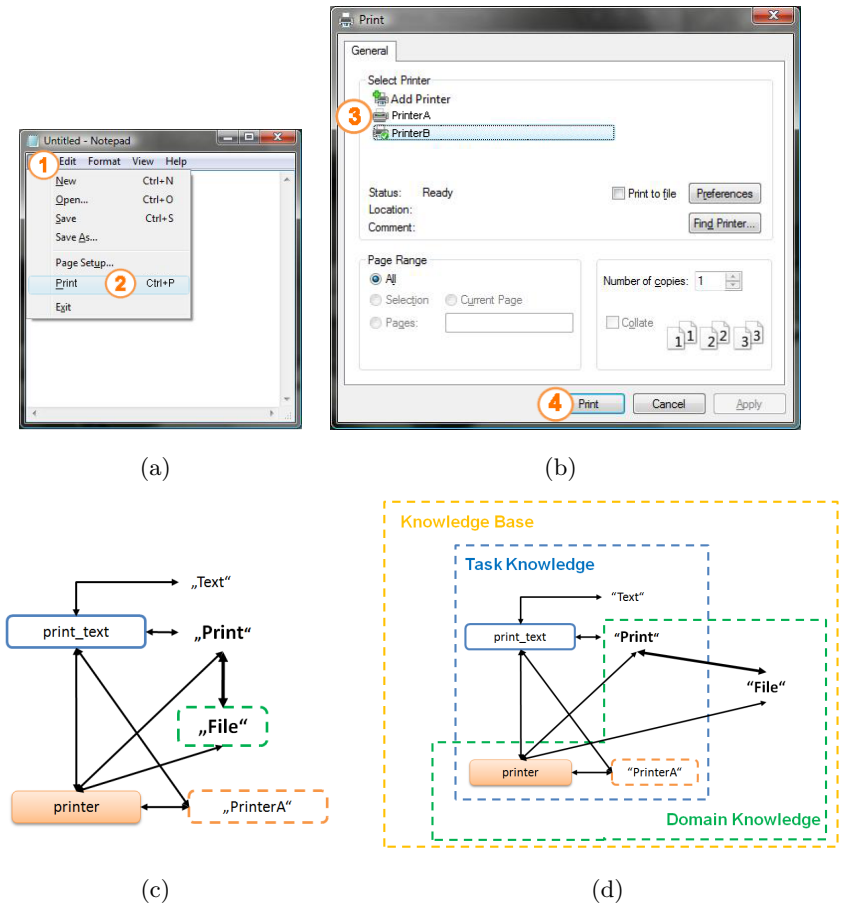



Figure 13: (a) and (b) presenting the needed steps for printing a text with Notepad. (c) Knowledge Base after successful task completion (d) Partitioning of the Knowledge Base

edge Base includes only Task Knowledge. The new learned keyword *"File"* and the found relations belong not to the Task Knowledge. The UIM learned it during its interaction with the system model and, therefore, in a domain context. As a consequence, the new generated knowledge is defined as Domain Knowledge. Figure 13(c) presents the new semantic network after the task completion, whereas figure 13(d) presents the division of this network into Domain and Task Knowledge.

Figure 14 presents the internal structure of the Knowledge Base after the task

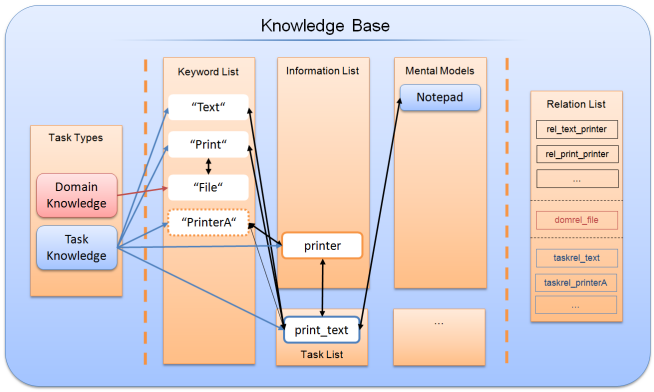



Figure 14: Internal structure of the Knowledge Base

completion and is divided into three parts. The middle part represents the semantic network itself. It consists of several item lists. The Keyword List, for example, contains all textual items, the Information List all known Information and the Task List all known tasks. The black arrows are relations between single chunks, whereas the blue arrows mark chunks as Task Knowledge and the magenta arrow marks them as Domain Knowledge.

The functionality of the Domain Knowledge is derived from the idea of analytical modeling procedures of usability evaluators (section 3) which are using GOMS models for predictions about what the user has to know for its interaction. The opposite view presents information about what the UIM learns during its interaction, or in other words, what specific Domain Knowledge the UIM generates and how fast it does that. The information is also relevant for the system engineer who is working with MeMo. The Domain Knowledge gives him the needed information whether the task is complicated or not.

Each relation in figure 14 is additionally stored in a Relation List (in the right of the figure). This is a technical need for load and save functionalities of the Knowledge Base. The list labeled with "..." symbolize the extensibility of the Knowledge Base, whereby any kind of chunks can be stored. These lists are similar to the areas of the human brain such as the visual cortex for visual pattern recognition

or the primary auditory cortex for the recognition of music ([19]). Similar activation are observable by retrieval. A real person who is trying to remember a song (sound pattern) and subsequently pronounce it, activates analogues areas as the UIM would do.
The modeled information retrieval of the UIM uses the same technique as the human brain does. The advantage is a fast access to information which are related to one chunk such as the retrieval of "Ball" in relation to "Child" or "Computer" in relation to "Mouse". The disadvantages, on the other side, are also modeled. One UIM version could associate "Cat" with "Mouse" instead of "Child", which depends on the experiences the UIM made. Conveniently, this disadvantage is a feature for the UE approach of MeMo. As it is mentioned before, most usability problems are caused by different point of views between the system engineer and the user, e.g. a user might expect, that he can give answers into a microphone if the system asks him something by its speakers.
Another disadvantage is the exponential complexity of the Knowledge Base. Each chunk might have several relations to other chunks which, in turn, also have several relations. Therefore, each relation contains a score. The higher the score, the higher the association between two chunks, e.g. "Ball" and "Child" has a higher score than "Ball" and "Work" (for the author). The score solves the problem of the relation differentiation but does not reduce the complexity of the Knowledge Base. The Knowledge Base is not readable for the system engineer and not structured like the ontology presented by figure 9 on page 30. Another module is needed for an adequate evaluation of the stored data, the Working Memory.

### 6.2.2 Working Memory

During the print example in the previous section, the UIM learned a new text item *"File"*, its relations to the text item *"Print"* and its relation to the Information *printer*. The Working Memory is the component of the UIM which provides the learning process. The Knowledge Base itself is only responsible for the storage and the retrieval of contained data, whereas the Working Memory is responsible for the evaluation and generation of new knowledge.
The processes of the Working Memory will be explained with the print example as well. Figure 15(a) illustrates the initialization of the Working Memory, which contains several sub modules, shown as orange boxes:

- **Task**
  The Task list contains the current User Tasks the UIM wants to fulfill and has the functionality of a FIFO stack[17]. New sub tasks might have to be

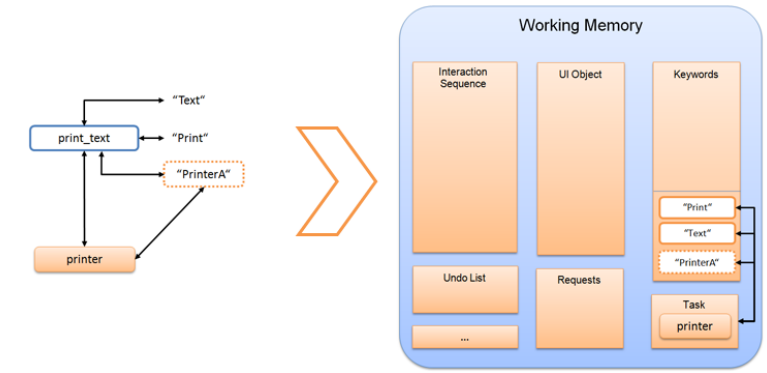
[17] FIFO stands for First In First Out and means that the contained items can only be dropped in the opposite order of their storing

fulfilled first and will also be stored. The Working Memory has only access to the last stored User Task.

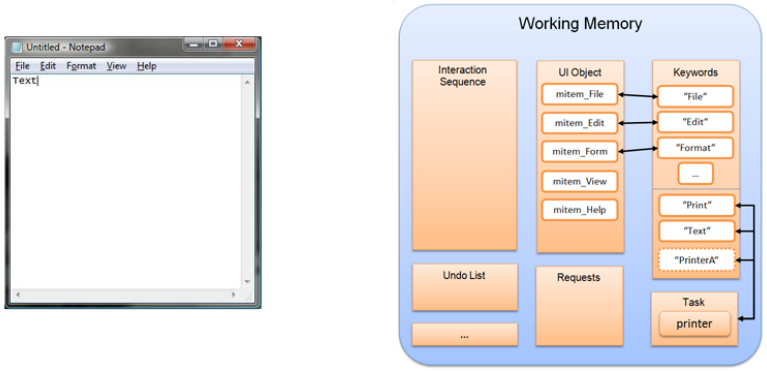
- **Keywords**
  The Keyword list stores every text item the UIM is confronted with.
- **UI Objects**
  The UI Object list stores the used or recognized UI Objects. Their labels are stored as text items in the Keyword list.
- **Requests**
  All requested Information Values that were recognized by the IPU are stored in the Request list. The UIM tries to respond to these requests, especially if the related Information belongs to one of the Information of the User Task.
- **Interaction Sequence**
  The performed interactions and the related UI Object are stored in this list. With the help of the list the UIM is able to remember recently performed interaction paths. After a successful task completion, the Interaction Sequence list provides the search for new relations and items which can be stored into the Knowledge Base.
- **Undo List**
  Mismatches between system Assignments and Assignments of the UIM are stored in the Undo List. The UIM tries to correct the Information Value of the system Assignments.
- **...**
  The Working Memory might be extended by more sub modules which are not listed yet.

Comparable to the human working memory, all described list contents have a probability of volatility. Items which are not contained in the Knowledge Base are fading away much faster than others.

