Chapter 2
State of the Art

The main task of this chapter is the presentation of the state-of-the-art for the research which will be presented in the Chaps. 4, 5, and 6. First, the term usability will be defined at the beginning of Sect. 2.1. Then, the same section presents the fundamentals of traditional usability evaluation methods and of methods for automatic usability evaluation. This will show the high efforts of the traditional evaluation methods and the advantages of the automatic usability evaluation methods. The subsequent Sect. 2.2 will introduce exiting tools for the automatic usability evaluation of interactive systems, and explains the decision for one of these tools as research tool in the frame of this book. After the introduction of related work from the human-computer interaction research community, Sect. 2.3 will introduce approaches of socio-technical research which aim at describing human-technology interaction by the acquisition of usage patterns. Thus, the three Sects. 2.1, 2.2, and 2.3 will present the state-of-the-art which is related to Chap. 5.

Section 2.4 will describe existing approaches for describing interactions between human users and interactive dialogue systems. This includes description languages which support the modelling and development of multimodal systems as well as rather parameter-based approaches which aim at summative evaluations of spoken or multimodal systems. Chapter 4 is based on these previous works and will introduce a framework that combines model-and parameter-based approaches.

Existing approaches for the evaluation of user behaviour simulations will be provided in Sect. 2.5. The advantages and disadvantages of different measures and methods are presented as well as assessment criteria for such evaluation methods. These findings are the base and motivation for the evaluation method that is later described in Chap. 6.

Finally, in Sect. 2.6 three research questions will be derived from the previously presented related work. This book is answering the three research questions and Chap. 7 provides the related conclusions.

S. Hillmann, *Simulation-Based Usability Evaluation of Spoken and Multimodal Dialogue Systems*, T-Labs Series in Telecommunication Services,
https://doi.org/10.1007/978-3-319-62518-8_2

2.1 Usability Evaluation

Before introducing methods for usability evaluation in general and automatic usability evaluation (AUE) in particular, the understanding of the term *usability* will be determined for the frame of this book. The used definition is given in the ISO 9241 (Part 11) [51] by the following 4 statements:

Usability The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.

Effectiveness The accuracy and completeness with which specified users can achieve specified goals in particular environments.

Efficiency The resources expended in relation to the accuracy and completeness of goals achieved.

User Satisfaction The comfort and acceptability of the work system to its users and other people affected by its use.

The terms of this definition are used in the general literature on human-computer interaction (e. g. by Dix et al. [53, 277]) as well as ITU-T recommendations for the quality evaluation of spoken dialogue systems [96]. The three categories (effectiveness, efficiency, and user satisfaction) of usability can be measured by certain usability measures and how they contribute [53, 240] to different usability objectives [52], e. g. suitability for the task or learnability. For example, in relation to spoken dialogue systems Möller [148, 57] names task success as a measure of effectiveness reported in the literature and "dialogue duration or the number of turns uttered by the system or the user" (loc. cit.) as metrics of efficiency. On the other hand, user satisfaction is a subjective metric and should be measured by asking the user after an interaction, e. g. in the case of spoken dialogue systems with a questionnaire according to the ITU-T recommendation P. 851 [96]. Beside the usage of questionnaires, Engelbrecht estimated user satisfaction automatically from recorded dialogues which were annotated with dialogue acts [58].

There are many more aspects influencing the usability of a system and its acceptance by the users, even if concentrating on the usability of spoken dialogue systems. For example, Möller's taxonomy of quality of service (QoS taxonomy) contains 17 categories for task-orientated human-machine interaction [148] and puts the focus on the user. There, usability is directly influenced by the three categories communication efficiency, task efficiency, and comfort. These three categories are again influenced by nine less general categories, and 29 different quality elements (e. g. room acoustics, dialogue strategy, or cognitive demand) are involved. Finally, all quality aspects in the taxonomy are influenced by seven user factors.

However, the methods for automatic usability evaluation, which are the central theme of this book, are appropriate to analyse a system's usability on technical properties of human-machine interaction. Classic examples of such properties are dialogue length, task success, concept error rate, words per user/system utterance, query density, and so on. Engelbrecht predicted for dialogues between humans and a

spoken dialogue system the users' overall judgement on a 5-point Likert scale ("bad", "poor", "fair", "good", "excellent") [58]. He showed that such high level subjective judgements can be estimated from sequences of dialogue acts.

Still, until now automatic usability evaluation methods are not appropriate for estimating judgements on low level subjective items of usability questionnaires. The semantic differentials "impractical—practical" and "complicated—simple" (parts of AttrakDiff2 [82, 250]) as well as the statements "The system reacted in the same way as humans do." or "You were able to control the dialogue in the desired way." (proposed items from Suppl. 24 to ITU-T P-series Rec. [96, 19]) are good illustrations of such low level subjective measures. However, the models and simulations which are used or addressed in this book, are able to describe or mimic the interaction behaviour. This happens on the basis of stochastic models or rules, but not by the simulation of individual cognitive processes.

Figure 2.1 shows an extended version of Nielsen's usability engineering life cycle [157, 2, Fig. 1], which was inspired by Gould and Lewis key principles for designing usability from 1983 [76]. Although being rooted in over thirty years old ideas, the development model of the usability engineering life cycle (UEL) is still valid when designing for usability [147, 58]. Compared to Nielsen or Möller, the UEL in Fig. 2.1 is extended by the step of the automatic usability evaluation. The lifecycle starts with the analyses of requirements of the new system or new version of a system. Beside an exhaustive analysis of the current user needs, this analysis also involves experiences form the past (feedback from field). The conception of the design and the prototyping (which can be anything in the range from mock-ups to full concrete implementations) is followed by alternating phases of evaluation and iterative design modifications. Finally, the system is in the field and can be used for collecting feedback. Nielsen proposed to first find usability problems with an expert evaluation and to evaluate the optimized prototype with empirical testing in later stages of the development, because an expert evaluation is usually conducted in less time than an empirical test and provides concrete hints how to change the tested interface.

However, also an expert analysis (e. g. a cognitive walkthrough) has considerable efforts as just described in Sect. 2.1.1. Automatic evaluation techniques promise to be more efficient than expert evaluations, if used frequently to evaluate small changes

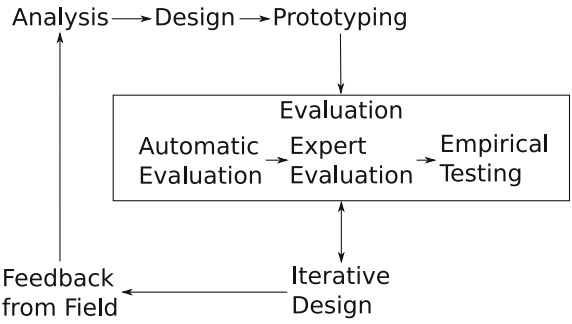

Fig. 2.1 The usability engineering life cycle with extensions regarding AUE methods

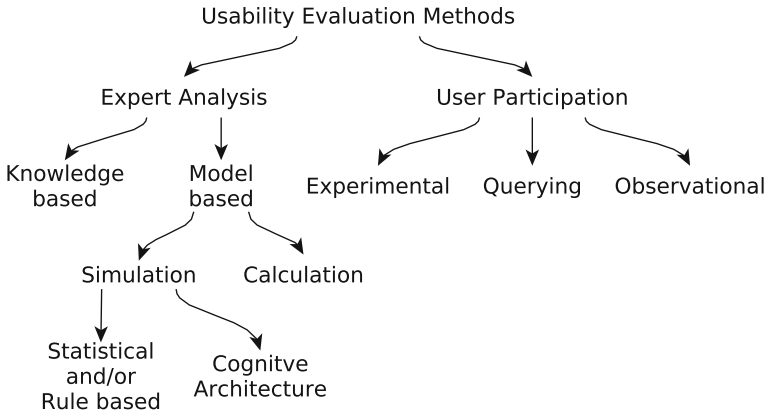

Fig. 2.2 Taxonomy of approaches for usability evaluation of interactive systems, summarising the methods described in [53, Chap. 9] and [115]

in the interface (e. g. as described in [180]). For that reason, automatic usability evaluation techniques should be used prior to expert analysis and methods with user participation in the UEL.

The following sections will give a brief introduction into human-based and automatic usability evaluation methods. Figure 2.2 shows a taxonomy of usability evaluation methods, which determines the structure of these sections. The differentiation of *human-based* methods into *expert analysis* and *user participation* follows the classification of Dix et al. [53, 319–364].

2.1.1 Methods Supporting Expert Analysis

A *usability expert* (expert as from now) is a person with strong experience in human-computer interaction, human factors or design. If such an expert analyses the usability of a system, it is called expert analysis. On the other hand, it is often useful to test a system with the participation of a so called *domain expert*, e. g. when evaluating a new information query system for medicines, pharmacists should be among the testing persons. Domain experts are important in order to test the *utility* of a system, i. e. to answer the question if the system provides "the right functionality?" [20, p. 350]. Bevan argues that usability depends on *ease of use* as well as utility (loc. cit.). A domain expert can be involved in both types of evaluations, either by participating in an expert analysis (e. g. a cognitive walkthrough) or as an acting participant of a user-based study. Dix et al. classifies analytic evaluation techniques by the 4 categories *cognitive walkthrough*, *heuristic evaluation*, *review based*, and *model based* [53, 360]. The former three are composed in the category *knowledge-based* in Fig. 2.2.

