# A Companion-System Architecture for Realizing Individualized and Situation-Adaptive User Assistance

**Pascal Bercher**[1a], **Felix Richter**[1b], **Frank Honold**[2b], **Florian Nielsen**[3b], **Felix Schüssel**[2b], **Thomas Geier**[1b], **Thilo Hörnle**[1b], **Stephan Reuter**[4b], **Daniel Höller**[1a], **Gregor Behnke**[1a], **Klaus Dietmayer**[4a], **Wolfgang Minker**[3a], **Michael Weber**[2a], **Susanne Biundo**[1a]

[1] Institute of Artificial Intelligence, Ulm University, Ulm, 89081, Germany

[2] Institute of Media Informatics, Ulm University, Ulm, 89081, Germany

[3] Institute of Communications Engineering, Ulm University, Ulm, 89081, Germany

[4] Institute of Measurement, Control and Microtechnology, Ulm University, Ulm, 89081, Germany

Email: firstName.lastName@{[a]uni-ulm.de , [b]alumni.uni-ulm.de} (replace "ö" by "oe" and "ü" by "ue")

***Abstract***—We show how techniques from various research areas – most notably hierarchical planning, dialog management, and interaction management – can be employed to realize individualized and situation-adaptive user assistance. We introduce a modular system architecture that is composed of domain-independent components implementing techniques from the respective areas. Systems based on this architecture – so-called *Companion*-Systems – can provide intelligent assistance in a broad variety of tasks. They provide a user- and situation-adapted sequence of instructions that show how achieve the respective task. Additional explanations are, like the instructions themselves, automatically derived based on a declarative model of the current task. These systems can react to unforeseen execution failures repairing their underlying plans if required. We introduce a prototype system that assists with setting up a home theater and use it as a running example as well as for an empirical evaluation with test subjects that shows the usefulness of our approach. We summarize the work of more than half a decade of research and development done by various research groups from different disciplines. Here, for the first time, we explain the full integration of all components thereby showing "the complete picture" of our approach to provide individualized and situation-adaptive user assistance.

***Index Terms***—Companion Technology, Assistance Systems, Planning-based Assistance, Human-Computer Interaction (HCI)

## I Introduction

At present, many innovative technological developments find their way into marketable products. As a consequence, most of the technical systems we constantly use in our everyday lives – household appliances, entertainment electronics, smart phones, vending machines, and electronic services of all kinds – are becoming increasingly "intelligent". Their functionality is changing, becoming more versatile and complex. However, the functional intelligence of these systems often contrasts strongly with their lack of user-friendliness and ease of operation. More importantly, they are static, i. e., identical for all users in all situations. Voluminous instruction manuals, rigid user instructions and non-existent user assistance are just a few of the obstacles that significantly hinder the barrier-free and truly user-friendly implementation of these systems.

*Companion*-Technology aims to bridge this gap by complementing the functional intelligence of technical systems with an equivalent intelligence in interacting with the user and by integrating the two [35]. It enables the implementation of so-called *Companion*-Systems, which assist users by providing to-the-point instructions and explanations in a completely individualized way: they adapt to the user's knowledge of the application at hand, to his or her capabilities, and to the current situation. Furthermore, they are able to communicate and interact with the user via various input and output channels, and they aim to gain their users' trust by making their behavior and functionality as transparent as possible. *Companion* characteristics of technical systems, such as *individuality*, *adaptability*, *flexibility*, *transparency*, and *trustworthiness* are created through the realization and interplay of various cognitive system processes. For a detailed overview about the various techniques and research areas that are combined to realize these capabilities, we refer to an overview article about the field [34] and to a recent book about *Companion*-Technology [17].

In this report we present an approach to realize *Companion*-Systems that assist users in complex tasks in a user- and situation-adaptive way. One of its major strengths is that all components of the underlying system architecture are completely domain-independent. Thus, one only requires to equip them with the required models and further data like pictures and videos of the current application at hand, but no software needs to be adapted. The architecture comprises a knowledge base with reasoning capabilities; hierarchical planning modules with plan generation, plan repair, and plan explanation facilities; a dialog and explanation manager; an interaction manager to handle multimodal input and output; and an advanced user interface.