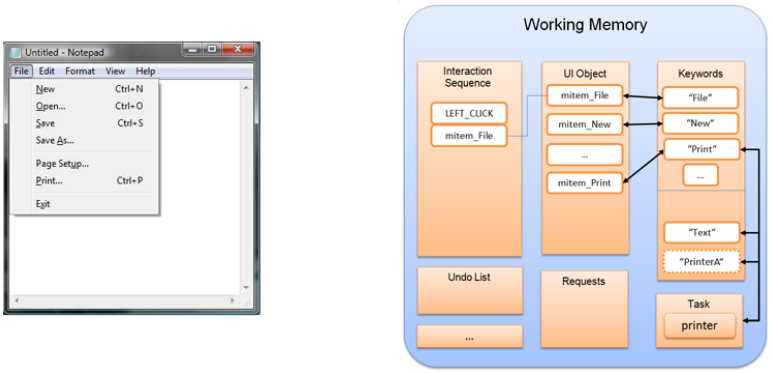
After the initialization, the Keyword list contains the text items which are related to the User Task *print_text* in the Knowledge Base. In the Simulation Phase, the Simulator commits the first UI Description to the UIM as presented in 15(b). During the evaluation of this UI Description by the IPU, all contained UI Objects are saved in the UI Object list, whereby the labels of these objects are stored in the Keyword list separately. Again, it is assumed that the UIM selects the menu item "File". Its interaction with the menu item is stored in the Interaction Sequence list of the Working Memory as illustrated in figure 15(c).
The UI Object list will be extended by the new appeared menu items and the Keyword list by their labels. In this situation, a direct relation between the label "Print" and the text item "Print" of the UIM knowledge was identified. The




(a)

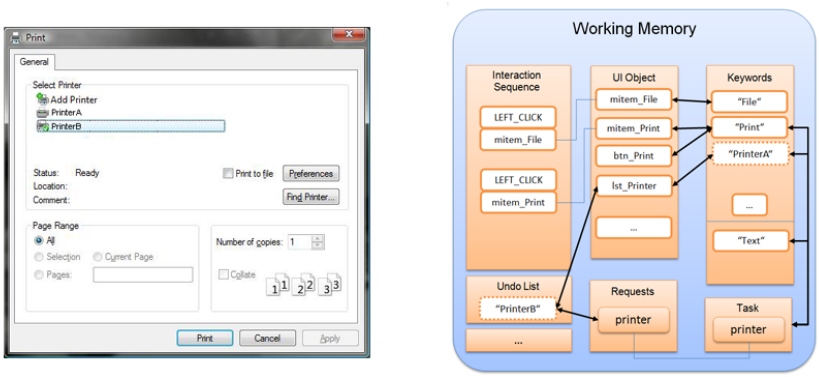



(b)

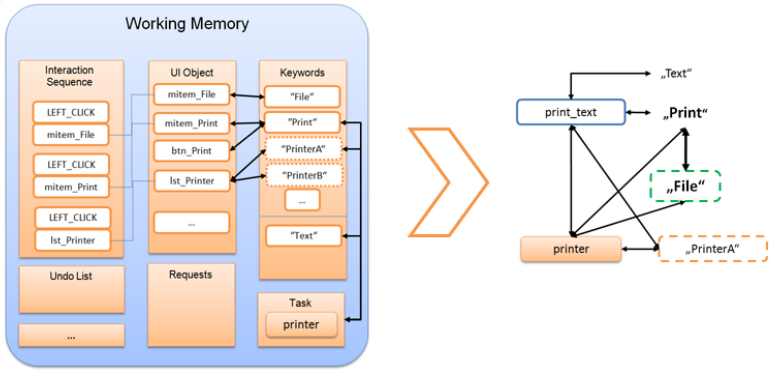



(c)

Figure 15: Working processes of the Working Memory during the UIM's interaction in the print example




(a)




(b)

Figure 16: Working processes of the Working Memory during the UIM's interaction in the print example

UIM, therefore, uses the menu item "Print" and the printer dialog appears subsequently.

The performed interaction and the related UI Object will be stored in the Interaction Sequence list. As presented in figure 16(a), the system informs the UIM about the current selected *printer* (*"PrinterB"*). Because of the mismatch between the system's Information Value "PrinterB" and the UIM's Information Value "PrinterA", the UIM stores the system value in its Undo list. The printer list in the dialog is labeled with "Select Printer". The related output interaction of the label is annotated with *request(printer)*, and each list item is annotated with direct informs (*inform(printer="PrinterA"')* and *inform(printer="PrinterB")*). The UIM prefers the list item "PrinterA" and informs the system about its printer selection. As a consequence, the Undo and Request lists are reseted.

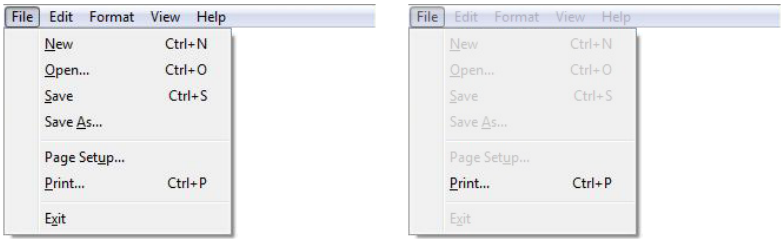
Finally, the task is completed. The Working Memory contains an Interaction Sequence list and is able to create relations between the task completion, the used

UI Objects and the labels of these UI Objects. The label of the menu item "File", for example, was important for the task completion and is not contained in the Knowledge Base. In this case, the Working Memory generates relations for a storing of the text item into the Knowledge Base. In other words, the UIM learned that "File" is somehow related to the task, especially to the text item "Print" because the menu item "Print" appeared after "File" was clicked. Thus, a relation between "Print" and "File" can be derived. Figure 16(b) presents the modification of the Knowledge Base by the Working Memory. A new item "File" is stored together with its relations to the item "Print" and the Information *printer*.

### 6.2.3 Mental Model

Working Memory and Knowledge Base are containing only facts and relations. A real user also has an imagination of how the system might react. A person, for example, who was using Notepad and tries to print a text with another program has an imagination of how the new program might work. Especially if the person recognizes a menu-item labeled with "File", the user might have an idea of another menu item labeled with "Print" that appears after he/she clicks on the item "File". Usability problems are often founded by mismatches between mental and system models, e.g. no menu-item labeled with "Print" appears after a left click on the item "File". The user is irritated and also has to find a new strategy for fulfilling the task, whereas the mental model can not be used in this case. The Mental Model is derived from that concept and represents the belief system of the UIM. It represents the UIM's imagination of the system model.
Knowledge Base and Working Memory of the UIM might lead to very complex structures and the engineer could be unable to find the usability problems of his system. Therefore, the Mental Model is a single system, based on an equal concept like the SIM is build with. The equality of both concepts makes it easier to generate the Mental Model with the help of the presented UI Descriptions. All the UIM has to do is annotating the presented UI States to an own representation of the system. Contrary to the SIM, the Mental Model scores the UI Objects and Interactions. These scores can be understood as sureness of the UIM which means that they implicate the probability of correctness. During the print example, the UIM scores the menu items "File" and "Print" higher than others. Figure 17(b) illustrates the file menu of Notepad as the UIM imagines it. Items with low scores have a lower contrast than items with high scores.

Another functionality of the Mental Model is the support for episodic relation generation. The Knowledge Base contains only semantic relations and gives an insufficient support for the generation of navigation strategies. The Mental Model, on the other hand, consists of UI Objects and Interactions. A generation of navigation strategies containing sequences of Interactions is possible. In the case




(a) (b)

Figure 17: (a) The file menu of Notepad. (b) The file menu how the UIM remembers it after one success completed print task.

the UIM has to navigate, it can generate a solution path with the help of the Mental Model. This methodology is equivalent of the problem solving algorithms of real users.
With the end of this section, processes that analyze the UI Description and manage received or generated data are derived and explained. The next sections explain approaches for the decision making process involved in the interaction selection of the UIM.

# 7 Simulation of Emotions

Up to this point, the UIM is able to analyze a presented UI Description with dependencies to weaknesses and skills of the simulated user. Furthermore, the UIM is able to store different kind of chunks with free relations into a memory system. The next requirement is the ability of choosing one interaction from the UI Description in relation to a successful task completion.
Early researches in how we make decisions, and how emotions are involved in the decision making process were even based on observations and studies of persons with brain damage such as Phineas Gage ([32]). Phineas Gage won nationwide fame in 1848 with the survival of a severe brain injury, caused by a three-foot-long, fourteen pound tamping iron that was propelled through his skull. It is determined that damage occurred to Gage's skull in three places: a relatively small area under the cheek bone where the tamping iron first impacted, the orbital bone behind the eye socket, and a very large hole where the iron rod emerged. Although the extend of damage to Gage's brain is an ongoing discussion, it is certain that the rod passed through the anterior frontal cortex.
The frontal cortex is believed to play a part in strategic processes involved in memory retrieval and executive functions, but it is still poorly understood. Frontal lobotomy[18], as it was proceeded in the last century, reduces distress but at the cost of blunting the subject's emotions, volition and personality.
Thus, todays psychologists place the cause and evaluation of emotions in the anterior frontal cortex, which is affirmed by the change of Phineas Gage's personality. People who had known him before the accident described him as hard-working, responsible, and popular with his workers. Several months after the accident he became fitful, irreverent and showed a lack of restraint. Plans for future operations were discarded very fast in turn for others appearing more feasible. His complete way of decision making changed ([32]).
Antonio Damasio used Gage's and other brain-damage cases to develop his thesis on emotion and its relationship to human activity. In his book "Descartes' Error: Emotion, Reason, and the Human Brain" ([16]) he argues that rationality stems from our emotion, and that our emotions stem from our bodily senses. The state of the mind, or feeling, is merely a reflection of the state of the body, and feeling

[18] frontal lobotomy consists of cutting the connections or destroying the complete prefrontal cortex

is an indispensable ingredient of rational thought.
A big debate is ongoing about whether thoughts that causes emotions belong to the emotion itself ([19]) or not. Thoughts about a loved person, for example, causes the emotion love, but the emotion might also occur independently of these thoughts or in relation to other things or persons. Additionally, other theorists argue that at least some emotions can be caused without any thoughts or cognitive activity, and over all, emotions might cause thoughts, which increases the complexity of this issue.
Although this debate is ongoing, it represents a major distinction between cognitive based emotions and non-cognitive emotions ([27]). This thesis is focused on cognitive based emotions, whereas theories about them are many-sided. The most active approach in the domain of emotion psychology is the appraisal theory, which consists of the subjective evaluation that occurs during the individual's encounter with significant events in the environment ([27, 19]). In other words, human beings interpret events as good and bad which subsequently causes simultaneous arousal and emotion. Therefore, cognitive activity such as judgments or evaluations, are necessary in order for several emotions to occur.
Ortony, Collins and Clore developed a theoretical cognition-based emotion model, the so called OCC model ([51]). Instead of attempting to describe every possible emotion, the OCC model works at the level of emotional cluster. These clusters are called emotion types and all emotions within a type share similar causes.

## 7.1 UIM Integration

The basic consequence of occurred emotions is the alteration of action tendencies, e.g. fear increases the tendency to flee. Many scientists invest huge effort in the research of this tendency alteration. Reilly, for example, developed an emotional agent "Em" which is based on the OCC model ([51]). His approach is based on the fact, that an agent, which interacts with the environment, causes events. These events are judged to be either pleasing or displeasing according to the agent's goals. Goals, in turn, represent anything that the agent wants, so they may be actively pursued.
The approach of Reilly matches well with the approach of the UIM, but not all emotions can be simulated within the implementation yet. Reasons are the different kinds of causes for these emotions. Hope, for example, requires the imagination of future events, whereas disappointment occurs if these events would not be caused. Thus, the implementation of the UIM is rather based on emotions which are caused by success or failures during the task completion and which have a big influence to the users behavior. These are:

- **frustration**
  Frustration is an emotion that occurs in situations where individuals are

blocked from reaching personal goals, e.g. the more important the goal, the greater the frustration, and the greater the frustration, the higher the probability of a task abortion.