Following Dix et al. [53, 324–327], Nielsen and Molich are the developers of the heuristic evaluation method [159, 115–163]. In this method, the evaluators critique a system guided by 10 heuristics (Dix et al. argue that also other rules than Nielsen's 10 heuristics can be used). Nielsen indicates that 5 experts can find about 75% of a system's usability problems, if they critique the system independently from each other [160]. For that reasons, the heuristic evaluation is a flexible and "relatively cheap approach" and often considered to be "a discount usability method" [53, 234–325].

The *cognitive walkthrough* [53, 246] was described by Wharton et al. as a revision of [176] and "is a usability inspection method that focuses on evaluating a design for ease of learning, particularly by exploration" [246, 105]. It detects more concrete usability problems than the heuristic evaluation, but causes also higher efforts, in particular for its preparation (in comparison to Nielsen's heuristic evaluation). While the heuristics are fixed for different systems and tasks, the cognitive walkthrough needs an individually adapted procedure, which contains each time the following five steps:

1. Definition of user groups and materials: identification of aimed users, description of sample tasks as well as action sequences for accomplishing the tasks, and a description, mock-up or implementation of the interface to be tested.
2. Pooling of usability experts (analysts).
3. For each task, walking through the action sequences and telling a plausible story of the use case in order to answer the questions if the user will achieve the right effect, notices that the correct action is available, associates the correct action with the effect to be achieved, and sees the progress which is being made, if the correct action is performed.
4. Recording of critical information after each walk through an action sequence, which are: the required user knowledge, assumptions about the aimed user group(s), notes about side issues, and design changes as well as the plausible success story of the task completion.
5. Revision of the interface concept and/or its implementation.

A further expert-based usability evaluation method is the *expert review*. Here, an expert uses results from previous studies (e. g. from psychology or human-computer interaction) in order to support or challenge aspects of a design [53, 326] or an interaction concept. This review-based approach is usually more appropriate for the evaluation of more generic or abstract design decisions (loc. cit.). Examples are menu types, choice of icons, number of addressed concepts in a speech-based prompt, or appropriate cross cultural difference in pointing and semantic gestures (e. g. [145]).

Dix et al. name GOMS (which will be described in Sect. 2.1.4.1) and the cognitive complexity theory (CCT, published by Kieras and Polson [116]) as well as approaches for dialogue and system modelling [53, Chaps. 12 and 16] as examples of *model-based* evaluation methods. Furthermore, *simulation-based* methods are also used by experts for usability analysis of interactive dialogue systems. Both, model and simulation-based methods, will be described in Sect. 2.1.4 and are not further considered at this point.

2.1.2 Methods with User Participation

In a usability evaluation with user participation, the future users of a system under development, in terms of a group of persons representing the aimed user group, interact with a mock-up, a Wizard-of-Oz [112] (WoZ) system, or an actual implementation of a system. Mock-ups are usually used in early stages of the usability engineering life cycle to compare different concepts (e. g. sequences of dialogues or prompts) with each other.

A WoZ study is a design by simulation technique [148, 80] where the user shall presume to interact with a fully functional system, although parts of the system are not yet implemented. For example, to compare dialogue management strategies a trained person can substitute the automatic speech recognition and the natural language understanding of spoken dialogue system (e. g. in Möller's experiments with BoRIS [148]).

The chief difference regarding the style of a usability evaluation is the distinction into field study and laboratory study [53, 327–328]. The ITG-Guideline ITG 2.1-02 [95, 149] summarizes the differences between both styles and the properties of further evaluation methods, following the descriptions of Dix et al. (loc. cit.). The author of this book is a co-author of the ITG 2.1-02 guideline and the following descriptions base on these summaries.

In a *field study*, the test participants use the system to-be-evaluated in real usage scenarios, e. g. a speech controlled smart TV in their own flat. Field studies can answer questions about natural usage behaviour, long-term learning effects, multi-user interactions, but also about the acceptance of system features related to the user's privacy (e. g. an always open microphone in a smart home). However, in a field study it is difficult to control all influencing conditions, which leads to high efforts if hypotheses shall be tested in such a study. Depending on the system to-be-evaluated and the number of participants, the effort of the test-bed preparation can be much higher than compared to a laboratory study. The reason is the necessity to prepare and install the test environment for each participant individually.

In contrast, when conducting a *laboratory study*, the test-bed is installed in a laboratory. Here, laboratory means a location (usually a room) which is specially prepared, according to the requirements of the evaluation. This could be an acoustically isolated listening cabin but also a homelike arrange of a living room environment. However, in each case it is (much) easier to control the influencing factors in such a controlled scenario than in a field study. The acquisition of participants and the experiment are temporally detached in a laboratory study, which allows an efficient use of the experimental time. Nevertheless, there are also in laboratory studies high efforts in terms of human resources (i. e. personal for acquisition of participants and running each experiment) and money (i. e. expense allowance and personnel costs). For a complete overview about (dis)advantages of field and laboratory studies see [95, 25–26].

Especially in laboratory studies for the evaluation of interactive systems, the evaluation techniques think aloud, protocol analysis, experiment, and card sorting

are used [95, 30–33] and [53, 361]. A further evaluation technique is a focus group, which will be described below.

Card sorting [220] is simple to use, low priced, and still helpful to determine a logical structure of menus [95, 30]. In *closed card sorting*, the user creates a hierarchy of predefined items, while *open card sorting* is used to get categories of a given topic from the participant.

A *protocol analysis* [53, 344–46] consists of two stages. First, the behaviour of the user and the system is recorded. Among others, the used recording methods are paper and pencil (used by the analyst), audio recording, video recording, computer logging, and user notebooks. In the second stage, the recorded data are preprocessed and analysed. While computer logs and an experimenter's notes can be relatively easily prepared for further analysis, the transcription of audio and video recordings can be very time-consuming, e. g. Reidsma, Hofs and Jovanović report that the "manual transcription of speech usually takes 10xRT [10 times the real time duration]" [193].

Think aloud [53, 343–344] can be seen as a special case of protocol analysis [95, 30–31]. When using this method, the participant is asked to explain everything that happens during the interaction. This includes, to talk about the participants believe what is happening at the system's side, why an action is taken, or what the participant is trying to do. The participant is observed while he/she interacts and talks, using the methods of the protocol analysis.

The origins of the *focus group* technique lies in marketing, but the focus group can also be used to find usability problems or to decide between different design concepts [31]. Nielsen suggests having a focus group of "six to nine users to discuss issues and concerns about the features of a user interface" and that the discussion should typically last two hours [158, 94].

A further possible method for a laboratory study is the *experiment* (loc. cit.), where specific aspects of the user and/or system interaction behaviour are analysed under controlled circumstances. In a simple experimental design, one hypothesis about the influence of one factor is formulated and tested by two different experimental conditions. There, the conditions differ only in the one manipulated factor that has to be evaluated. Scientific experiments are necessary to test theories about the influence of certain factors (on user or system side) on human-machine interaction. However, the main problem is that users do not behave naturally under the restrictions of a controlled experiment.

All previously described user-based evaluation methods can be executed in usability evaluations. In order to actually get information about the system's usability, it is necessary to collect data about the interactions between user and system. Thus, the following section describes measures which are appropriate to collect such data.

2.1.3 Measures in Case of User Participation

When evaluating the usability of an interface or system in an empirical test, data can be collected by observing the interaction (e. g. by writing a protocol, recording audio and

video data, or generating log files by the system) or by asking the participants (e. g. using questionnaires, interviews, or usage diaries). The observing techniques and their problem to be partially very time-consuming in the post-processing were already addressed in Sect. 2.1.2, particularly in the description of the protocol analysis. The other measures are directed at the impressions and ratings of the user about the interface and the interaction [53, 348–351].

An interview allows to directly get information about the user's impressions in a structured way. On the one hand, it is a very flexible method, because the interviewer can ask questions on topics but the participant addresses the topics of the interview. On the other hand, an interview has to be planned in advance to be effective as well as efficient, and it usually needs much additional effort in the postprocessing. Furthermore, an interview generates a lot of qualitative data which makes it difficult to directly compare the results of several interviews with each other. Especially for scientific hypothesis testing, interviews are inappropriate.

As all questions are fixed, a questionnaire is less flexible than an interview. However, it is much less time-consuming and cheaper to collect data by a questionnaire, particularly if dozens, hundreds, or even more users have to be asked. When using questionnaires with scalar, ranked or multi-choice questions, the results can be compared by statistical methods.

Interviews and questionnaires can be used in field studies as well as laboratory studies. In contrast, a usage diary is most often used in the frame of field studies. Here, the user's experiences with the system or usage situations and procedures are noted by the user. This happens on a regular basis (e. g. daily) or in dependency to certain events (e. g. each time a service is used) [95, 34]. Furthermore, when filling out the diary, the participant is not observed and also not influenced by an experimenter (e. g. by social desirability bias [41]), which can generate more realistic statements.

2.1.4 Model-Based Usability Evaluation

The previous section gave an overview on usability evaluation methods which rely on the participation of users or analysis by usability experts. Using such a method means that users or usability experts directly interact with a design concept, an interface, or a system to-be-evaluated, which leads to high evaluation efforts. In contrast, in model-based evaluation (MBE) approaches, the analyst does not directly interact with the system to-be-evaluated, but uses a model of the user (and sometimes also of the system) to analyse possible interactions between a human user and a system.

As depicted in Fig. 2.2 on page 12, model-based evaluation methods are used by experts for usability evaluation. In addition to Dix et al., Kieras refines the distinction of MBE methods in his definition of model-based evaluation: "Model-based evaluation is using a model of how a human would use a proposed system to obtain predicted usability measures by calculation or simulation" [115, 1300]. Both, calculation-based and simulation-based approaches, rely on models of the users' interaction behaviour, but they differ in their need for a detailed task analysis.