*Companion*-Systems implemented via this architecture work as follows. Given a complex task to solve, first, an action plan is generated automatically. It provides a sequence of actions the execution of which accomplishes the user's task. This plan serves to create step-by-step instructions for the user. As plans are generated on a declarative model of the task at hand, the system can – independent of the current application domain – answer questions about the instructions generated, elucidating, e. g., why a particular step is part of the plan. Such plan explanations are an essential means to make the system's behavior transparent. Furthermore, a plan repair component enables the system to respond to unexpected execution failures by providing adequate support in those cases. Dialogs between the system and the individual user are realized by the dialog management component of the system. It is concerned with controlling both the structure and the content of this dialog and is, among others, responsible for transforming the instruction steps into concrete dialog steps. It does so by taking the user's knowledge and preferences into account to ensure that the instructions are presented in an individualized and adequate manner. This includes improving the user's understanding of the task at hand by offering explanations about relevant concepts and instruction steps, thereby contributing significantly to establish and maintain the user's trust. A *Companion*-System needs to be flexible and able to communicate with its user in various environments and situations. Therefore, it is equipped with an advanced interaction management that realizes multimodal user-system communication and interaction. It provides appropriate user interfaces and is able to receive and interpret multimodal user input. Choosing the right modality or an appropriate combination of different modalities is important for both adapting to the current environment or situation and conducting an individualized user dialog. *Companion*-Systems are knowledge-based systems: The various components that implement their functionality and characteristics rely on a multitude of different pieces of information that are required and produced and that need to be exchanged. Therefore, a central knowledge base serves to represent and combine the different models used. It is also responsible for monitoring the world state as well as the user's situation. To this end, a probabilistic approach is employed to ensure that inferences relying on possibly uncertain sensory input are robust.

To prove that the proposed architecture and the various components are appropriate to realize *Companion*-Systems of rich functionality, we implemented a large-scale prototype that assists users in the task of *setting up a complex home entertainment system* [16, 59]. This prototype, which also serves as a running example for this report, assists its user by its manifold capabilities. In a nutshell, it generates – based on AI planning – a sequence of actions that serves as a basis to present the instructions showing the user how the various HiFi devices should be connected with the available cables and adapters. The dialog manager then decides about the level of abstraction of these steps, i. e., to decide what information is conveyed to the user, thereby taking his or her knowledge about the specific application domain into account. Finally, the interaction manager decides about the most appropriate way how (and where) this information is presented. In addition, the system can produce repaired solutions in case cables turn out to be broken during execution. Further, at any time during interaction, the user can ask questions about the currently presented instruction, including knowledge about the involved hardware as well as about the purpose of the respective instruction for the overall goal.

**Contributions** We propose a modular system architecture that is composed of domain-independent components implementing techniques from knowledge representation and reasoning, hierarchical planning, and dialog and interaction management. It allows the realization of so-called *Companion*-Systems – systems that provide individualized and situation-adaptive assistance to their users in a broad variety of different applications. Partial integrations of the proposed approach and system were published before with focus on one particular aspect/capability of the architecture and system (such as planning [45, 55] or adaptive dialogs and multimodal interaction [69]), but without highlighting the others and their interplay in particular. In a short book chapter, we sketch the entire system giving pointers to the respective related work of this system by the involved research groups [16].

In this report we introduce the entire system in detail, thereby showing “the complete picture” of our approach dealing with all integration issues. We explain how all involved system components are communicating with each other and shortly explain their underlying technology and give pointers to related work for further details. We also summarize the main findings of our empirical evaluations that were done for these involved technologies (such summaries have also not been given before) as well as for partial integrations of the systems, and give pointers to the original publications. In this report, we also include one novel evaluation that was not published before. Another contribution is the related work section, where we give – to the best of our knowledge – the first detailed overview of other planning-based assistant systems and architectures.