- **dislike**
  Dislike might cause the same alteration of a user's behavior as frustration but has a different nature. Many individuals, for example, have aversions against computational task completion which is different to frustration because frustration could occur even if the user has no aversions.
- **joy**
  Joy decreases the probability of task abortion, contrary to frustration. Joy and frustration are mutually exclusive emotions, but they are not representing two poles of the same dimension.

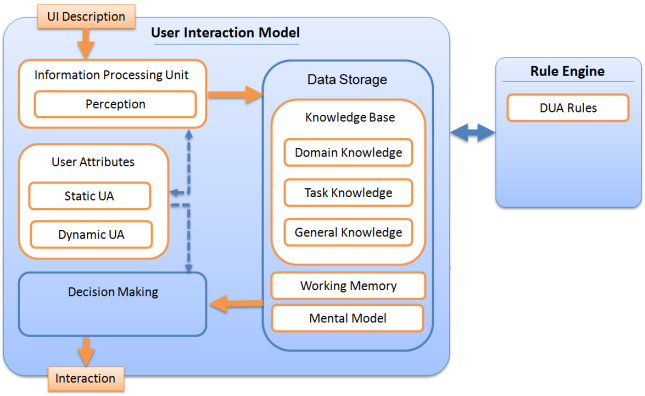



Figure 18: The UIM is extended by Dynamic UA and a Rule Engine which alters the DUA of the UIM.

Figure 18 presents the extensions of the UIM which are derived in this section. Comparable to figure 10 on page 33, DUAs are added to the User Attributes and a new module "Rule Engine" occurred. Both modules are explained in the following.

### 7.1.1 Dynamic User Attributes (DUA)

Only the ability to "feel" frustration is implemented in the UIM yet because the relation between frustration and the interaction process of the UIM is easy to model. Whenever a task is successfully completed, the level of frustration is decreased and vise versa. Additionally frequently requested but already responded Information Values increase the frustration.
The relations between frustration and the UIM interaction is defined, but the alteration itself is not. It is a fact that individuals react in different ways to events

that causes frustration. Choleric users reach the maximum level faster than calm ones, whereby the choleric characterization belongs to the Static UA of the UIM. Thus, emotions belong to Dynamic UAs with dependencies to SUA.
As mentioned before, DUAs are attributes of the UIM with abilities of change during the interaction with the system. Therefore, they also include attributes that do not represent emotions such as the awareness of time pressure.
Frustration and other emotions might also be caused by attributes of the interface, such as the appearance of foreign words or missing confirmations by the system. Additionally, combinations of these causes are also possible. The enormous complexity and individuality of the DUA alteration leads to an approach of a Rule Engine.

### 7.1.2 Rule Engine

The Rule Engine is based on Jess[19] and alters the DUA of the UIM with the help of special DUA Rules. A rule is defined as single element with a condition and a consequence part. Conditions refer to facts such as the UIM's UA, happenings in the past or attributes of UI Objects which are contained in the UI Description.
Like it is demonstrated in figure 19, the UIM commits the UI Description and the

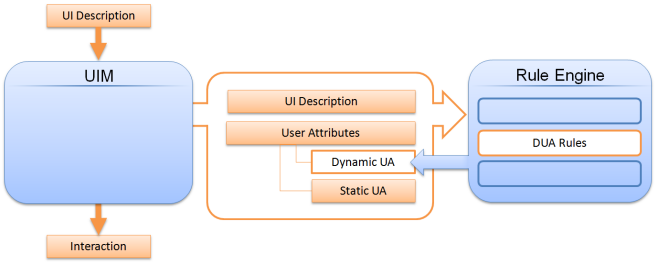



Figure 19: Rule Engine DUA Rules

UA (DUA and SUA) to the Rule Engine. DUA Rules in turn, alter only the DUA of the UIM. The blue boxes inside the Rule Engine represent other rule types which will be explained in the next section. Thus, comparable to the extensions of the UIM, figure 19 will also be extended in further sections.
In general, the usage of rules for an alteration has several advantages:

- the rules are stored in a separate directory and can be edit, added or removed
- each rule depends to special conditions which divides the complex problem of alteration into smaller parts.

[19] Jess is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill ([4])

- a single rule has only little influence, whereas the accumulation of several consequences of a rule leads to more complexity

Once the DUAs of the UIM are altered, they influence the decision making process of the UIM like it is explained in the next section.

# 8 Decision Making

Situations in real-life often require decisions between many complex and conflicting alternatives. From a cognitive perspective, decision making processes are regarded as processes integrated in the interaction with the environment, or in context to this thesis, in the interaction with a computer interface. These processes can be rational or irrational, based on assumptions, facts or biases. One rational technique, for example, is the listing of advantages and disadvantages of each option in combination with the calculation of their expected utility. This technique is based on facts and requires knowledge about the consequences of each option, which, in turn, requires information of how the system might react. When the functionality of the system is unknown, an acquiesce to another person such as an administrator, or domain expert in general, could bring the needed information. What if neither the own knowledge is adequate nor an expert is available? In this case, a person who has to choose an interaction could give up, or use a random technique such as flipping a coin or cutting a dock of playing cards. In general, random methods are mostly used, if the significance of consequences is expected as very low such as a decision between many different but preferred flavors for a small amount of scoops.
Rational decision making is easier to implement than the irrational equivalent. Several theories exist for the rational approach such as the decision tree methodology. In this approach, all decisions and their resulting system reaction will be combined in a tree structure ([14, 10]). The shortest path in the tree that leads to a solution includes the best decisions. Extensions such as low-cost-functions can be added and evaluated by cost-benefit algorithms ([45, 46]).
One of the biggest problems for a simulation of human decision making is the irrational factor which is focused by this thesis. Emotions or other attitudes of human beings have an astonishingly high influence. Researchers such as Isabel Briggs Myers searches for dependencies to a person's decision making process in relation to his/her psychological profile. Myers developed a set of four bi-polar dimensions, called the Myers-Briggs Type Indicator (MBTI) ([19]). Each of these dimensions represent preferences of a person similar to the left or right handedness: individuals are either born with or develop certain preferred ways of thinking and acting. These preferences are: Extraversion $\longleftrightarrow$ Introversion, Sensing $\longleftrightarrow$ Intuition, Thinking $\longleftrightarrow$ Feeling and Judging $\longleftrightarrow$ Perceiving.

The indicator is frequently used in areas such as career counseling, marriage counseling, group dynamics or employee training. Most of the four dimensions are hard to measure, and they might change in time, even during an interaction with a computer interface.
Although dependencies between the decision making process and attributes in the User Profile matches well with the approach of MeMo, the profit of an implementation is relatively low, whereas an implementation approach is not documented by Myers. Similar approaches in Applied AI such the "Intuitive Decision Making" ([46]) are based on facts of system reaction. Disadvantages of fact based systems such as ACT-R, or production systems in general, are already discussed in section 6. Again the implementation needs new approaches to be developed.

## 8.1 UIM Integration

The MBTI of Myers is an indicator which allows to make predictions about how decision making processes of people are influenced by their psychological profile. The influence to the User Profile is already implemented (SUA) and the formalism of the MBTI can be used for additional rules of the Rule Engine. As mentioned before, emotions and the user's attributes rather influence the action tendency of the user than the decision making process directly. As a consequence an intentional component was developed for a representation of the UIM's tendency, the Intention.

### 8.1.1 Intention

As it is illustrated in figure 20, the Rule Engine is extended by Intention Rules for an alteration of the Intention. Comparable to figure 19 on page 48, the UIM commits the UI Description, the User Attributes, and additionally its Intention to the Rule Engine. The alteration of the Intention is influenced by the DUAs of the UIM. A frustrated user, for example, has a higher attitude for a task abortion in unexpected situations. These dependencies between the Intention and the DUA are also implemented. The mentioned frustration example can be formalized as rule, and added to the Rule Engine.

Four intentions have been implemented so far:

- *FORWARD* - The UIM tries to transfer its task knowledge to the system and therefore prefers interactions which support the transfer. In case that no interaction is preferred, the UIM tries to navigate through the SIM.
- *CANCEL* - If the UIM was navigating and believes further navigation is not possible or reasonable, it tries to navigate back (e.g. it was navigating into a menu structure and tries to go back to a higher level).

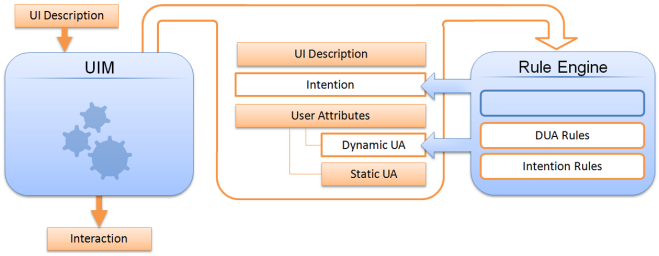



Figure 20: Extension to the Rule Engine - Intention Rules for the Intention alteration

- *HELP* - The UIM could neither find interaction for a knowledge transfer nor meaningful interactions for navigation and needs help. If no interaction can be associated with help, the intention switches to ABORT.
- *ABORT* - The UIM gives up.

*FORWARD* and *CANCEL* are only reasonable when the UIM believes it still can fulfill its task, whereas the probabilities for *HELP* and *ABORT* are increased in hopeless situations. Intentions can also be grouped by special contexts. Navigation or the wish for the transfer of Information Values also are kind of action tendencies and might belong to all Intentions of the UIM, which increases the complexity. Thus, it is for sure that a single constant value for the representation of the UIM's action tendency (Intention) is insufficient and has to be extended in future.
Up to this point, the UIM is able to capture the meaning of the output interactions of a UI Description. Furthermore, it is able to transfer Information Values into the system with the accomplishment of related input interactions. The evaluation and usage of UI Objects is related to User Attributes, derived from User Profiles. Emotions are implemented as Dynamic User Attributes and are influenced by the user performed interaction, the system reaction and at least by the user SUAs. The UIM reacts in relation to its Task Knowledge, comparable to task descriptions in real User Tests. The Knowledge Base of the UIM can be extended during the interaction process (similar to learning). The absence of the Mental Model, or (in future) the missing understanding of the SIM caused by an insufficient Mental Model, leads to the last problem that has to be solved. If the UIM is navigating, it might have no relations between the labels of the presented Navigation Objects and its knowledge. Thus, the UIM is not able to navigate in a realistic, plausible way. Thus, a secondary module for the navigation support was developed, the Keyword Engine.