The following two sections introduce both approaches and explain their distinction by the need of task analysis.

2.1.4.1 Calculation-Based

Card, Moran and Newell introduced the model-based evaluation approach into the field of human-machine interaction [115]. They describe in their book on "The Psychology of Human Computer Interaction" [33] the GOMS model [33, Chap. 5] as well as it usage for MBE. A GOMS model "consists of four components: (1) a set of Goals, (2) a set of Operators, (3) a set of Methods for achieving the goals and (4) a set of Selection rules for choosing among competing methods for goals" [33, 140, 144–147]. GOMS is an acronym, that is formed by the initial letters of the four components.

Goals can be hierarchically ordered and they either describe a rather abstract goal of the user (e. g. editing a manuscript in a text editor) or a rather concrete goal (e. g. deleting a certain character in the manuscript). Furthermore, a goal determines a set of methods which could be used to accomplish that goal.

Operators affect either the mental state of the user or the state of the task environment (e. g. a computer with is running software), and they represent elementary perceptual acts, motor acts, or cognitive acts. The types and effects of the operators, as well as their individual execution times, base on the *Model Human Processor* [33, Chap. 2] (MHP). The MHP works for auditory and visual inputs which are processed by three processors representing a human's perception, cognition and motor system. All three processors are connected by a shared working memory. The MHP simulates the flow of information from sensors (i. e. eyes and ears) through the processors and its effect to the motor system (e. g. hand movements). Each processor has an individual cycle time and depending on the current task of the user, the processors work either parallel or serial. However, the MHP can be used to predict the overall processing time, i. e. the time between input is perceived at the sensor and output is generated by the motor system. In other words, the sequence of operators which is used to accomplish a goal describes the behaviour of the user and allows calculating the time needed to accomplish a task.

A *method* describes a possible procedure to accomplish a goal. In the GOMS model, each method is represented by a conditional sequence of goals and operators. The test of that conditions bases on the user's memory as well as on the state of the task environment.

Finally, if more than on method is available, *selection rules* are used to decide which method should be executed to accomplish the current goal. Card, Moran and Newell state that "The essence of skilled behaviour is that these selections are not problematical, that they proceed smoothly and quickly, without the eruption of puzzlement and search that characterizes problem-solving behaviour." [33, 146] This addresses the limitation of GOMS to model the goal directed interaction behaviour for skilled expert users (without operating errors), but not for unskilled users (explorative with operating errors).

In the end, a GOMS model for a design concept or system to-be-evaluated can be used to compute the execution time of a given task (i. e. time to accomplish a goal). However, in order to build a GOMS model it is necessary to carry out a detailed analysis [53, Chap. 15][39, 221] of the task and the system. This analysis determines which methods and operators are supported by the system, and which of them a user would use to accomplish one or several goals. An exemplary case study of such an analysis is provided in [33, 313–331].

An important question for the task and system analysis is its level of abstraction. Card, Moran and Newell distinguish between the Unit-Task Level, the Functional Level, the Argument Level, and the Keystroke Level [33, 259]. For the latter, they developed the Keystroke-Level Model (KLM) [33, Chap. 8], which describes a user action as a sequence of six types of primitive operators. The KLM operator set consists of four physical-motor operators (keystroking, pointing, homing, and drawing), one mental operator, and one system response operator.

The execution times of single operators are determined by empirical studies and [33, Fig. 8.1, p. 264] gives an overview of the originally used execution times. The times for the physical operators, as well as the mental operator, are fixed, while the system response time depends on the system to-be-evaluated. Beside the definitions of the operators, the KLM relies on a set of 5 rules ([33, Fig. 8.2, p. 265]) describing the placing and deleting of the mental operator. This operator represents the time which is needed to mentally prepare the next physical-motor operations.

The KLM was developed as an engineering model of human-machine interaction and it is used until today, e. g. in the CogTool which will be described in Sect. 2.2.1 or to evaluate mobile interactions [88, 217]. However, the original execution times were changed and extended over the time, according to new research results, and requirements [88, 217].

Mainly, three different variants of the GOMS technique are considered in literature [105, 106], reflecting three different methods of task analysis. These are *CMN-GOMS* (Card Moran Newell GOMS, which is actually the original GOMS described above), *NGOMSL* (Natural GOMS Language) [113], and *CPM-GOMS* (**C**ognitive-**P**erceptual-**M**otor level of analysis and the **C**ritical-**P**ath **M**ethod) [104] [105, 337–344].

While the CMN-GOMS (using the KLM) solely predicts the execution time, NGOMSL can predict the execution and the learning time of tasks. Therefor, the procedures a user has to learn and execute in order to fulfil a task are structured in a program-like representation [113, 734]. For example, the analyst can define a procedure for the selection of a word in a text editor. Once learned, the user can reuse the procedure, independently from a higher-level goal, for e. g. either deleting or copying the selected word. Additionally, the concept of procedures allows the recoverability of already defined methods in the NGOMSL model, which eases the work of the analyst.

CPM-GOMS [77, 104, 105] is an extension of CMN-GOMS and is closely connected to the Model Human Processor, that was already described above. While CMN-GOMS and NGOMSL basically model a serial sequence of operators, the *critical path method* can be used to determine the overall execution time for tasks

with parallel activities which influence each other. CPM-GOMS uses the critical path method to analyse tasks where the user perceives auditory and visual information in parallel, while using (also in parallel) the motor system to generate an output which is related to the input. Because of their own durations and their dependency on other operators, the sequence of operators at the critical path determines the total execution time of a task in CPM-GOMS.

Independently from the used GOMS model or task analysis technique the usability expert has to build the model manually, meaning to decide which operators and methods are used as well as writing the whole model down. The later can be done by pen and paper, spreadsheet or dedicated tools. GLEAN4 [114] and GOMSED [199, 242] are tools which support building of NGOMSL models and which automatically calculate execution and learning time. Furthermore, SANLab-CM [170] (Stochastic Activity Network Laboratory for Cognitive Modelling) supports the creation and analysis of CPM-GOMS models. Execution times for individual operators can be defined as distributions in SANLab-CM, and the tool computes a related distribution of the execution time.

Finally, the program CogTool goes one step further. Here, the analyst builds a system model and demonstrates with this model how a user would interact with the system given a certain task. On the basis of this demonstration, CogTool automatically generates a KLM of the users physical-motor operations, which is translated into ACT-R production rules in a second step. CogTool will be further examined in Sect. 2.2.1, but simulation-based methods will be introduced previously.

2.1.4.2 Simulation-Based Methods

As depicted in Fig. 2.2 on page 12, simulation-based methods of usability evaluation can be divided into the two categories of cognitive architectures as well as statistical and/or rule-based approaches. The following gives an overview on existing approaches, without an exhaustive review of the individual properties and features. Concrete simulation-based tools for usability evaluation, which base on the simulation of user behaviour, will be presented in Sect. 2.2.

Cognitive Architectures "A cognitive architecture is a broad theory of human cognition based on a wide selection of human experimental data, and implemented as a running computer simulation program" [30, 94] is stated by Byrne. Additionally, he addresses in [92] the definition of a cognitive architecture: "A cognitive architecture embodies a scientific hypothesis about those aspects of human cognition that are relatively constant over time and relatively independent of task" [92, 312]. Both definitions reflect well the view on cognitive architectures in this book.

The Model Human Processor [33, Chap. 2] (MHP) was already introduced in the previous section and it represents the theoretical base of the different GOMS methods. It models human cognition as an information processing system that consists of three subsystems. This architecture can be used to compute the time needed to process an information and to react on it. Together with the Cognitive Complexity Theory [116]

(CCT) and Soar [124], the MHP belongs to relatively old (all are developed in the 1980s) and past cognitive architecture systems [30].

EPIC [146] (executive process interactive control) is a more recent cognitive architecture that uses ideas of the MHP (e. g. different processors) and the CCT (i. e. the production generation of the cognitive processor) [30]. All processors in EPIC work in parallel and on a more detailed level than the MHP. One of the most important differences is the possibility to produce words that are spoken as response to input by the system (loc. cit.).

Nowadays, ACT-R [7] (Adaptive Control of Thought–Rational) is probably the most frequently used cognitive architecture and ACT-R 7 [1, 25] is the current version of the software. Originally, ACT-R was developed for research on human cognition in the cognitive psychology. However, it matured over decades and now it can be used for other applications, e. g. to assess different computer interfaces with ACT-R based user models [25, 18]. At this point it must noted, that ACT-R is a framework which allows building user models that base on a cognitive architecture, but it requires a deep understanding of ACT-R and the underlying theory in order to create valid models and to assess the results.

Beside CogTool, Distract-R [203, 204] is another example of ATC-R's application to evaluation of human-machine interaction. Distract-R can measure the distraction of a car driver who operates a device (e. g. an air conditioning or a cell phone) while driving a car. In Distract-R, the ACT-R based user model controls a car in a driving simulator and operates the device to-be-evaluated in parallel. Salvucci compared 4 different interfaces (each voice or keyboard-based), regarding dialling time, and driver distraction, with Distract-R and in an empirical test. The results show, that the predictions of the ACT-R model regarding the difference between the conditions (user interface variants) are in line with the results of the empirical test [203].