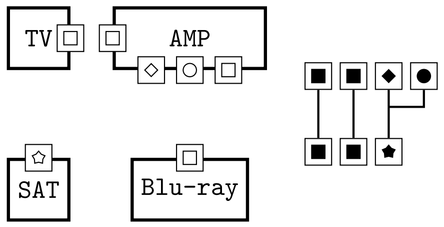
**Outline** The remainder of the report is organized as follows. Section II introduces the application domain that serves as a running example throughout the report. It is concerned with the setup of a complex home entertainment system. Section III presents the overall architecture in an abstract way, describes the core functionality of its sub components, and the general interplay between them. A more detailed view on the interplay of the various components is given at the beginning of the sections on the particular sub components and disciplines. We start by explaining the knowledge base, which serves as central hub of the system’s knowledge (Section IV). The following sections are organized in the order of one user-system interaction loop: We start with the planning components that are responsible, among others, for generating the plan of actions, the presentation and execution of which supports the user (Section V). Next, the dialog components are explained, which are responsible for presenting the current instruction in a user- and situation-adaptive way (Section VI). The interaction management is then described, which is responsible for modality-independent output, as well as for handling multi-modal user input (Section VII). Finally, the user interface is described, which serves as the main interface for the user to communicate with the system (Section VIII). All these sections discuss lessons learned from applying the respective technologies in the described system, their limitations, mention real-time issues (including evaluations), as well as future work. In addition, Section IX gives a rough overview for the main steps that need to be taken when building such a system in a specific application scenario. Here, we particularly highlight issues that arise from integrating the different techniques into a single system. Furthermore, we try to give an intuition about the effort to come up with such a system by providing suitable examples from our application domain and prototype system. Section X reviews related work – both research projects as well as work on actual systems – that is concerned with a similar endeavor of providing user assistance based on AI planning. Section XI then concludes this report with some final remarks.

## II The Home Theater Assembly Task

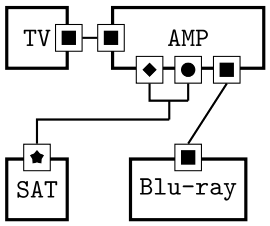
The following sections introduce an architecture and the involved components (and their underlying research disciplines) that serve as the basis for a system that provides advanced assistance functionality in a wide variety of tasks. To exemplify our approach, we have implemented a prototype system in a specific application domain, in which we believe many people would appreciate automated and intelligent assistance (which we regard confirmed by one of our empirical studies, cf. Section E). This application domain further serves as a running example throughout the report. As noted before, our approach is fully generic and can be applied to many different problem scenarios (cf. Section IX) – our example scenario mainly serves as proof of concept and for illustration purposes.

Our running example is concerned with providing assistance to a user who wants to set up his or her home theater. The theater consists of several HiFi devices that need to be connected with each other in such a way that each of the devices is fully functional. To that end, the user must pick the correct cables and, if required, adapters to connect the devices in a correct fashion, so that each device receives the required audio/video signals [55]. In the specific example that is used in our implementation, there are the following devices: a television, an amplifier, i. e., an audio/video receiver (Figure 2 shows its back panel), a Blu-ray player, and a satellite receiver. For connecting the devices with each other, several cables, adapters, and adapter cables (cables with different kinds of ends) are available to the user. The home theater is fully set up as soon as two main goals are fulfilled: (1) the television needs to receive the video signals of both the Blu-ray player and the satellite receiver, and (2) since the loud speakers are attached to the amplifier, it needs to receive the audio signals of the two devices.

The home theater assembly task was chosen for our prototype system and hence as an illustrating example, because complicated technical systems that need to be set up via connecting them in the right fashion can be found in many households, which makes it a commonly known task. Also, many people do not consider themselves an expert in such technical domains; only a few concepts are known, but not all the details about the (interaction of the) variety of ports, cables, and adapters.