### 8.1.2 Keyword Engine

If the UIM has to navigate through the SIM, it analyzes the labels of all Navigation Objects. Relations between these labels and the User Task must be found which causes a big problem. Real users found their relations with the help of an enormous knowledge base. It is impossible to model this knowledge, let alone to model the world. Information Retrieval (IR) Engines are using technologies which are dealing with the opposite problem ([30]). The World Wide Web contains a huge amount of information, and it is impossible for a user to read all websites for an information extraction. In general, a user gives a query for an information retrieval to the IR Engine, whereas the query contains a set of words the user associates with the information he wants to receive. The IR Engine generates Topic Maps for several documents from the Internet ([26, 25, 34]). A Topic represents objects identified by names, e.g. ball represents a round object children are playing with. Topic Maps contain different Topics and their amount of occurrences in a document. These occurrences will be compared with the users query. The documents with Topic Maps that matches well with the query will be forwarded to the user ([41]).

The Keyword Engine uses the methodology of these IR Engines. During the

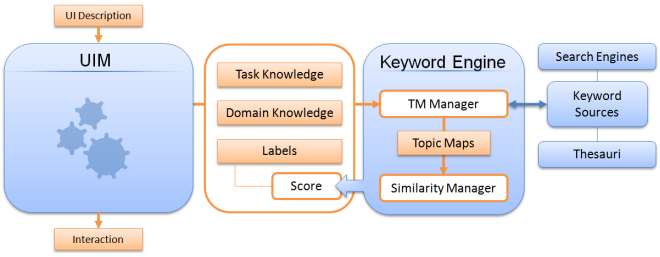



Figure 21: Keyword Engine

navigation process, the UIM generates a list of labels from all Navigation Objects and lists of relevant keywords depending on its knowledge as illustrated in figure 21. The Keyword Engine tries to score the given labels in relation to the keywords and generates Topic Maps with the help of Keyword Sources such as GermaNet ([12, 28]), EuroWordNet ([28]) or Yahoo! ([53]). The content of these Topic Maps will be compared with the list of labels. Labels that have a huge amount of occurrences in these maps will be scored higher than others. The scores are annotated by the Keyword Engine to the committed labels of the UIM. A Navigation Object with a high scored label has a big probability of semantic relation to the keywords. The UIM, therefore, prefers the Navigation Object and performs the default input interaction (e.g. *LEFT_CLICK*).

### 8.1.3 Final UIM Version

All requirements for the interaction selection are implemented so far but without the expertise of UE experts which are using a lot of formalisms while analyzing computer interfaces. These formalisms are, for example, dependencies between the interaction and the layout of the interface itself. The majority of these formalisms can be modeled as rules for an alteration of the interaction selection, comparable to the alteration of the DUAs or the Intention of the UIM. In this way, empirical data of usability experts can be brought into the decision making process of the UIM, and, therefore, into the design process of the system engineer.

For an integration of the interaction selection alteration, the UIM generates

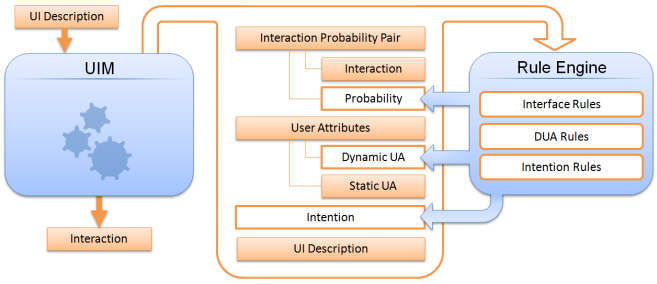



Figure 22: Final version of the Rule Engine

Interaction Probability Pairs (IPP) containing combinations of interactions and probabilities (see figure 22). These probabilities were set up by the UIM and without dependencies to the user's deficiencies. The Rule Engine analyzes the UI Description and generates facts in relation to the UI Object, the interaction to which the interaction of the IPP belongs. Each contained rule consists of a condition part and a consequence part. The condition part includes facts which will be compared with the generated facts for the UI Description, Intention and User Attributes. In case both fact lists match, the consequence part of the rule is altering the probabilities of the committed IPP.

The previous section described the requirements and the implementation of several cognitive processes. Each section extended the model of the UIM step by step and the final version is presented in figure 23 in the next section.

# 9 Simulations

The previous sections described the implementation of several cognitive processes but did not explain how they support the system engineer in the UE process, which is done by this section with the help of concrete simulation examples. For a better understanding of these simulations and the UIM's behavior, a short summary of the final UIM implementation (see figure 23) will be given first. As mentioned

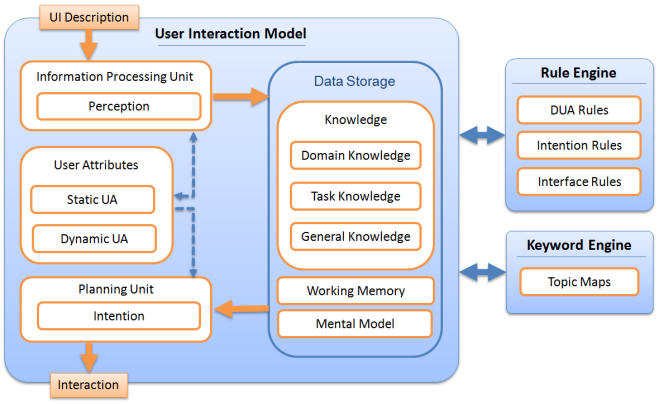



Figure 23: UIM components

in section 4, the UIM is initialized by the Simulator with a User Task and the generated Dynamic and Static User Attributes. The interaction selection process starts by committing the first UI Description, whereas all contained output interactions are analyzed by the IPU. An Information Value transfer, supported by one of these output interactions, leads to four different situations:

1. The UIM receives new Information Values of unknown Information.
2. The UIM receives an Assignment which is already contained in its Knowledge Base.
3. The UIM receives already contained Assignments but with differing Information Values.
4. The UIM receives one or more requests for an Information Value.

Situations 1 and 2 are easy to handle. Either the UIM already knows what the system is presenting or the UIM stores the presented Assignment in its Knowledge Base. Situation 3 is a mismatch between the knowledge of the UIM and

the presented Assignment by the system. In this case, the UIM can react in different ways. If the mismatch refers to the Task Knowledge, the UIM stores the wish for correction in its Working Memory, otherwise its Domain or General Knowledge will be corrected. If the requested Information Value in situation 4 is contained in the UIM's Data Storage, it is able to answer the request, e.g. if the system asked for the *username* and the Data Storage contains the Assignment *(username,"user1")*. After analyzing the output interactions, the Rule Engine is called to modify the Dynamic User Attributes according to UA modifier rules. Continuously requested Information Values from the system might increase the frustration level related to the Static UAs, e.g. a choleric person would be more frustrated than a normal one. As mentioned before, the Dynamic UAs influence the Intention of the UIM (e.g. high frustration or irritation levels increases the possibility for a task abort, time pressure reduces the probability for searching help).
After the Intention alteration, the UIM starts to evaluate the available input interactions. Assuming the Intention is *FORWARD* the UIM tries to transfer its Task Knowledge to the system and prefers interactions which support a transfer. These interactions are set up with a higher probability of selection. In case that no interaction was preferred by the IPU, the Planning Unit systematically evaluates the UI Objects for navigational functionalities (Navigation Objects) in correlation with its knowledge and increases their interaction selection probability.
If the Intention is not *FORWARD*, the UIM tries to find Navigation Objects related to its Intention, e.g. the Intention is *HELP*, it prefers buttons labeled with "?", "i" or "help". In case no interaction related to *HELP* or *CANCEL* can be identified, the Intention will be set to *ABORT* and the UIM gives up.
Finally, each interaction is set up with a probability of selection, whereas the preferred interactions have higher probabilities then the others. The Rule Engine is called to modify these probabilities. Therefore, the engine analyzes the complete UI State of the UI Description as well and generates facts for numerous attributes of the contained UI Objects, e.g. the contrast of the button labels with the button background or their size and arrangement to other UI Objects. A dice throw finally selects the interaction according to their probabilities. The selected interaction is given back, and the process will be started again until the task is completed or the UIM gives up.

## 9.1 Login

The login example (compare figure 5 on page 21) was used in section 5 for the explanation of the Information Exchange Concept. The dialog is successfully passed by typing the Information Values for *username* and *password* into the text input fields followed by a click on the button labeled with "Login".

For the simulation, a User Task *login* was defined as presented in figure 24. As expected, the UIM prefers the two input interactions of the text fields and the dice throw finally decides which of these interactions will be selected by the UIM, whereas the order has no influence on the task completion.

The dialog contains three interactions, two belong to the text fields and one to the

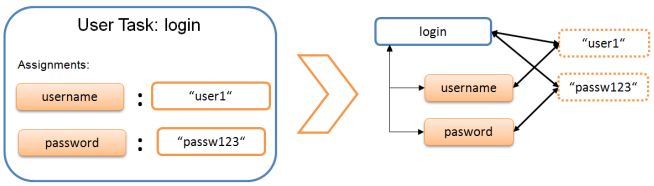



Figure 24: Assignments for the User Task *login* and the related representation in the Knowledge Base of the UIM

button. Thus, the UIM automatically uses the Navigation Object (button) after the Information Values are transfered. As a consequence, the UIM always fulfilled the task *login* without mistakes. A typical usability problem of such login dialogs is the absence of confirmations or errors whether the *username* and *password* were typed correct or not. Therefore, a probability of typing wrong letters should be implemented in future for simulating these mistakes.

## 9.2 Notepad

The knowledge management of the UIM was explained in detail with a model of Notepad. Figure 25 presents the model of Notepad and its UI States and Transitions, whereas only the menu structure and a print dialog are modeled. For printing a text on the *printer "printerA"*, a user has to navigate to the printer dialog, selecting the correct printer and perform a click on the print button (compare figure 13 on page 36).

Figure 26 presents the User Task *print_text* again. The navigation is impeded by the insufficient knowledge of the UIM because no label of the Navigation Objects in the UI State Notepad matches with the text items in the UIM's Knowledge Base.

The Keyword Engine solves the problem of missing associations between the labels of the UI Objects and the chunks of the Task Knowledge. Figure 27 presents the interaction sequences of a simulation with an enabled Keyword Engine (a) and when the engine was disabled (b). Experiments also showed, that once the UIM learned the relation between *"Print"* and *"File"* (in simulation without the Keyword Engine), it never uses wrong Navigation Objects again.