Statistical and Rule-Based Simulations

In a statistical user behaviour simulation, the behaviour of the user model is learned from empirical data. Such data can be collected in an empirical test with the system to-be-evaluated or with other systems which uses the same modality and in the same domain as the evaluated system. In the field of usability evaluation of spoken dialogue systems, the statical model is usually an n-gram model which represents sequent system and user turns. Eckert, Levin and Pieraccini used a bi-gram model that based on user and system utterances [56]. Subsequent approaches use utterances which were augmented with additional information, e. g. the transferred information by Levin, Pieraccini and Eckert [131]. In the next generation of these models, the semantic of an utterance was additionally added to the model. Examples for that are SpeechEval [211] or the approaches of Lemon and Konstas [130], and Williams [247] as well.

In the case of rule-based user behaviour simulations, the behaviour of the user model in the interaction with a system is determined by rules. Such rules can be directly implemented as algorithms in the user model or separately defined and evaluated by a rule engine (i. e. an inference engine). Hard coded behaviour rules are used for relatively simple and rather specialised user simulations, e. g. to work as good as possible with a certain system or in a well-defined domain. Examples are

the CogTool Explorer [231], Williams's "handcrafted user behaviour model" [247], or the model used by Hillmann and Engelbrecht [86].

The usage of a rule engine leads to a more complex implementation of the user model, but gives the advantage of a broader (i. e. more general) applicability of the model. The reason is, that the behaviour of the model can be changed by modifying the underlying rule set, instead of changing and recompiling the source code. Furthermore, a rule engine can apply hundreds or even thousands *if-then-else*-like rules to a given context.

In the case of a user model interacting with a system, the context is the current state of the user model (e. g. task knowledge and the currently perceived user interface). The rules describe in which way the user model behaves given its current state. The application of rules to the context (through the rule engine) can decide either directly about the next action of the user model, or indirectly by changing in the model such parameters which control the algorithmic decision making of the user behaviour simulation. For example, the approach of indirect modification of model parameters is used in the MeMo Workbench.

While this section gives solely an overview on techniques for user behaviour simulation, concrete applications (e. g. like the named examples SpeechEval, CogTool-Explorer, and the MeMo Workbench) will be shortly described in Sect. 2.2. Furthermore, the functions and concepts of the MeMo Workbench will be explained in Chap. 3 in detail.

Statistical User Simulation for Training of Dialogue Managers

Particularly in the field of spoken dialogue systems, user behaviour simulations are used for the training of statistical dialogue managers. A statistical dialogue manager uses a policy to decide in each dialogue state which system action (move to which system state) is most appropriate to the user's goal (e. g. predicted from his last utterance). Such a policy has to be learned by the dialogue manager, which needs numerous training interactions. The large number of training trials (e. g. several thousands) can not be achieved with human users interacting with the dialogue manager. For that reason, a user behaviour simulation can be used for the generation of example dialogues. Here, the UBS interacts (usually at semantic level) with the dialogue manager to be trained, in order to confront the dialogue manager with many various dialogue paths. However, randomly or also systematically but still arbitrary generated responses of the user model are not appropriate to train a reasonable policy.

For example, the hidden agenda user model was used to train a statistical dialogue manager which based on a hidden Markov model [210] or partially observable Markov decision processes [208]. Furthermore, Cuayáhuitl used an ontology-based user model [42, Chap. 4] to train by a dialogue manager that based upon Semi-Markov Decision Processes [42, 91–105]. The ontology represented the user knowledge about the ongoing dialogue and enabled the user model to generate "coherent user responses, i. e. responses that make sense to humans" [42]. A last example for statistical user simulation is Pietquin's approach to simulate user behaviour with a Bayesian network [174, 84–93]. Here, the user simulation was used to train a dialogue manager by unsupervised reinforcement learning.

2.2 Tools for Automatic Usability Evaluation

2.2.1 CogTool

With CogTool [17, 103], an evaluator can model systems which get input by keyboard, mouse, touch-screen, or microphone. Display and loudspeaker can be used as output devices by the modelled system. It is possible to use any combination of input and output in CogTool's system model.

The system to-be-evaluated is modelled by a finite-state machine and each state represents one dialogue of the system parts to be evaluated. Here, a dialogue means a specific occurrence of the user interface, e. g. a window with labels and buttons in a GUI. Each state (i. e. dialogue) contains all interaction elements that can be used by a user, e. g. buttons, check boxes, or text fields. Usually, interaction elements are annotated in screenshots, which determine their types and relative sizes as well as their relative position to each other. The single dialogues are connected by transitions of the underlying finite-state machine. Here, each transition reflects a system action that is trigged by a user action on a certain interaction element.

The evaluation approach of CogTool is partially based upon ideas of GOMS, which was introduced in Sect. 2.1.4.1, and is appropriate to evaluate interfaces which are used by expert-users (i. e. users who interact goal-oriented). For that reason, there is no special task model or model of user behaviour, but the evaluator demonstrates with the system model all steps that are needed to fulfil a task [107]. CogTool's user behaviour simulation describes the execution of user actions and is implemented as a Keystroke-Level Model (KLM), which was also described above. In preparation of the actual simulation, the demonstrated interactions steps are divided into basal KLM operators and then translated into the according ACT-Simple [205] commands. Finally, ACT-R productions are generated by the ACT-Simple compiler out of the ACT-Simple commands. These productions are evaluated in ACT-R in order to compute the duration of single steps (at KLM level) as well as the total task duration. Beside the task duration, CogTool provides a visualisation of all used KLM operators over the time and the according ACT-R script.

2.2.2 CogTool Explorer

CogTool-Explorer (CT-E) is an extension of CogTool (see above), which allows simulating a user's goal-directed search for a hyperlink on a hierarchically nested website [231]. A text (e. g. a task description) describes the user goal. During the simulation, the text is semantically compared with all hyperlinks which are perceived by a cognitive SNIF-ACT (Scent-based Navigation and Information Foraging in the ACT architecture) user model. SNIF-ACT describes the navigation behaviour of users on websites, by modelling the users' perception and rating (according to

the goal) of hyperlinks. However, modelling the searching and rating behaviour is not sufficient in order to estimate the task completion duration. For that reason, additional operators for visual search and hand movements are added for the use in CT-E.

In contrast to CogTool, the behaviour of CT-E's user behaviour simulation is non-deterministic and can find a valid click path to the given goal without the need of a demonstration by the evaluator. Thus, CT-E simulates different (but also possible) variants to reach the user goal. For each variant, the task completion time is computed as in CogTool. Actually, CT-E automatically generates different demonstrations, which are analysed by CogTool as described above.

2.2.3 BisWas

Biswas and Robinson describe and validate a simulation-based tool for the evaluation of GUI-based systems, according to their usage by users with physical impairments, i. e. vision- and motor-impairments [22, 23]. Their approach simulates cursor-based (e. g. a mouse) interactions between a system and an impaired user. Similar to CogTool (see above), the system is modelled as a finite-state machine. Here, a state reflects a screen (i. e. an image of the actual graphical dialogue) and a certain user action is represented by a transition. However, the system-model does not use further interaction elements (e. g. buttons), but actions are only bound to appropriate areas of the screen. The simulation of an interaction results in data about the interaction duration, movements of the cursor and probable eye movements.

The user model is composed of three sub-models which reflect the perception, cognition and motor behaviour of a user. According to the kind and degree of impairments of a certain user group, the models are parametrised for the simulation. The perception model simulates the perception of interfaces elements and the search for the next possible element that could be used. Than, the cognitive model decides, if the element is used for the desired action, or if the search for interaction elements must be continued. There are two implementations of the cognitive model. One of them relies on the GOMS method and simulates decision making of experts, while the other can be fitted to empirical data in order to simulate the behaviour of non-experts. For the former, the evaluating person (e. g. a designer) demonstrates the click-path to-be-evaluated. For the later, it is necessary to collect data about the interaction behaviour by an empirical user study. However, such data can also be used for systems which deviate from the empirically tested version. Finally, the motor behaviour model simulates the movements of the cursor in dependence of the supposed motor impairments. This model is based upon a study of Trewin and Pain about "keyboard and mouse errors due to motor disabilities" with 20 motor-impaired users [235].

2.2.4 SpeechEval

SpeechEval can be used for the usability evaluation of spoken dialogues systems [211]. In SpeechEval, the user model interacts over telephone directly with the system to-be-evaluated. That is realized by a user model that uses an automatic speech recognizer to receive system utterances and a text-to-speech system to utter textual responses. The main purpose of SpeechEval is the generation of natural dialogues without the need of user tests. The generated dialogues can be analysed afterwards, e. g. in order to detect usability problems or to estimate user judgements [152].

SpeechEval's stochastic model for utterance generation relies on the VoiceAward (a challenge of commercial spoken dialogue systems) dialogue corpus. This corpus contains 1900 dialogues with 130 spoken dialogue systems. A bigram model is used to determine a feasible user utterance as response to a received system utterance. Here, the actually uttered user utterance is selected according to its probability in the bigram model. As the communication between user and system is naturally disturbed, no additional error simulation is applied.

2.2.5 MeMo Workbench

The MeMo Workbench allows the simulation of interactions between users and GUI-systems as well as spoken dialogue systems [59]. An extension which was implemented in the frame of the work, which is described in this book, enables the simulation of sequential multimodal interactions with systems supporting GUI as well as spoken speech input or output. In the MeMo Workbench, an application (or its concept) to-be-evaluated is reproduced as a system model. This system model is based upon a finite-state machine describing the relations between the current occurrence of the user interface and a set of typed variables describing the emulated system state. This allows the emulation of dialogue sequences and internal system states (i. e. the system logic). This approach seems similar to the modelling of dialogue sequences in CogTool, but only at the first glance. In contrast to CogTool, it is possible to define transition conditions and transition actions. Furthermore, the state machine used in the MeMo Workbench is implemented as a transducer, which means it can generate output according to given input.