(a) The task: an assortment of unconnected devices and cables.



(b) A solution: the devices are properly connected.

Figure 1: The figure schematically illustrates the task [25, Fig. 6.1]. It shows the available devices, some of the available cables, and how they are compatible with respect to each other: The TV, amplifier, satellite receiver, and Blu-ray player each have various female ports, where □, ☆, ◇, and ○ denote HDMI, SCART, cinch video, and cinch audio ports, respectively. Black ports on cables ■ denote male ports. There are two HDMI cables and one SCART-to-cinch-AV cable. The devices should be connected in such a way that the video signals of the Blu-ray player and the satellite receiver reach the TV. The respective audio signals should be transported to the A/V receiver, which is connected to speakers.

Figure 2: The figure shows the back panel of the amplifier used for our running example [55, Fig. 1].

There is also the combinatorial aspect in connecting the devices, which becomes apparent if there is only a limited number of cables available[1]. Solving such combinatorially hard problems is one of the strengths of AI planning technology, which makes it a reasonable application scenario. It also nicely shows the strengths of *Companion*-Technology: According to its vision, technical systems of the future will be *Companion*-Systems [35]. These systems will be equipped with a model of their own functionality that can directly serve as a basis for the required models without the need to build them by hand. Furthermore, it shows that *Companion*-Technology goes far beyond simple instruction manuals, as those can only give a description of a *single* device/cable, or a *predefined* set of devices (which could be part of a set). Thus, at the most, such a manual could explain how to set up the devices which are known to be part of a bundle, but only a generic approach like ours can assist in this task when the available devices are specified by the user.

The implemented system runs in an example setting, where the user has two possibilities for interaction: there is a standard desktop PC that has a large touch screen monitor and two loud speakers attached to it as well as a laser range finder for user localization (cf. Section 5). A second device, a standard desktop computer, is placed a few meters next to the touch monitor. That computer also has a separate pair of loud speakers attached to it. An additional complication in the setting stems from the possibility of further persons that, apart from the user, enter and leave the scene and have to be ignored by the system. Also, it cannot be assumed that the scene is free from obstacles, i. e., occlusion is likely to occur. Together, these complications account for the typical scenario in which assistance for setting up a home theater takes place – at the user's home, where family members and additional furniture might be present.

## III System Architecture

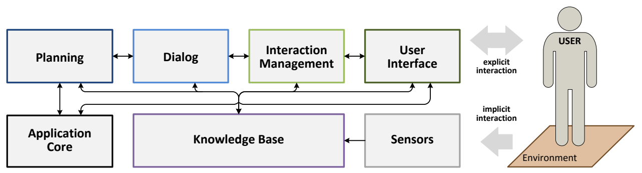


Figure 3: The underlying architecture of the assistance system. A more detailed view on its sub components and their interaction within the system is given at the beginning of the respective sections.

Our architecture (Figure 3) is based on established architectural principles for interactive systems, like Arch or Arch/Slinky [180], as well as architectural recommendations [78, 107]. Moreover, the presented architecture and its components respect established practices and approaches like model-driven software development [161] and automatic UI generation using different levels of abstraction, based on the CAMELEON reference framework [168, 160].

[1]In fact, we believe that the problem is in general NP-hard; we do, however, not yet have a formal proof. NP membership is obvious, since one can simply guess a setup and check its validity in polynomial time.

With these principles, we integrate planning capabilities (plan generation, execution/monitoring, repair, and explanation) with components providing advanced human-computer interaction, which include dialog management (explanation and dialog manager) and interaction management (multimodal fission and fusion, content manager, and nominator) in a knowledge-based system. Related is the work by Stahl et al. (2005) [152], who present a layered architecture for assistance systems in intelligent environments.