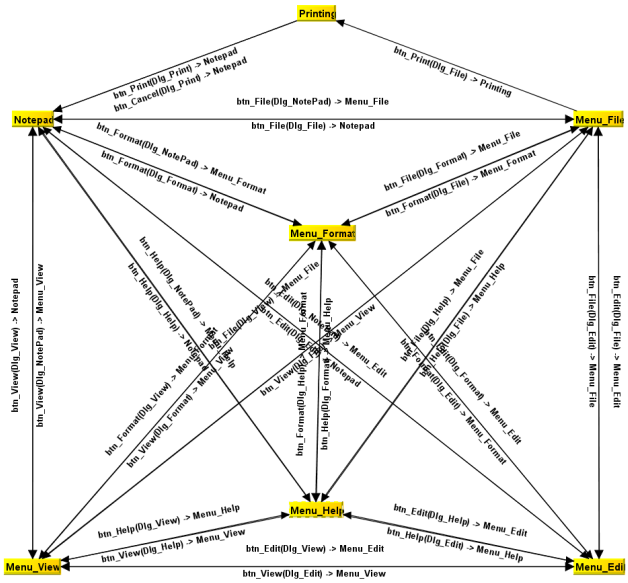



Figure 25: State chart of the the menu structure and one print dialog of Notepad.

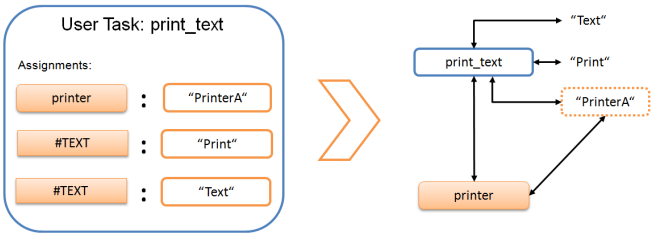
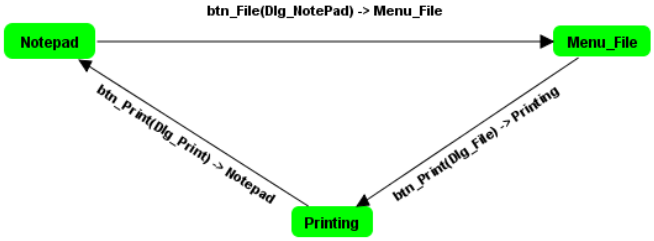
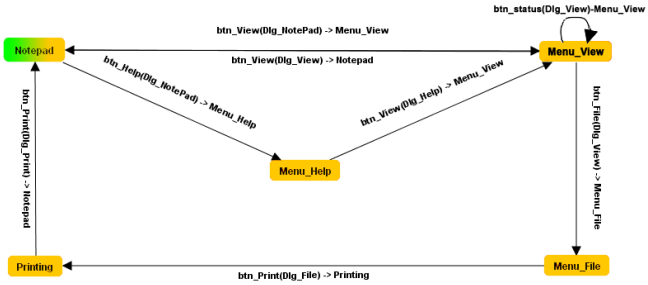



Figure 26: Assignments for the User Task *print_text* and the related representation in the Knowledge Base of the UIM




(a)




(b)

Figure 27: Interaction sequences of the UIM with enabled (a) and disabled (b) Keyword Engine

## 9.3 4 Star Cooking Assistant

The "4 Star Cooking Assistant" is an ongoing research project at the DAI Labor. It provides a user in selecting recipes, in buying the needed ingredients and finally in cooking the selected recipe. Figure 28 presents the interactions which are needed to be performed by a user for the preparation of "Birnenkuchen" (germ. for pear cake) guided by the Cooking Assistant. The single UI States and interactions are explained in the following (compare figure 28 on page 63):

1. ***Welcome***
   The UI State *Welcome* proposes the user two recipes. None of both is called "Birnenkuchen" and the user has to click on the button labeled with "Rezeptsuche starten" (germ. start recipe search) which makes the system presenting the UI State *Search*.

2. ***Search***
   In this UI State, the user has the possibility to define some filters such as the meal type or the cuisine. Subsequently or directly, he/she has to click on the button with the label "Suche starten" (germ. start search).

3. ***RecipeDetails***
   On the right side of this UI State, a list with available recipes appeared. After the user selected the list item "Birnenkuchen", a short description is given below the list.

4. ***RecipeDetails***
   The next interaction the user has to do within the UI State *RecipeDetails* is a click on the button labeled with "zur Einkaufsliste" (germ. to the shoppinglist).

5. ***PersonAmount***
   The UI State *PersonAmount* requests the amount of person the meal should be for, whereas the default amount is one. Subsequently the user has to click on "Einkaufsliste erzeugen" (germ. create shopping list) and the dialog *ShoppingList* appears.

6. ***ShoppingList***
   In this state, all ingredients are listed in the left. The user has the possibility to create a shopping list in confirming each ingredient and subsequently performing the button "Einkaufsliste verteilen" (germ. distribute shopping list) or he/she could navigate to the UI State *CookingAssistant* by performing the button labeled with "Mit Kochassisstent fortfahren" (germ. continue with cooking assistant).

7. ***CookingAssistant***
   Finally, the UI State *CookingAssistant* presents the recipe for "Birnenkuchen".

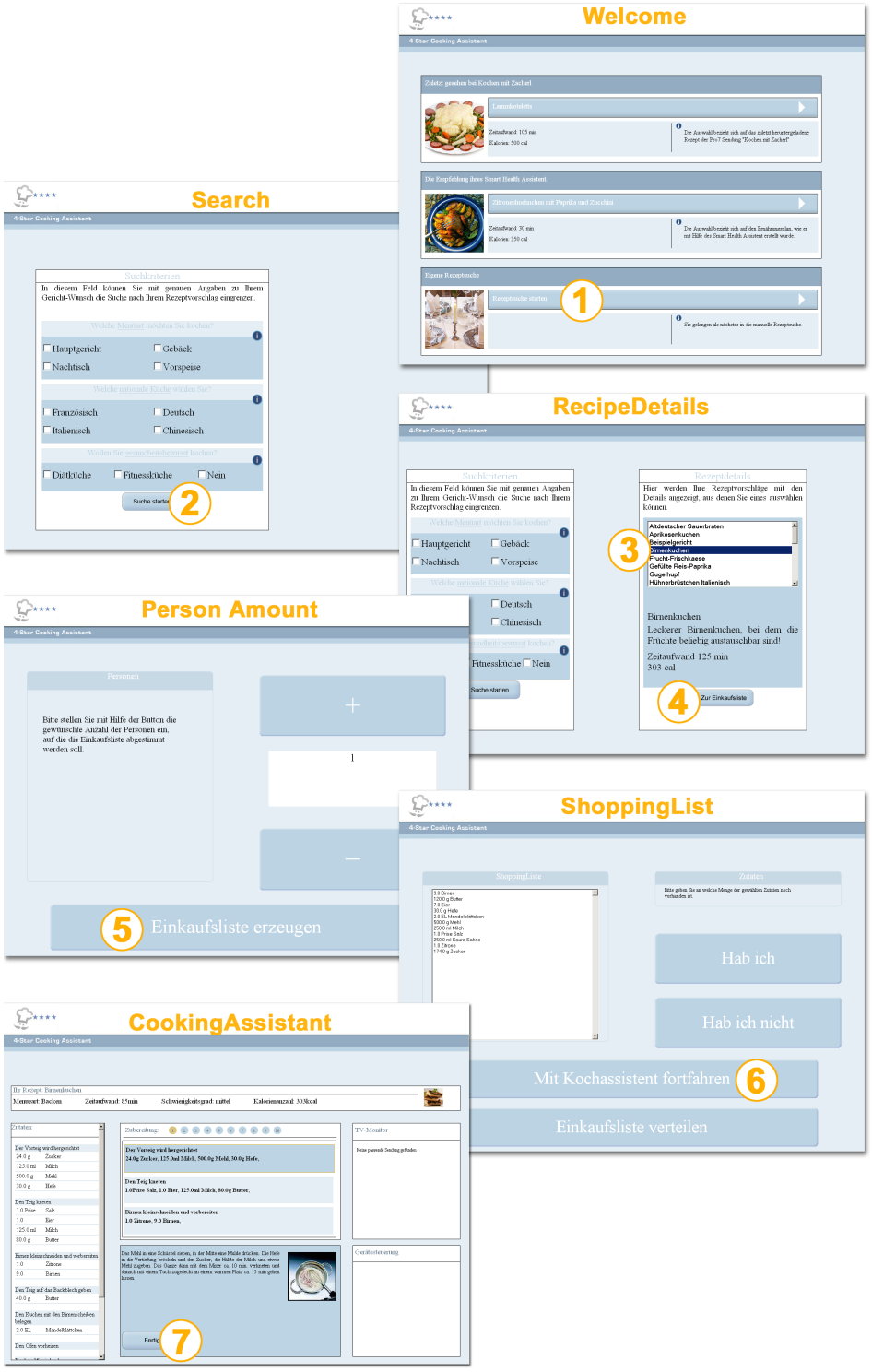



Figure 28: Interaction sequence for a successful task completion

The "4 Star Cooking Assistant" was remodeled with MeMo and several tasks could be defined for an usability evaluation in the Simulation Phase. A task description for the selection and presentation of the recipe "Birnenkuchen" in real user test could be "Please search the a recipe for 'Birnenkuchen'". An equivalent User Task for the UIM is given in figure 29 on page 64. The textual Assignments are in German because the labels of the system model are German as well.
Contrary to the previous User Tasks, long navigation sequences are needed for a task completion, and a shopping list is not mentioned in the task description. For the task completion, the UIM has no other knowledge than the presented one in figure 29.

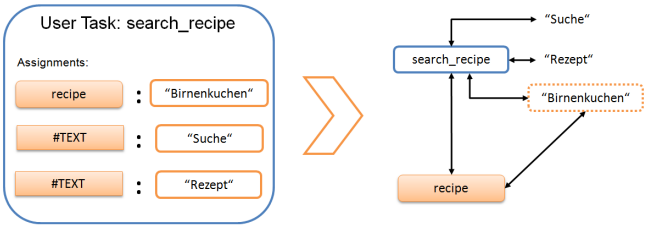



Figure 29: Assignments for the User Task *"search_recipe"* and the related representation in the Knowledge Base of the UIM

As it might be expected, the UIM had problems in navigation. The task goal was only reached in 4 of 50 simulation runs. Although this weakness is caused by the insufficient knowledge of the UIM, the impairment for the task completion could be interpreted as insufficient labeling of the Navigation Objects. In the UI State *RecipeDetails*, the UIM has to click on the button "Zur Einkaufsliste" whereas the label does not match with the User Task *search_recipe* or its contained Assignments. Even the Keyword Engine did not found high scored relations between the task related keywords and the presented labels. In this situation, the UIM searches for help, can not find some and gives up.
This behavior is not unrealistic at all, real users might also stop at this point because they do not want to buy something. Finally, an addition of a third textual Assignment *(#TEXT,"Einkaufsliste")* solved the problem.