The modelling of graphical user interfaces is based upon the annotation of screenshots or mock-ups. Spoken dialogue systems are modelled by interaction elements representing prompts (system output) and slots (system input). While CogTool and the Biswas Simulator can model only one transition per interaction element (e. g. a button), the system model of the MeMo Workbench allows several (in principle any number) of transitions per interaction element which change form one state to another.

The MeMo user model is part of the MeMo Workbench and simulates the behaviour of users who interact with the system to-be-evaluated. The information exchange

between the rule-based, probabilistic user model and the modelled system occurs at text- and concept-level (for the spoken speech modality only at concept-level). The perception, processing, and execution of interaction elements, which are provided by the system model, draws on the Model Human Processor's division in to three subsystems (perceptual system, cognitive system and motor system). However, it is important to note that the internal implementation of the three modules in the MeMo user model is completely different from the concept of processors and memories which is used in the subsystems of the Model Human Processor. All interaction elements (e. g. buttons, drop-down menus or speech-based prompts) are weighted with usage probabilities. These probabilities are influenced by properties of the dialogue interface (e. g. font size and content of a label, which reflect readability and suitability for the task, respectively) as well as user properties (e. g. age, eye-sight, or technical skills). The influence of system and user properties is determined by rules [212]. Because of the probabilistic component of the MeMo user model, several interaction simulations of the same task can produce different—successful and unsuccessful—interaction paths.

The result of a simulation (of several interactions for the same task) in the MeMo Workbench is provided by accumulated as well as an individual visualisations of the simulated interaction paths. The visualisation provides a graph reflecting all visited dialogue states (vertices) and the relative amount of transitions from one state to another (edges and their width). Deviations from the shortest possible solution path, for the given task, are colour-coded. Furthermore, the rules which caused a deviation are provided and they can help to determine possible usability problems [58, 41–60]. Beside the graphical analysis of the simulation results, the MeMo Workbench provides reports which contain detailed data about the state of the user model and the system model for each interaction step. For simulations with system models of pure spoken dialogue systems, all parameter which are needed for a prediction of user satisfaction with the PARADISE framework [240] are provided.

The preceding paragraphs gave an overview about the main principles of the MeMo Workbench and the related MeMo user model. A more detailed description of the user model, the system model, and the evaluation process with the MeMo Workbench will be provided in Chap. 3.

2.2.6 Selection of the Used Research System

The preceding subsections have introduced five tools for automatic usability evaluation. Three of them, i. e. CogTool, SpeechEval, and the MeMo Workbench, are able to model or simulate spoken speech based interactions. Furthermore, only the user models of SpeechEval and the MeMo Workbench can automatically interact with a system or system model on the basis of a predefined task. SpeechEval communicates with the system-to-evaluated at a signal level, while the MeMo Workbench simulates interactions at a concept level, using a model of the system to-be-evaluated. The latter approach is easier to implement in a testbed and appropriate for the evaluation

of interaction concepts in early stages of the usability engineering process. For that reasons, the MeMo Workbench and the MeMo user model were chosen as a platform for the implementation and evaluation of a new simulation approach in the frame of this book.

2.3 Describing Interactions

In sociological technology studies two different essential approaches exist regrading empirical and conceptional research on human-technology interaction. On the one hand, the empirical gathering of typical patterns—or practices—in those interactions, and on the other hand typing of usage behaviour (also on the basis of empirical observations). Here, typing happens either by the identification of group specific behaviour (e. g. of elderly people) or by the classification of certain usage situations. The following two subsections will introduce both approaches, because a mixture of both approaches will be used to build a user behaviour simulation in Chap. 5.

2.3.1 Methods for the Reconstruction of Practices

2.3.1.1 Technographical Observation

Successful usage of technology (e. g. computers, washing machines or even light switches) is usually performed by incorporated practices (i. e. typical usage patterns) or routines. The user transfers such practices form former usage contexts to new technologies. In order to understand how users acquire the usage of new technologies, it is necessary to reconstruct their practices. Typically, that happens in a methodically controlled and broad empirical observation of human-technology interaction. This method links to the field of research of ethnography (see [4, 10, 87]), and includes participating in long-term observations in the field, direct journalising of interaction events in observation diaries (including prosodic properties), and an extensive qualitative analysis of the collected data (*holistic description*).

Rammert and Schubert define the research field of technographics which is linked to ethnographics, but also extends the later [183, 186, 189]. The method of technographics tries to enable new methods of theory-driven analysis, e. g. by looking for the socio-technical order while gathering and interpreting the medial and structuring force of technological artefacts [185, 19]. Technographics relies on former work of Suchman, Woolgar, Latour, and Hutchins [93, 126, 227, 250]. These works consider the individual course of human-technology interactions as well as identify general interactivity problems. Furthermore, their findings were used to deviate recommendations for the development of new technologies.

Another approach, which is also included in technographics, is called *focused ethnographics* (*fokussierte Ethnografie* in German) [118]. Here, long-term field

observations and holistic descriptions are substituted by short (focused) field visits with the aim to reconstruct practices in the selected sections of social life. Finally, technographical methods are open for the usage of technical recordings (e. g. audio and video recordings) as an additional data source for the behaviour analysis [118, 215]. This is not a new method in human-computer interaction research (in the sense of the description in Sect. 2.1), but a discrepancy to the classical ethnographic research which is defined by the abstinence of extracorporeal research instruments [87, 19]. This discrepancy represents a methodical benefit, because it allows the review of subjective interpretations of observed interaction behaviour [118, 25]. In line with the usage of such recordings are recordings and analyses of log files (written by the technical partner of the interaction, i. e. the system) [80]. All these methods facilitate the observation of interactions between a human user and the user interface of the machine or system to-be-operated. Again, such methods are standard in the human-computer interaction community since decades, but their introduction into sociological field studies marks an paradigm shift.

2.3.1.2 Simulations

Simulations are used by different research fields in sociology and they can be divided into three approaches. Firstly, the social simulation which supports decision making process when acting out scenarios or extrapolating trends in complex social systems. Secondly, the theory simulation which tries to formalise and simulate theoretical characters (e. g. the prisoner's dilemma) or fundamental problems in the theory of sociology (e. g. appearance of emergent phenomena or structures). This approach tries to clarify formalised problems by a multitude of simulation trials (cf. [63]). The third approach interconnects simulation and social research. For example, role-playing games are often named [15] in the research field of participatory simulations [79, 182].

2.3.2 Classification of User Behaviour

In the reconstructive and the prospective socioscientific research on usage of technologies, three different research approaches can be distinguished. Firstly, the acceptance research which is based on the properties of technology and asks for individual or situational factors that are responsible for the usage of technology. Secondly, approaches which are demographically motivated and which emphasize pure social factors, like age, education, gender, or income. Thirdly, approaches of the new technical sociology which use typical interaction patterns (i. e. practices) in human-technology interaction of different user groups.

The acceptance research asks for factors explaining the usage or non-usage of current or future technologies. This approach distinguishes input-output-models as well as feedback models. In the former, the examination of the individual user skills

[46, 75, 75], the perceived utility of devices, or the perceived usability of devices are paramount [5, 6, 43, 45, 46, 239].

Feedback models (cf. [62, 119]) rely on the hypothesis that a sustainable usage of technology only arises over the time and has to be proved again and again. For that reason, such approaches focus on the user's experience as well as factors influencing this experience [3, 119, 192, 198].

Both kinds of studies can only be applied to existing products (cf. [119]) or good imitations of those, e. g. in a Wizard-of-Oz test (see Sect. 2.1.2). Furthermore, the explanatory power of such models is limited, as only half of the actually observed usage behaviour can be explained by the supposed variables [46, 239]. Finally, the differences between the users are seldom considered and also empirical founded classifications of user are rare.

In contrast, for a great number of demographical studies a typed usage of technology (i. e. the usage is determined by social characteristics) was shown. However, there is no common standard of knowledge which are the crucial social factors; yet, generation affiliation, eduction level, and income level are often named.

Wopfner argues that the affiliation to a generation (which depends on the user's age) implies an absence of media competence, a missing natural learning environment and a missing self-image of the acquisition of knowledge [251]. In addition, Sackmann and Weymann have shaped the term of *technology generations* (*Technikgenerationen* in German). It means, that the handling of technology is shaped by typical experiences of a certain generation with typical technologies in the respective primary stage of socialisation [201] (see also [236]).

Rogers derives from the factors income level and education level the adopter groups of the *innovators*, the *early adaptors*, the *early majority*, the *late majority*, and the *laggards*[198, p. 263 ff.]. Here, income and education level decrease from group to group in the order of their naming. For example, a laggard belongs to a group of users which have the strongest anchoring in their practices and traditions. This leads to an a priori critical attitude towards new technologies of the laggards.

Recent socio-technical research ties at demographic approaches as well as at social constructivist explanations of usage behaviour. Here, the individual usage and acquisition behaviour is explained by social rationalities, schemas, and practices of the users, and their fit with the respective innovation of technology (cf. [3, 132, 187, 198]). Furthermore, the recent and more symmetrically aligned socio-technical research emphasises that this acquisition process is not one-sided (cf. [184, 185, 190]). In fact, Licke and Rammert emphasize that users as well as technology have to come closer to one another [132, 187] in order to enable an alternating adoption to each other [132, 126].

The just described approach is relatively new in socio-technical research. For that reason, there is only little research regarding the question of how technology usage is embedded in well-proven circumstances (cf. [198]). The generalisation of research results from empirical studies to a model of the influence of historically learned and experienced practices is still mainly an open research question.