### A Subsystem Interaction

Figure 3 gives an abstract view on our system architecture, where every component is depicted by one single box. Many of these components consist of several sub components, however. A more detailed view on the system architecture is given at the beginning of the respective sections, where these sub components (and their interaction) are depicted. Here, we first give an abstract view on the involved components and their main functionality before we explain their interplay via one user-system interaction cycle.

**Application Core Module** One main ability of *Companion*-Technology is to act as enabling technology. The presented modules with their generic interfaces can act as a wrapper for pre-existing applications, tools, or services. By extending an application's core functionality, e. g., a media player, the basic system can be operated with the use of the generic components [111, 94]. With these possibilities, the user perceives the former system as *Companion*-System. Aside from wrapping an existing app to enhance its usability, another ability is to make use of existing web services for information retrieval. This can for example be used to derive user-specific or user-requested data and media.

**Knowledge Base Module** The knowledge base component (Section IV) organizes the required knowledge for the whole system. With the use of specialized sub-systems, various modeling paradigms can be used to meet the different requirements of other modules. Static or quasi-static context knowledge, for example, can be organized using less complex approaches than time-dependent data or data of high structural complexity. Knowledge representation is further complicated by the fact that a large part of the knowledge is of an uncertain nature. To maintain a consistent representation of time-dependent, uncertain information, the knowledge base component employs probabilistic temporal filtering to its input. The filtered current belief is offered to other modules through an appropriate query interface.

**Planning Module** The planning module (Section V) enables the overall system to realize goal-directed behavior. Thus, it provides one of the core functionalities of a *Companion*-System. Based on a model of the given application domain and a description of the user's goals that shall be achieved, it generates a step-by-step description of how it can be reached. This is necessary in several situations: it can solve a problem for the user and guide him or her via instruction on how to solve it. For this, the steps are successively passed on to the dialog management to initiate presentation to the user. The planning modules are further responsible for deciding the execution order of these steps (taking the user into account) and to detect unforeseen environmental changes during plan execution. When such a change prohibits the further execution of the plan at hand, a plan repair component enables the system to find a new plan that is capable of dealing with the changed environment. To enhance the user's benefit of the system's guidance and enhance the user's trust, it is able to explain its behavior, which is based on so-called plan explanations.

**Dialog Management Module** In general, a dialog manager serves as a flexible interface between an application and the user (Section VI). Specifically, it is used to map a current plan step to user interface concepts for user participation. The dialog manager employs diverse emotion-, knowledge-, and skill-specific dialog strategies to adapt the system to the individual user. In the same way, explanations about the current task as well as user-initiated requests for additional information are provided by the explanation manager. The dialog manager's modality-independent output represents the so-called abstract user interface (AUI) [160], which is further refined by the interaction management. Once the output for a plan step and the resulting input is processed, control is passed back to the planning module.

**Interaction Management Module** The interaction management's fission component analyzes the dialog management's modality-independent output, decides about the modality-specific output (Section A), and provides a so-called concrete user interface (CUI) [160]. This CUI description is then passed on to the user interface components (Section VIII) for rendering. As depicted in Figure 3, the decision process depends on context knowledge and highly dynamic data (e. g., the user's location). Such data is provided by the knowledge base. The content manager (Section C) analyzes the fission's output and provides a corresponding configuration for the fusion component (Section B and C). The fusion component is able to handle diverse user inputs from multiple modalities. It then passes the fused input back to the dialog management for further dialog processing. Also, the fusion component is able to detect the user's wishes about specific output configurations and media use. These so-called *nominations* are organized by the nomination manager and can influence the fission's process of modality arbitration (Section C).

**User Interface Module** The user interface module (Section VIII) comprises diverse rendering components for different input and output modalities. Whenever a new CUI is available, the addressed output components are able to interpret the CUI description and render the modality-specific final user interface (FUI) [160]. For input components, the CUI serves as configuration for the internal sensors (e. g., a situation-, action-, or user-specific grammar for automatic speech recognition (ASR)). Different inputs from multiple modalities are forwarded to the knowledge base as well as to the fusion component for multimodal input processing.