Figure 30 presents one performed Interaction sequence by the UIM. As it is symbolized by the red-orange color of the UI State *CookingAssistant*, the UIM did not reached the task goal which is caused by an usability problem.
In the UI State *RecipeDetails*, the UIM selected the wrong recipe "Frucht-Frischkäse" (germ. fruit cream cheese) and did not recognized its mistake. The following states *PersonAmount* and *ShoppingList* do not name the selected recipe, therefore, they do not transfer the system's Information Value for the Information *recipe*. Thus, the UIM reached the state *CookingAssistant* without recognizing the wrong se-

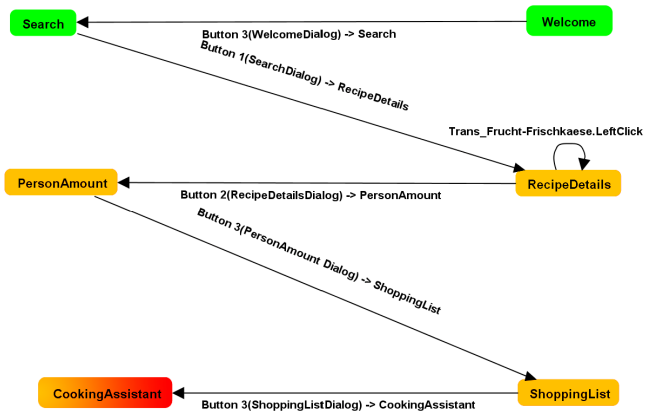



Figure 30: Interaction sequence performed by the UIM

lected recipe. In the UI State *CookingAssistant*, in turn, it recognizes the wrong selected recipe but there is only a Navigation Object labeled with "Fertig" (germ. finish). Although the Intention might have switched to *UNDO* or *CANCEL* in this situation, the UIM has no chance to correct its mistake and gives up.
The found usability problem belongs to the hidden ones and might have fatal consequences because the user might buy wrong ingredients. A simple label which informs the user about the selected recipe in the UI State *PersonAmount* and/or *ShoppingList* could avoid these consequences.

## 9.4 Voice System

The last presented simulation was done with a model of the INSPIRE system. INSPIRE is a Speech Dialog System (SDS) for controlling several devices such as lamps or TVs. A command for a device control always consists of three Information: *device*, *action* and *location*. A typical user command could be : "Please turn on the lamp in the corner". The sentence contains Information Values for all of the three Information. The named device is the lamp, therefore *"lamp"* is an Information Value for the Information *device*, *"on"* for *action* and *"in the corner"* for *location*. In the context of SDS, an Information could have different Information Values for a successful task completion. The lamp in the corner, for example, could also be defined by their light color, e.g. the blue lamp. Both values for *location* might be correct. Combination of one Information or attribute with different values are defined as Attribute Value Pair (AVP) in SDS.
Figure 31 shows the first UI State of the INSPIRE model. The blue colored element represents a prompt which has one output interaction *TRIGGER_OUTPUT*, whereas the text of the prompt itself is encoded in *informs* and *requests* as explained in section 5. The green colored elements in the dialog represent the AVPs

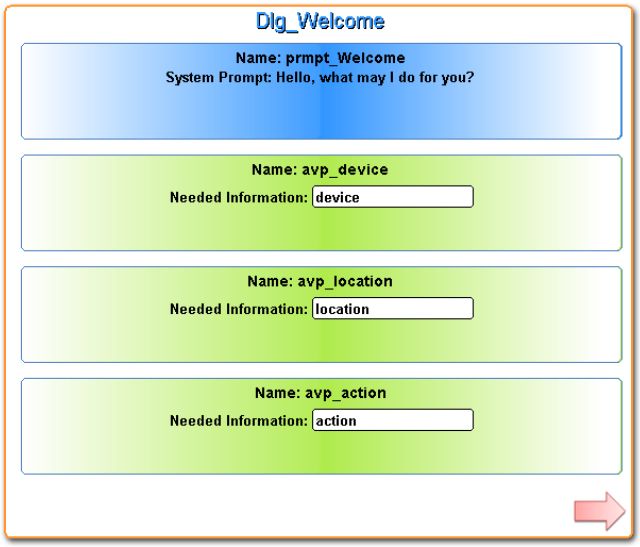



Figure 31: Interaction sequence performed by the UIM

for the Information *device*, *location* and *action*. Contrary to other UI Objects, AVPs do neither have input nor output interaction and the UIM ignores them. Thus, AVPs just represent what the system understands in the corresponding UI State. The UIM communicates with a Voice Input Object instead, a specialized class of UI Object with one input interaction *TR IGGER_INPUT* as equivalent to a microphone.
If the UIM decided to use the Voice Input Object, the simulator, subsequently, asks the UIM which Information Values it wants to transfer. These Values depend on what the system was asking or committing to the UIM before (*request* and/or *informs*). If the system, for example, asks for the *device*, the UIM transfers its corresponding value *"lamp"*.
Because DSDs often have tendencies for a misunderstanding of the users output, a filter component was implemented to simulate these weakness, whereby the UIM always believes that the Information Values it has transfered are committed correctly to the system. In case that these values were not committed correctly, the system needs to inform the UIM about the problems it has.

Figure 32 presents a little model of the INSPIRE system with the ability of controlling a lamp. The dialog of the start UI State is presented by figure 31. A simple command by the user such as "Please turn the blue lamp on!" should be evaluated in the first dialog without further interactions. For a simulation of this task, the text was encoded into Assignments as presented in figure 33.

Although the task *blue_lamp_on* could be completed in the first dialog, the filter component leads to more complex interactions between the UIM and the system

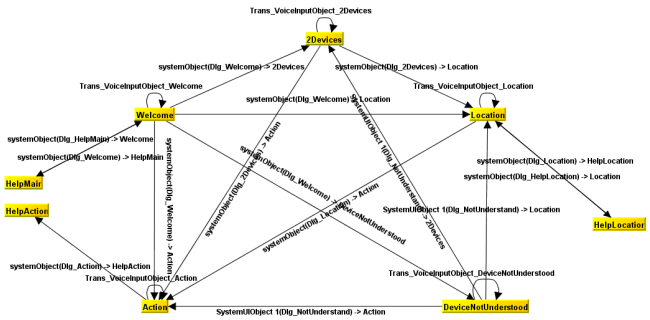



Figure 32: The state chart of the modeled INSPIRE system for a lamp control.

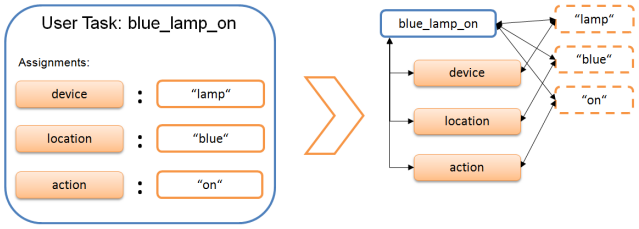



Figure 33: Assignments for the User Task *"blue_lamp_on"* and the related representation in the Knowledge Base of the UIM

model. Figure 34, for example, shows an interaction sequence with two additional used UI States. Obviously, the system did not understood the values for location and action in the first dialog and asked the UIM directly for completing the command.

It is rarely possible that the system misunderstood all three transfered Information Values of the UIM as shown in figure 35(a). Generally, prompts in SDS can be closed or open. Closed prompts are direct requests for concrete values, e.g. "Which device do you want to use?" whereas open prompts do not contain concrete requests such as "Hello, what may I do for you". If the UIM is confronted with open prompts, it simply transfers all Information Values of its Task Knowledge. As a consequence, the UIM believes that all values are committed to the system. Values that were misunderstood have to be requested directly. The UIM aborted the task in figure 35(b), for example, because the system was giving the same open prompt two times. The reason is a missing transition for the case, that device and location were transfered correctly and the value for action gone lost. As a consequence the system just repeated the prompt of "Hello, what may I do for you?" (compare figure 31 on page 66).

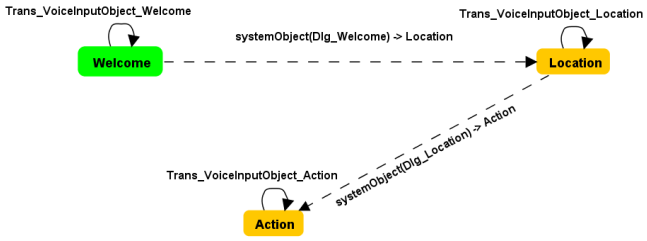
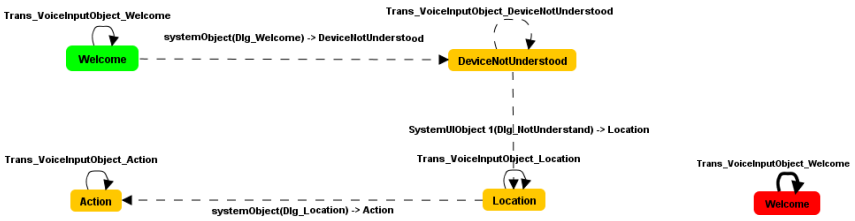



Figure 34: Interaction sequence with misunderstanding by the modeled INSPIRE system in 2 cases




(a) (b)

Figure 35: (a) Interaction sequence with misunderstanding by the system in 3 cases (b) a task abortion of the UIM

# 10 Discussion

Although cognitive science is a very young research field, the understanding of the human brain functionality is a covetousness of scientists since ages. The brain communicates with itself through complex, highly parallel communication processes involving the firing patterns of neural impulses, biochemical stimulation that bathes the brain structures, and methods as yet unknown. Even how information is stored, regenerated, and interpreted within brain circuits remains a major mystery, and will unlikely be solved soon ([39]). The fundamentals of life such as our biology, the environment, social settings or physical laws did not changed in the past ages, but all men of genius who tried to copy human anatomy or behavior failed. Even nowadays scientists with all of the technical equipment failed.

It is frustrating, that all existing approaches could not be used directly for the development of the cognitive processes for an AUE support, which are derived and explained in this thesis. The complete knowledge management of the UIM was developed with new approaches because none of the analyzed technologies was sufficient enough. Neuronal approaches, for example, are derived directly from brain anatomy and are unbeatable in pattern recognition, but their way of information storage is too fuzzy for systems such as software agents, which are communicating knowledge and therefore need ontologies with well defined semantic relation between single information. Both systems have abilities for learning. Neural networks generating new information itself whereas common software agents are learning during communication. The reason is founded by the level of abstraction as it was explained in section 6. Patterns are a kind of sensory data whereas the ontologies of software agents contain abstract information of the world. Automated usability evaluation and the need of human behavior simulations lead to an approach which combines both technologies.