However, such a generalisation could rely on several accepted approaches of sociology. Regarding the German-speaking sociology, practice-theoretical approaches of

Reckwitz [191] and Höerning [90, 91] are relevant for this topic. Both refer to Bourdieu who emphasises that the usage of each and every technical artefact is determined by practices which were learned by former handling with (other) objects [26].

A certain kind of operationalisation of the relation between practices and technology is given by the concept of lifestyle-types (see [90]). This concept was identified by Hörning in his analysis of the practical usage of computers (loc. cit.) and it phrases that a user's lifestyle influences the individual practices of technology usage.

2.4 Description and Logging of Multimodal Interactions

This section gives an overview of existing approaches for describing interactions between human users and interactive dialogue systems. The review considers related work from the two research fields in human-computer interaction, which are tightly connected in the usability engineering life cycle which was introduced in Sect. 2.1. The fields are the design and development of multimodal system, and the evaluation of such systems as well.

The following subsection provides related work on the *design and development* of interactive multimodal systems. In contrast, the second subsection addresses the *evaluation* of such systems. Finally, the third subsection will introduce an approach for classifying and comparing description languages and structured models of multimodal human-computer interaction. This approach will be used later in this book to compare a logging framework, which will be described in Chap. 4, with approaches selected form those introduced below.

2.4.1 Design and Development of Multimodal Systems

For many years standardisation bodies have focused their efforts on finding a common notation to describe different aspects of interaction in a multimodal system. This subsection describes some of the most representative approaches aimed at modelling or representing multimodal interactions in the frame of system development. The presented approaches were selected because they aim at describing interaction in a more generic way, either by finding equivalences between the different modalities or by combining data of different interaction modes into the same representation.

2.4.1.1 Markup Languages

The Multimodal Utterance Representation Markup Language (MURML) [120], the Multimodal Interaction Markup Language (MIML) [8], the USer Interface eXtensible Markup Language (USIXML) [134], as well as the Device-Independent MultiModal Markup Language (D3ML) [72] are examples for the usage of markup

languages for the description of interaction when implementing interactive dialogue systems. Other, more recent markup languages for interaction description are the eXtensible Markup language for MultiModal interaction with Virtual Reality worlds (XMMVR) [163] and the Extensible Multimodal Annotation markup language (EMMA) [108]. All six approaches will be briefly described in the following.

In MURML utterances are composed of speech output which is augmented with gestures. This aims at describing the equivalence between the two modalities.

In contrast, MIML aimed at describing interactions at a semantic level and for various platforms. Three layers are used in MIML to describe interactions on the basis of the task to be fulfilled, the modalities which are used for system input and output as well as the concretely used device (i. e. available modalities) in the target scenario.

The formal markup language UsiXML was developed for the description of user interfaces. It supports "Multi-path Development for User Interfaces" [133], which is a paradigm for the specification and production of user interfaces at several levels of abstraction. Thus, it allows the development of interfaces, which adapt to the context of use on the basis of a formal specification. UsiXML is composed of a set of integrated models which are structured according to Furthermore, UsiXML supports the transformation of the models which specify the user interface. This is ensured by the definition of systematic rules describing processes for abstraction, reification or translation at the interface design stage.

The Device-Independent MultiModal Markup Language (D3ML) is domain-specific (web-based user interfaces) but independent from the used input and output modalities. Thus, it allows the specification of systems which adapt dynamically to devices and modalities which are available in the actual usage situation. In order to enable the modality-independent interface specification, D3ML aggregates a remarkable amount of relevant meta-information about the course of interaction.

Virtual 3D worlds and especially voice-based interactions to manage objects in such worlds can be described with XMMVR. It is a hybrid markup language which embeds VoiceXML [164] to model speech-based dialogues and the Virtual Reality Modelling Language (VRML) [34] to model 3D scenes and actors in those scenes.

Finally, EMMA allows the annotation of system input (which is given by the user) for several modalities. EMMA contains various recommendations from the Multimodal Interaction Framework [125] of the W3C (World Wide Web Consortium). This annotating language can be used to describe user input and information uttered by several interaction modes. A certain input action and its semantic interpretation by the annotator are stored side by side in EMMA.

2.4.1.2 Models and Architectures

Beside markup languages, other approaches which suppose architectures or use models for describing speech-based and multimodal interaction exist. In the following, CAMELEON-RT, ICO, NiMMiT, MARIA, and an MARIA-based approach of a logical language by Manca and Paternó are regarded.

CAMELEON-RT [13] is a reference architecture that describes and classifies user interfaces which support multiple usage contexts by providing various interaction modalities. It uses three levels of abstraction in order to decouple the task to be carried out with a system from the actually used shape (e. g. speech-based interface, GUI with touch input, or GUI with mouse and keyboard) of the user interface in a certain usage situation. The three abstraction levels are the abstract user interface (AUI), the concrete user interface (CUI), and the final user interface (FUI). For an interface, the three levels are specified independently from each other, but with respect to the possible contexts in which the FUI will be used. Both, the design and the runtime phase, are covered by CAMELEON-RT. Thus, it can be used to structure the development lifecycle of a system's user interface in accordance with the usability engineering life cycle (which was described Sect. 2.1).

The ICO (Interactive Cooperative Objects) [167] notation provides a formalism to describe and analyse interactions in interactive systems, as well as to reason on the basis of the formalized interaction. In the ICO approach, system components and their structure are specified using object-oriented modelling methods, while Petri nets are used to describe the system's behaviour. Low-level events are used to model the user input with each modality. Several low-level events from different modalities are converted into high-level events. This conversion is used model modality fusion and can be formally analysed. Finally, ICO models are executable and can be used for the simulation of system behaviour.

The state-and event-driven notation of NiMMiT [237] (Notation for Modelling Multimodal Interaction Techniques) allows to graphically model uni- and multimodal interactions in three-dimensional environments. In NiMMiT the developer specifies interactions and their effects by drawing and editing state diagrams.

Another framework which allows the modelling of systems supporting multimodal interactions is the MARIA (Model-based lAnguage foR Interactive Applications) framework [169]. MARIA models interactions according to the CAMELEON-RT reference architecture, which was described above. The framework provides a model-based language which can be used to define domain-specific languages by refining a predefined abstract vocabulary. Thus, the designer can describe participants and events of the interaction according to the aimed target platform (hardware) and the planed modalities. Because of the meta-language approach, MARIA can be used independently from the planed modalities for interaction.

Manca and Paternó described a logical language [137] which based on MARIA and the CARE properties [40] (the next subsection gives additional information on the CARE properties). Form now on and in the rest of this book, this approach is referred by the acronym SOMBDE (**S**upporting multimodality in service- **o**riented **m**odel-**b**ased **d**evelopment **e**nvironments). This language supports the development of graphical-vocal user interfaces. Here, interactions are described at an abstract level and they are transformed into more specific interface components when applied to a certain modality. As the language bases on MARIA, the authoring environment of the MARIA framework can be used to model interactions using a graphical (diagram-based) tool.

2.4.2 Evaluation of Multimodal Interaction

This subsection describes evaluation approaches for spoken dialogue systems (SDS) and multimodal dialogue systems (MMDS). Such evaluation approaches can be classified into four different groups according to the nature of the metrics which are used for the evaluation process: (1) parameters quantifying the user-system interaction, (2) parameters describing task efficiency and task success, (3) parameters measuring modality efficiency, and (4) recorded user behaviour.

2.4.2.1 Parameters and Measures

In the course of time, some metrics and parameters to quantify and assess interactions in the area of SDS and MMDS were proposed by different evaluation approaches. Fraser proposed a common set of metrics [66] which based upon on the EAGLES (Expert Advisory Group on Language Engineering Standards) recommendations [128], in order to measure the performance of spoken dialogue systems. He defined key aspects of the system, the test conditions and the test results. His aim was to arrive at criteria which could facilitate comparison across systems (e. g. to compare the performance of two speaking agents performing the same task). The criteria also describe what to evaluate and report as well as how to do it. Furthermore, Dybkjar, Bernsen and Minker discussed the problems related to such metrics [55]. They argue that the methodology can be difficult to follow and may not fit equally well into projects with different usage contexts.

Möller provided an overview of interaction parameters (e. g. dialogue duration, number of system and user turns, etc.) [148] which have been used to evaluate SDS in the past 20 years. This overview provides a characterisation of these parameters, including the interaction aspect each one addresses as well as the measurement methods which are required to determine them. This work includes also an overall description of the parameter extraction process and the level at which these parameters are instrumented during the dialogue. The parameters and measurements addressed in [148] are also published in Suppl. 24 to ITU-T P-series Rec. [97].

Parameters to evaluate SDS (see above) have also been used as a basis to define metrics for the evaluation of MMDS [98, 122, 245]. A set of parameters to describe user interaction with multimodal dialogue systems is recommended in [98]. These parameters aimed at quantifying the interaction flow, the behaviour of the system and the user, and the performance of input and output devices. This approach from Kühnel, Weiss and Möller [122] was not only aimed at transferring some spoken dialogue parameters to a multimodal context, but also to provide new parameters which are inherent to multimodal interaction.

Furthermore, Wechsung et al. (under participation of Kühnel and Möller) presented a taxonomy of the most relevant aspects of *quality of service* (QoS) and *quality of experience* (QoE) [244]. Furthermore, they provided metrics which are able to capture QoS and QoE of multimodal human-machine interaction (loc. cit.).