For a fast development, sensory systems of the UIM for the analysis of the UI Description were disregarded. As a consequence, the avoidance was compensated by the Information Exchange Concept which is derived from the theory of speech acts. Section 5 explained how these principles could be used for a description of multimodal interactions with computer interfaces. The IEC made an analysis of all interface objects and interactions of a single dialog by the UIM possible. On the other hand, the complete information flow has to be annotated to the related

input and output interactions by the system engineer.
Nevertheless, the intricate and time-consuming annotation is justifiable in the context of usability evaluation. Most usability problems that were found by MeMo are caused by insufficient confirmations and, therefore, are based on an insufficient information flow. The discovered usability problem of the "4 Star Cooking Assistant", for example, might have fatal consequences and it is difficult to find for usability experts, contrary to the UIM. A simple confirmation of the selected recipe in one or two dialogs could avoid the problem. In general, usability problems are often based on mismatches between the system and the user's knowledge and most of them could be avoided by advancements of the information flow.
The IEC is also the basis for the knowledge management of the UIM. All communicated information could be stored into a memory system without any transformation. Additionally, these information also could be abstract such as text items. In this way, all of the communicated information could be brought into relation with the labels of navigation objects such as buttons or menu items. Thus, the UIM is able to learn how it has to use the system model as it was explained in section 6.
During this learning phase of the UIM, relations are generated without meanings and, therefore, have to be interpreted by the system engineer. Ontologies with well defined relations could solve this problem in future. Text items of these ontologies could be compared with text items in the UIM's memory system, and relations between matching text items in the memory system of the UIM could be added by the meaning of the relations from the ontology. In this way, new learned knowledge of the UIM could be communicated with other software agents. The approach might solve the Symbol-Grounding Problem which was explained in section 5, and over all, sensory systems could be simply added to the UIM. During the evaluation of the presented UI State, the UIM could store special visual sensor data into special lists. Relations can be added to each chunk that is stored, even abstract or discrete patterns (compare section 6.2.1), whereas special patterns might have associations to text items which can be communicated, contrary to the sensor data itself.
Another concept, that is not implemented yet, is the possibility of forgetting. Each chunk and relation in the Knowledge Base can be extended by forgetting scores, whereby the score of new added or used chunks and relations will be increased and those of unused decreased or deleted in case of very low scores. The alteration of these scores can also depend on emotions or the effort for a task completion, e.g. scores of chunks used during a difficult task will be more increased than during an easy task.
The current implementation of the UIM includes a Jess based Rule Engine which alters, among other things, the interaction selection. In this way, the engine

brought dependencies on sensory skills, weaknesses and attention of the simulated user, empirical data of common decision making, and usability expertise in general into the decision making process. Dividing the complexity of human decision making dependencies into rules has several advantages. Each rule alters the interaction selection in relation to special attributes of the UIM and/or the system model and can be overlooked by the system engineer. On the other hand, the usage of this approach for the alteration of the interaction selection has also disadvantages. The system engineer is told to annotate additional information such as the color, size or contrast of single interface objects. These are facts that will be used by the Rule Engine. Unfortunately, the decision, whether the contrast of an interface object is high, medium or low, or the annotation in general, is relatively subjective.
The same problem of subjectiveness is also relevant for the rule generation itself. The alteration of emotion for example is based on assumptions. People react in different ways and with different emotions to the same event. Anyway, as long as emotions are not clearly defined and categorized by psychologists, an approach of emotion alteration with the help of the Rule Engine seems to be the best way. Current theories are based on assumptions which might change in future. The correlated rules are based on these assumptions and could simply be edited, added or removed. Unfortunately, the current implementation uses one set of rules for all simulated users. A dynamic approach might be useful to adjust these rules and especially their influence to the behavior of the simulated user. The approach could also be used by psychologists to evaluate their theories about emotions and the influence to the action tendencies of the simulated user.
The current implementation of the UIM only includes the simulation of frustration. Dependencies which increase or decrease frustration can simply be found and they are even simply to implement, whereas frustration also has a big influence on the UIM's Intention. More complex emotions such as hope or disappointment require the implementation of a belief system. The Mental Model of the UIM is still in development, and once it is implemented, more complex emotions can be simulated as it was discussed in section 7.
The Mental Model also represents the knowledge of a user about the system it interacts with. The avoidance was compensated with the help of a Keyword Engine which was explained in section 8.1.2. Although this approach works well for the little amount of simulations that have been made with MeMo yet, the engine simulates always the same knowledge for all users. A calibration of that knowledge could be done by the Mental Model in future.
The requirements of the system model such as the information flow which has to be annotated, the decoding of the User Task into Assignments, the complex conditions and consequences of the used Transitions or the interpretation of the

simulation results impede a comfortable modeling of systems with MeMo. Furthermore, the behavior of the UIM was not evaluated with real user tests yet. Therefore, it is not for sure whether the cognitive processes lead to realistic or unrealistic user behavior.
But for all of that, the simulated cognitive processes support the semi-automated usability evaluation with MeMo in an astonishing good way.