The ISO 24617-2 standard [94] describes a semantic annotation framework (SemAF) which is based on dialogue acts. It describes a set of concepts (e. g. turns, functional segments and different types of relations between those) for dialogue annotation which are empirically as well as theoretically motivated. One important part of the framework is DiAML (dialogue act markup language) which is used to formally describe dialogue annotations. The concepts for dialogue annotation which are used in the standard are derived from concepts for the description of spoken dialogue, but they can also be applied to multimodal dialogues [94, 6].

Former approaches like PARADISE (PARAdigm for Dialogue System Evaluation) and PROMISE focused on assessing usability in a more predictive way, basing on parameters describing user efficiency and task success. PARADISE [240] is a framework to compare the performance of different dialogue management strategies in a spoken dialogue system. The framework considers user satisfaction as a measure of system usability, which is in PARADISE objectively predicted by measuring task success and dialogue costs. Task success is measured using attribute-value matrices, which describe the aim of a dialogue and the actual reached dialogue state. Dialogue costs are calculated using cost functions. Then, the importance of these values for the system performance is weighted via multiple linear regression.

PROMISE [18] extends PARADISE for the evaluation and comparison of task-oriented MMDS. The approach provides a new way to define system performance by splitting the performance function of PARADISE into two parts. For that, the performance function of PARADISE is reduced to a normalised cost function. Furthermore, an alternative way to calculate task success is defined in PROMISE. The result is a new formula to evaluate multimodal systems performance, since different attribute-value matrices (with different weights) can be computed for the different modalities.

Perakakis and Potamianos also used efficiency parameters for spoken dialogue systems in order to assess interactions in MMDS [172, 173]. They proposed efficiency and synergy as new objective metrics to identify usability problems in interactions with MMDS. In this approach, efficiency incorporates the added value from combining several input modalities during the interaction. The results demonstrate how multimodal systems should adapt, in order to maximise the synergy of modalities and to improve usability and efficiency of multimodal interfaces.

As written above, other related work focused on defining new parameters to determine the most suitable (i. e. efficient) combination of modalities and thus to maximise the system quality. One example for those approaches are the CASE (Concurrent, Alternate, Synergistic, and Exclusive) properties [161]. This classification space describes the properties of input and output modalities of multimodal user interfaces. The CASE properties classification are based on the concurrency of data processing, and the fusion of input or output data.

Coutaz et al. extended the CASE properties by the CARE (Complementarity, Assignment, Redundancy, and Equivalence) properties [40] to assess aspects of multimodal interaction with particular respect to the user input. *Complementarity*, *assignment*, *redundancy*, and *equivalence* are the properties of the interaction which are used to denote the availability of interaction techniques in a multimodal user

interface. For that reason, they can be used to predict the usability of an interface during the design of a system.

Finally, Lemmelä et al. described a 5-step iterative process for identifying issues affecting the usefulness of interaction methods and modalities in different contexts [129]. The described process mainly focuses on evaluating applications using tactile and auditory cues and provides a description of the parameters affecting the suitability of a particular modality in a specific context. The addressed parameters can be processed to select the best option for the current usage context.

2.4.2.2 Recording and Logging of Interactions

After the review of parameters and measures for the usability evaluation of interactive dialogue systems, the following provides an overview on methods for recording and logging the behaviour of human users which interact with such systems.

Balbo, Coutaz and Salber described an approach to record user interaction data with the aim to detect concrete user behaviour patterns, e. g. direction shift, action repetition, and action cancellation [12]. The discovered patterns were used to analyse deviations from a data flow-oriented task model and to detect potential usability problems.

For the evaluation of multimodal groupware systems, Damianos et al. used the Multi-Modal Logger [16] to record user behaviour [44]. They combined the recorded data with observations of usability experts and user feedback to detect usability glitches in systems under development.

The MATIS (Multimodal Access to Transaction and Information Services) system [225] can be used to find ways to maximise an interface's usability by combining speech and GUI interaction. Sturm et al. used MATIS to automatically log user interactions and to measure the usability of unimodal and multimodal interfaces. Furthermore, such data can be used to describe to what extent users notice and use the extra interaction facilities that are available in a MMDS (loc. cit.).

Tycoon [138] (TYpes and goals of COOperatioN), a theoretical framework to study the multimodal behaviour of observed human users, is described by Martin and Kipp. For the modalities which are used by the user during an interaction, the framework specifies four different types of cooperation which base on the CARE properties: equivalence, specialisation, complementarity, and redundancy. Furthermore, Tycoon offers a coding scheme and metrics aimed at analysing issues like how redundant the behaviour of a user is, how much the user relies on specific modalities, how the user switches between modalities, and so on.

Finally, a component-based approach for developing and evaluating multimodal interfaces on mobile phones is described by Serrano et al. They propose an evaluation process that captures usage data in realistic situations and implements in-field evaluations later [218]. In that approach data are captured at four different levels of abstraction (i. e. device, interaction, composition, and task level). This process can be used for a continuous user evaluation in an iterative development process, e. g. as in the usability engineering life cycle (see Sect. 2.1).

2.4.3 *Classification of Description Languages and Modelling Approaches*

Beside considering the results and insights of previous work on modelling and evaluating interactions with multimodal interactive systems, a method to compare such approaches is needed. An appropriate approach is proposed by Dumas, Lalanne and Ingold and will be introduced by this subsection.

Dumas, Lalanne and Ingold propose a set of nine guidelines for languages dedicated at multimodal interaction [54] (Table A.1 in Appendix A provides a short description for each guideline). These nine guidelines "are to be seen as a 'checklist' of potential features a given multimodal interaction description language can provide. By no means should every language follow all of them. Guidelines should be used as design tools, or as language analysis criterias [sic]" [54, p. 4]. Furthermore, four distinct roles are defined for which an interaction description language should target. These roles are communication, configuration, teaching, and modelling [54].

The guidelines one (G1) up to eight (G8) will be used to compare a set of representative languages and data models describing multimodal interaction by the features each one has (i. e. the guidelines each one fulfils). Section 4.5.2 will provide and discuss a comparison of PALADIN (which will be described in Chap. 4) against ITU-T Suppl. 25 to P-Series Rec., EMMA, ICO, and SOMBDE. All the mentioned approaches are used for the modelling role. The ninth guideline G* (find the right balance between usability and expressiveness) is not considered in that comparison, because it was not possible to analyse the usability of all named languages and models.

2.5 Measures for the Evaluation of User Behaviour Simulations

The former sections of this chapter introduced methods for the description of human-computer interaction: the human-, the model- and the simulation-based usability evaluation as well as tools for the automatic usability evaluation of dialogue systems. The current section gives an overview on methods for the evaluation of user behaviour simulations (UBSs). Here, an evaluation measures the ability of a user behaviour simulation to mimic the behaviour of human users when interacting with an interactive dialogue system. There are several approaches to simulate user behaviour as well as a multitude of metrics to evaluate such simulations. The Sects. 2.1.4.2 and 2.2 introduced mainly models and simulation approaches for the usability evaluation of interactive dialogue systems. However, the following paragraphs consider methods which were developed to measure the performance of UBSs for the purpose of usability evaluation but also (and in majority) for UBSs which are used for the training of dialogue managers (which was described in Sect. 2.1.4.2). This is not a

critical point, as all approaches aim at generating realistic (in the sense of natural) user behaviour.

Schatzmann, Georgila and Young, Schatzmann et al., Frampton and Lemon, as well as Pietquin and Hastie provide with their works an extensive overview on approaches for user behaviour simulations and related evaluation metrics [65, 175, 207, 209]. The suggestions and findings from all named works will be briefly presented in the following in order to motivate the work which will be presented in Chap. 6 of this book.

It should be noted that among the available literature Pietquin and Hastie [175] present the most systematic contribution regarding evaluation metrics and necessary properties (criteria) of such UBS evaluation metrics. Table 2.1 provides criteria which an evaluation metric should fulfil and which were defined by Pietquin and Hastie [175, 60–61]. The criterion *quality of learnt strategy* will not be considered here, as it addressees the training of a dialogue manager, what is not part of this book.

Schatzmann, Georgila and the linked work of Schatzmann et al. suggest considering six kinds of metrics/features:

1. High-level features of the dialogue: mean and distribution of turn length and task length, ratio of user and system actions (participant activity)
2. Style of the dialogue: similarity between simulated and real user responses (precision and recall), frequency of speech act types as well as speech act classes (goal-directed actions, grounding actions, dialogue formalities, unrecognised actions)

Table 2.1 Set of criteria describing the abilities of a metric which shall be used to evaluate a user behaviour simulation. The criteria and their meanings are defined by and extracted from Pietquin and Hastie [175, 60–61]

Criteria	The ability of a metric to …
consistency	Measure statistical consistency of generated dialogue acts with data
Consistent sequences	Measure the ability to generate consistent sequences of dialogue acts
Quality of learnt strategy	Assess the quality of learnt strategies when the user simulation is used to train a machine-learning-dialogue management system
Performance prediction	Predict the performance of an SDS with real users
Generalization	Measure the generalization capabilities of the method
Ranking and optimization criteria	Compute a scalar value to rank and optimize user simulation
Automatic computation	Automatically compute an assessment measure from objective information

3. Success rate and efficiency of the dialogues
4. Perplexity
5. HMM similarity using the Kullback-Leibler divergence
6. Evaluation of learnt strategies

The metrics of the items 1 to 3 are suggested and analysed in [207], the items 3 to 6 are considered (together with the three others) in [209]. Below, the metrics will be further explained.

Beside the definition of criteria for UBS evaluation metrics (see Table 2.1), Pietquin and Hastie analyse 12 different metrics (containing those of Schatzmann et al. and Frampton and Lemon) which are divided into turn-level and dialogue-level metrics [175, 61–68]. It should be noted that, in principle all interaction parameters (e. g. as defined in Suppl. 24 to ITU-T P-series Rec. [97]) at turn-level and dialogue-level can be used to compare dialogues generated by a UBS with empirical collected dialogues. Thus, the following does not describe the usage of certain parameters, but the general metrics for analysing them.

2.5.1 Turn-Level Metrics

All turn-level metrics have in common that they cannot measure the consistency of dialogue act sequences, and also the criterion of generalization is not (easy) to assess (cf. Table 2.1) [175, 61].

However, the frequencies and distributions of dialogue acts can be analysed. Furthermore, it is possible to compare the ratio of user and system acts, the classes of the dialogue acts (see above) as well as the user cooperativeness (provided slot values if requested) [175, 209]. Other common metrics are precision, recall and the related F-measure. These metrics are very general and further information on their computation will be provided in Sect. 6.4.1.3.

Finally, data which represent a distribution (e. g. frequencies of used dialogue acts) can be compared by the Kullback-Leibler divergence [175, 62]. The properties of the Kullback-Leibler divergence are in detail explained later in Sect. 6.2.3.3. However, at this point it is important to note that the Kullback-Leibler divergence is unbound (i. e. it can take any arbitrary value greater than 0).

2.5.2 Dialogue-Level Metrics

Simple dialogue-level metrics are for example the dialogue length, dialogue success and task completion (task success). The later can be evaluated using the κ coefficient [35] in order to compute how well information was transferred between user and system (e. g. as done in PARADISE [240]).

The three more complex metrics perplexity, log-likelihood, and hidden Markov model (HMM) similarity evaluate a UBS by statistical properties of the generated dialogues. While perplexity and log-likelihood directly compare empirical and artificial data, the HMM similarity compares two HMMs, each trained with either empirical or artificial data [175, 62].

The Cramér-von Mises divergence is refereed by Pietquin and Hastie as a measure that relies on an empirical distribution function. Thus, it makes no assumptions about the distribution of the compared data [175, 247]. Williams demonstrated the usage of the normalised Cramér-von Mises divergence for the evaluation (and compartment) of user behaviour simulations [247]. This metric uses a cost function to score each dialogue and compares the resulting score distributions by computing a distance value, i. e. the Cramér-von Mises divergence. The cost function does not consider the sequence of dialogue acts, but incorporates dialogue properties like task success or dialogue length. That measure was also used by Hillmann and Engelbrecht to compare four UBSs with each other [86]. A brief introduction into the Cramér-von Mises divergence and its application will be provided in Section 6.4.1.3.

Furthermore, Pietquin and Hastie name the two metrics BLEU (Bilingual Evaluation Understudy) [168] and SUPER (Simulated User Pragmatic Error Rate) [194, 195] which base actually on the word error rate measure [175, 67]. Both metrics evaluate the naturalness of generated utterances at the word-level [175, 65–66]. As this book focuses on UBS which work at concept-level, these metrics are not further considered here. Furthermore, measures which consider the strategy learnt by a dialogue manager (i. e. absolute performance of learnt strategy and strategy evaluation on real dialogue data in [175]) are also out of the scope of this book.

Finally, it also possible to evaluate artificial dialogues by human judges, e. g. as proposed by Ai and Litman [2, 175]. Here the judgements of human judges are used to train an automatic classifier. The main disadvantage of this metric is the need for human resources (the judges) and its time-consumption. Furthermore, the methodical procedure is still unclear [175, 67], which makes it difficult to obtain reproducible results.

2.5.3 Dialogue Act n-grams and Kullback-Leibler Divergence

Beside metrics at turn-and dialogue-level, n-gram models which base upon sequences of dialogue acts can be compared by a distance measure. For example, Georgila, Henderson and Lemon proposed [70, 175] the usage of the Kullback-Leibler divergence to compare distributions of dialogue act sequences. The consideration of sequences of dialogue acts addresses the criterion of consistent sequences (see Table 2.1). Furthermore, comparing n-gram models allows handling of unseen data, e. g. as it is common in language modelling (cf. [110, Chap. 4]). Here, unseen data addresses dialogue sequences which exist either in the artificially generated or the empirically collected data but not in both. Smoothing (see Sect. 6.3.2 for details) of n-gram

models makes a prediction for such data, and addresses the generalization criterion (Table 2.1).

However, the Kullback-Leibler divergence has the drawback of being unbound, which means its scalar value can be in the range of 0 to ∞. This makes it difficult to compare several UBS to each other [175, 247]. For that reason, Williams used the Cramér-von Mises divergence, because it gives a scalar value in the range 0 to 1. Unfortunately, it was only used to compare distributions of dialogue scores.

In applications in the field of information retrieval (e. g. [21, 36, 202, 228]) and language modelling ([110, Chap. 4] provides an extensive overview), further distances measures are known and were successfully applied to n-gram model comparison. Such measures, i. e. the Jensen divergence, cosine distance, and the rank order distance, as well as two variants of the Kullback-Leibler divergence, will be introduced and evaluated in Chap. 6, in order to rectify the just addressed shortcomings.

2.6 Research Questions

Until now, this chapter has presented current state-of-the-art on the description and evaluation of multimodal dialogue systems. For the usability evaluation of such systems, the high efforts of traditional methods and the possibilities of automatic usability evaluation has been shown. Furthermore, a socio-technical view on the observation and the analysis of human-machine interaction was presented in Sect. 2.3.

The main aim of this book is to describe methods for the improvement of automatic usability evaluation. This shall be achieved by the extension of the knowledge about multimodal human-computer interaction (especially about the courses of such interactions), as well as by the integration of this knowledge into user behaviour simulations. Beside the description of certain approaches which are appropriate to reach that aim, the book does also cover the descriptions of related methodologies. This affects the observation and analysis of human-computer interaction, and the evaluation and the comparison of user behaviour simulations.

Section 2.2 described several existing tools for automatic usability evaluation. They all differ from each other in their underlying approaches and the supported interaction modalities. Still, one AUE tool is needed as base for the work described in this book. The development of a totally new tool would have been to elaborated, which is why the MeMo Workbench and the MeMo user model were selected as the base for the intended extensions. The reasons for the selection of the MeMo Workbench and the neglect of the other considered tools were already stated in Sect. 2.2.6.

Section 2.4 of this chapter introduced a variety of approaches for the description and logging or annotation of multimodal interaction. Nevertheless, it is unclear which information about the interactions is needed to achieve an optimal usability in a usability engineering process which is supported by AUE methods. For this, it would be helpful to parametrically describe and quantify interactions in real usage scenarios as well as simulated interactions.

Q_1: Which parameters can be used in an AUE process to describe interactions? How can such parameters be efficiently collected?

There are already descriptions of parameters describing interactions with spoken dialogue systems [97, 148] and their extension for describing interactions with multimodal systems [98, 121]. However, these works do not gauge how these parameters can be efficiently collected neither from empirical experiments nor in simulation-based evaluation scenarios. For that reason, an interaction logging framework for the practice-oriented analysis and description of multimodal interactions (PALADIN) [140] was developed. PALADIN and its application to the evaluation of interactive systems will be described in Chap. 4.

To sum up, Q_1 asks for parameters in observed interactions regardless of the source of the interactions. This can be an empirical study or a simulation-based experiment. Q_2 is related to the latter, and considers the simulation of user behaviour in the context of automatic usability evaluation.

In Sect. 2.3, it has been shown that user behaviour is largely based on expectations which have been built in the past through prior exposure during the socialisation of the user. These expectations about the operation and functionality of electronic devices result in interaction patterns which can be observed in interactions between users and interactive systems. For a better understanding, such patterns are called interactivity patterns.

Q_2: How can interactivity patterns be integrated into an automatic usability evaluation process?

Chapter 5 will describe a socio-technical approach for the analysis of interactions between real users and interactive systems in order to identify interactivity patterns. These pattern were integrated into an automatic usability evaluation tool, the MeMo Workbench. The dialogues which are generated with the new interactivity pattern-based MeMo user model are more similar to the interactions of real users than former approaches [60].

Once interaction patterns have been artificially generated and parametrically described, it is important to evaluate in how far the generated dialogue sequences of a user behaviour simulation are similar to usage patterns of real users. For this purpose the user behaviour simulations to-be-evaluated have to be compared with interactions of real users. Furthermore, such an evaluation is not only in this case necessary, but a general requirement on the development process of such simulations. Section 2.5 introduced existing metrics and their shortcomings regarding the comparison of dialogue act sequences.

Q_3: How can the performance of user behaviour simulations be measured?

Chapter 6 will provide a method for comparing observed empirical and artificially generated dialogues at the level of dialogue act sequences which reflect interactivity patterns. The evaluation is based on the computation and analysis of distance measures between collected empirical data and one or several data sets which are generated by the user behaviour simulations to-be-evaluated. Different distance measures will be analysed and evaluated according to their ability to distinguish sets of

dialogues regarding dialogue length, task success, word accuracy, and overall user judgement, as well as the origin of the dialogue pattern (i. e. empirical study with real users or a simulation experiment). An open source software which determines the optimal distance measure for certain dialogue data and which can be used for the evaluation of user behaviour models is provided as additional contribution of thesis published in this book.

http://www.springer.com/978-3-319-62517-1

Simulation-Based Usability Evaluation of Spoken and Multimodal Dialogue Systems

Hillmann, S.

2018, XXX, 240 p. 31 illus., Hardcover
ISBN: 978-3-319-62517-1