# A Screenshots of MeMo

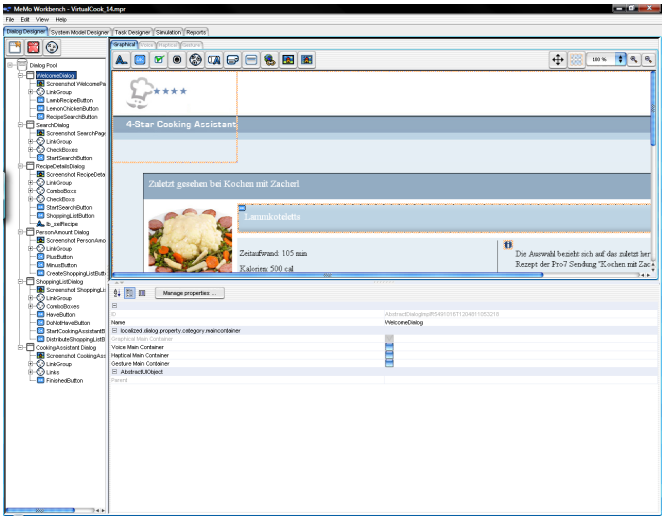



Figure 36: Modeling Phase - Dialog Designer

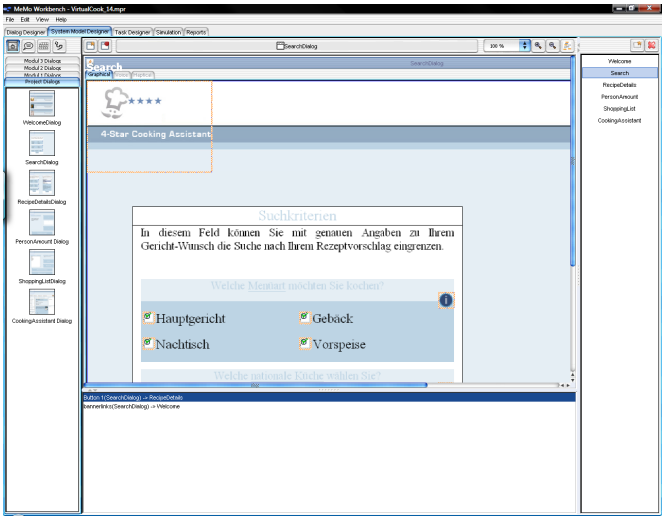



Figure 37: Modeling Phase - System Designer

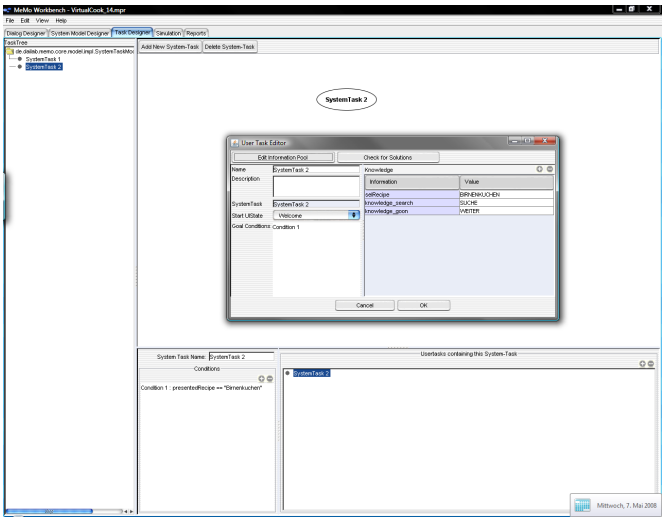



Figure 38: Modeling Phase - Task Designer

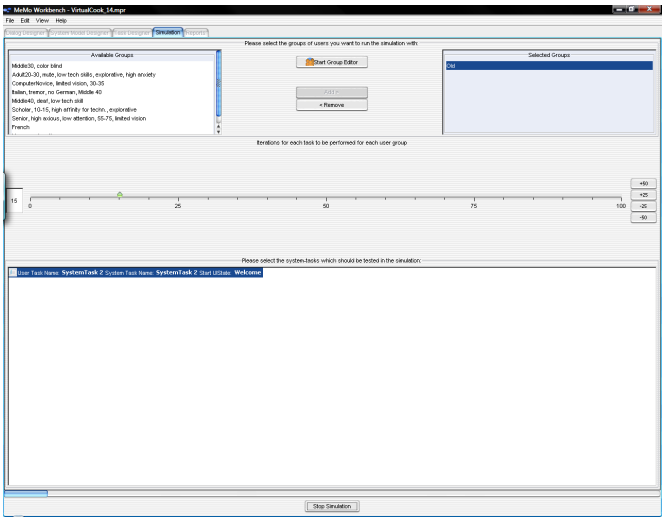



Figure 39: Simulation Phase - Simulation Settings and Simulation run

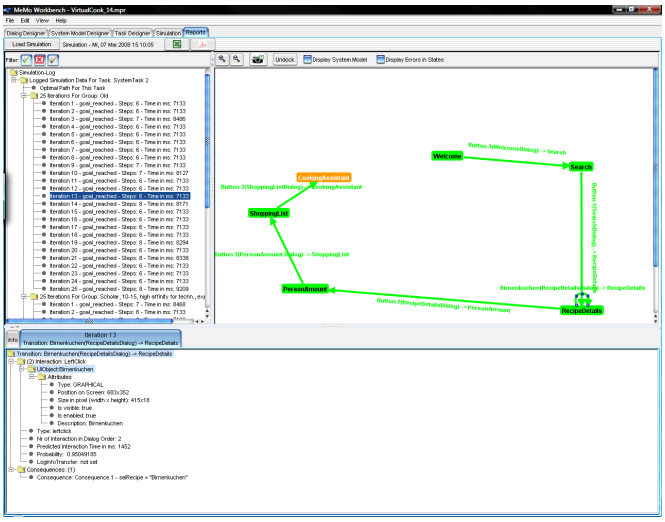



Figure 40: Reporting Phase

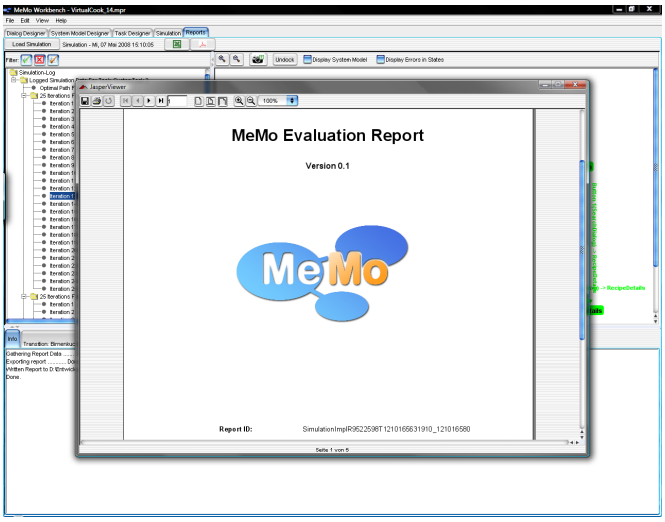



Figure 41: MeMo Usability Report - Part 1

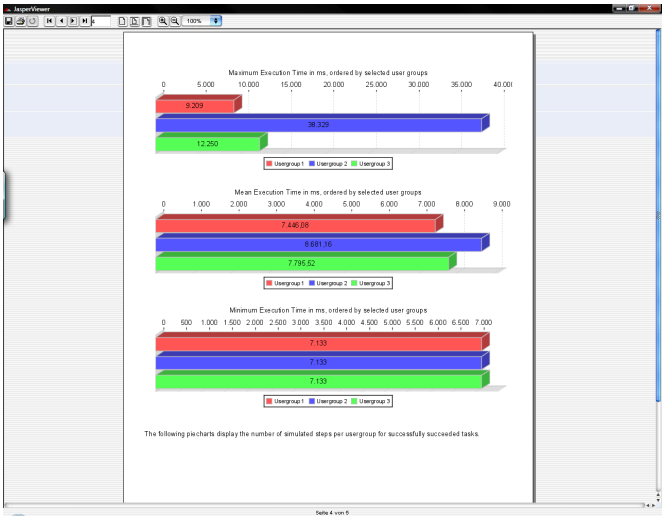



Figure 42: MeMo Usability Report - Part 2

# B Additional Material

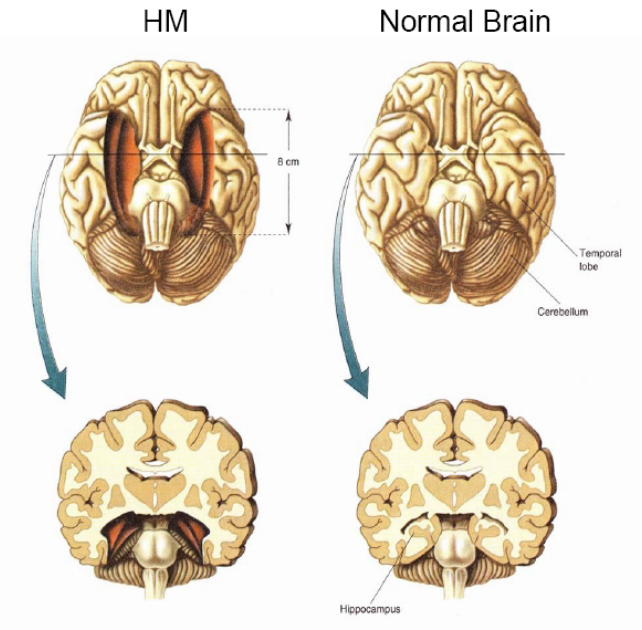



Figure 43: The brain of Henry. M after his treatment in relation to a normal one. During the treatment, his amygdala, his entorhinal and perirhinal cortices and about two-thirds of his hippocampi were removed.

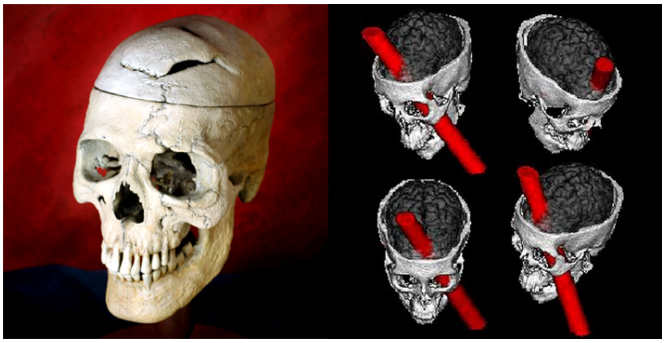


Figure 44: The skull of Phineas Gage and a computational reconstruction of the tamping iron that propelled through his left anterior frontal cortex.

# References

[1] http://www.krii.com/who_we_are.htm.

[2] The Knowledge Management Forum - "knowledge vs. information" http://www.km-forum.org/t000008.htm.

[3] http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf.

[4] http://herzberg.ca.sandia.gov/.

[5] Din en 9241.

[6] Din en iso 13407, 2000.

[7] Ghassan Al-Qaimari and Darren McRostie. Kaldi: a computer-aided usability engineering tool for supporting testing and analysis of human-computer interaction. Norwell, MA, USA, 1999.

[8] John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 1983.

[9] John R. Anderson. *Kognitive Psychologie*. Spekturm Akademischer Verlag, 2007.

[10] David L. Kleinman Yuri N. Levchuk Andras Pete, Krishna R. Pattipati. An overview of decision networks and organizations. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: APPLICATIONS AND REVIEWS*, 28:173–193, 1998.

[11] Roberto Pirrone Andrea Ruisi Antonia Chella, Massimo Cossentino. Modeling ontologies for robotic environments. *SEKE*, 15-17:77–80, 2002.

[12] H. Feldweg B. Hamp. Germanet - a lexical-semantic net for german. *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, 1997.

[13] Alan D. Baddeley. *Essentials of Human Memory*. Psychology Press, 2006.

[14] J.R.B. Cockett. Decision tree reduction. *Joarnal of the Association for Computing Machinery*, 37:815–842, 1990.

[15] Suzanne Corkin. What's new with the amnesic patient h.m.? *Nature Reviews*, 3:153–160, 2002.

[16] Antonio R. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain*. Harper Perennial, 1995.

[17] John R. Dance Dan R. Olsen, Jr. Macros by example in a graphical uims. *IEEE Computer Graphics & Applications*, 1988.

[18] Peter Dayan and L.F. Abbott. *Theoretical Neuroscience - Computational and Mathematikal Modeling of Neural Systems.* MIT Press, 2001.

[19] H. Jungermann; H.-R. Pfister; K. Fischer. *Die Psychologie der Entscheidung - Eine Einführung.* Spektrum, 2005.

[20] Henning Brau Florian Sarodnick. *Methoden der Usability Evaluation.* Huber, 2006.

[21] Sigfired Giedion. *Die Herrschaft der Mechanisierung.* Europäische Verlagsanstalt, 2000.

[22] Carmen Santoro Giulio Mori, Fabio Paternò. Ctte: Support for developing and analyzing task models for interactive system design. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 28:797–813, 2002.

[23] Melody Y. Ivory; Marti A. Hearst. The state of the art in automated usability evaluation of interfaces. Technical Report Report NO. UCB/CSD-00-1105, Computer Science Division (EECS), University of California Berkeley, California 94720, June 2000.

[24] Clark L. Hull. *Principles of Behavior.* D. Appleton-Century Co., 1943.

[25] L. Maichner K. Böhm. Real-time generation of topic maps from speech streams. *Proceedings of TMRA*, 5, 2006.

[26] U. Quasthoff C. Wolff K. Böhm, G. Heyer. Topic map generation using text mining. *UCS-Jornal of Universal Computer Science*, 8:623–633, 2002.

[27] Tom Johnstone Klaus R. Scherer, Angelika Schorr. *Appraisal Processes in Emotion: Theory, Methods, Research.* Oxford University Press, 2001.

[28] C. Kunze. Semantics of verbs within germanet and eurowordnet. *Proceedings of the ESSLLI-99 Workshop on Lexical Semantics and Linking in Constraint-Based Theories*, pages 189–200, 1999.

[29] H.C. Longuet-Higgins. Comments on the lighthill report. *Artificial Intelligence*, 1973.

[30] H. Liu M. Dash. Feature selection for classification. *Intelligent Data Analysis*, 3:131–156, 1997.

[31] John R. Anderson M. Matessa. Model of adaptive communication. 5000 Forbes Ave Pittsburgh, PA, USA, 2000. Carnegie Mellon University.

[32] Malcolm Macmillan. *An Odd Kind of Fame: Stories of Phineas Gage.* The MIT Press, 2000.

[33] Mario Verdichio Marco Colombetti. An analysis of agent speech acts as institutional actions. *AAMAS*, pages 1157–1164, 2002.

[34] F. Menzer. Mapping the semantics of web text and links. *IEEE Internet Computing*, 9:27–36, 2005.

[35] Ralph Rengger Miles Macleod. The development of drum: A software tool for video-assisted usability evaluation, 1993.

[36] Hans Moravec. *MIND Children - The Future of Robot an dHuman Intelligence*. Harvard University Press, 1988.

[37] Allen Newell. *Unified Theories of Cognition*. Harvard Univ. Pr., 1990.

[38] John Anderson Niels Taatgen, Christian Lebiere. Modeling paradigms in act-r. *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, 2006.

[39] Donald A. Norman. Cyborgs. *Communications of the ACM*, 3:36–37, 2000.

[40] Leo Obrst. Ontologies for semantically interoperable systems. *CIKM*, 3-8:366–369, 2003.

[41] Gautam Pant and Padmini Srinivasan. Link context in classifier-guided topical crawlers. *Transactions on Knowledge and Data Engineering*, 18(1), January 2006.

[42] Mala Mehrotra Pat Hayes, Thomas Reichherzer. Collaborative knowledge capture in ontologies. *K-Cap*, 2-5:99–106, 2005.

[43] Faith D'Aluisio Peter Menzel. *Robo Sapiens - Evolution of a New Species*. MIT Press, 200.

[44] Matthias Rauterberg. Amme : an automatic mental model evaluation to analyse user behaviour traced in a finite, discrete state space. *Ergonomics ISSN 0014-0139 CODEN ERGOAX*, 36:1369–1380, 1993.

[45] Avi Pfreffer Sanmay Das, Barbara Grosz. Learning and decision-making for intention reconcilation. *AAMAS*, 13-19:1121–1128, 2002.

[46] Vicki L. Sauter. Intuitive decision making. *Communications of the ACM*, 42:109–115, 1999.

[47] Rolf Pfeiffer; Christian Scheier. *Understanding Intelligence*. MIT Press, 1999.

[48] John Searle. *Speech Acts*. Cambridge University Press, 1969.

[49] Sigvard Strandh. *Die Maschine. Geschichte, Elemente, Funktion.* Herder Verlag GmbH, 1983.

[50] Don McKay Robin McEntire Tim Finin, Richard Fritzson. Kqml as an agent communication language. *CIKM*, 11:456–463, 1994.

[51] Joseph Bates W. Scott Reilly. Building emotional agents. 1992.

[52] John B. Watson. Psychology as the behaviorist views it. *Psychological Review*, 20:158–177, 1913.

[53] T. Finin Y. Labrou. Yahoo! as an ontology - using yahoo! categories to describe documents. *Proceedings of the eighth international conference on information and knowledge management*, pages 180–187, 1999.

[54] Zhiqiang Gao Yuzhong Qu. Interpreting distributed ontologies. *WWW*, 17-22:270–271, 2004.