**Sensor System Module** Using different sensors (physical, logical, or virtual [136, 70]), the sensor system module provides the capability to recognize the environment (e. g., illumination- and noise level, persons present in the scene) as well as the user's implicit and explicit actions (e. g., implicitly trigger an action by moving from one spot to another vs. explicitly touching an item on a screen). Their inferred data is passed on to the knowledge base for further processing, as well as to the input fusion component in order to detect possible implicit user interactions.

**Exemplary User-System Interaction Loop** We now describe an exemplary life cycle of one user-system interaction loop. The interaction cycle starts by generating an initial plan, which consists of the current (initial) world state and the user's goals (which are given in terms of logical formula describing the desired world state and in terms of abstract tasks the user wants to have achieved). This initial plan is refined by the Planning Module, i. e., abstract tasks are decomposed successively into a partially ordered set of primitive tasks (actions), which can be executed. These actions are passed on one-by-one to the Dialog Management. The Dialog Management translates each action into a user-adaptive dialog, selecting a dialog model whose dialog steps are necessary to achieve the desired effects of the action, but at the same time is most suitable for the current user. For this decision information coming from the Knowledge Base, namely user emotion or user knowledge, is taken into account. For example, if necessary, additional dialog steps, explaining missing knowledge, are included in the course of the dialog. These dialog steps are in turn passed on one-by-one to the Interaction Management, which selects an adequate combination of diverse user interface components for the present situation to be presented on the available and selected devices. The results of the user interaction are processed and fused by the Interaction Management, which is able to combine user inputs from different modalities, such as pointing gestures in conjunction with verbal cues, and passed on to the Dialog Management. The Dialog Management maps the results to the effects of the current action. These effects are transmitted back to the Planning Module, and the world state in the knowledge base is updated accordingly. Hereafter, the cycle begins anew, i. e., the next action is passed on to the Dialog Management. However, if the interaction results do not match the system's expectations, the ongoing plan must be repaired by the Planning Component, i. e., it searches for an alternative solution, before re-initiating the cycle.

## B *Technical Realization*

A *Companion*-System that is realized based on the proposed architecture includes a number of software components that may be implemented in different programming languages and may run on several operating systems. It is important to be able to operate across different types of devices distributed in a network to fit the components' needs. These range from mobile devices like smartphones or tablets, e. g., for interface components, to powerful computers to be able to run, e. g., the planning component. To connect these distributed and heterogeneous components, a flexible and efficient middleware is necessary. The used middleware is an adaptation of the middleware from the SEMAINE project [106]. It realizes a client-server-architecture with a central broker instance, called the *system manager*, which offers a central connection point for the communication between the clients in the network. The actual communication between the components is asynchronous and message-based. It implements the publisher-subscriber pattern. Though we realized the system using standard Ethernet components, we found neither the communication nor the central server component to be the critical part when aiming for fast response times.

So far, the components of a system are started based on a predefined configuration and are thus statically defined for a given overall system. The system manager checks the current state (online or offline) of all components in the system. Currently, the system manager reports problems to the operator, but though the integrated components might be able to handle changes of, e. g., the set of available sensors (see Section VIII), it is not able to automatically reactivate components that are offline or re-configure the system. For the sake of reliability and overall system performance, the system manager should be able to restart components or replace them with others that are not part of the current configuration. Thus, as future work, we want to realize an automated dynamic configuration that can deal with failures of components or communication. To realize these capabilities, the system manager, which is responsible for the configuration, needs to be provided with knowledge about the different components that includes not only definitions of interfaces, but also a description of the functionality of each component. To preserve short response times, a special attention needs to be payed to organizing sensor components and which data needs to be sent over the network. Such an approach may even optimize the overall system performance by choosing a good configuration from all available components or by starting new components to realize an overall system with maximum utility to the user [49, 23].

## IV Knowledge Base

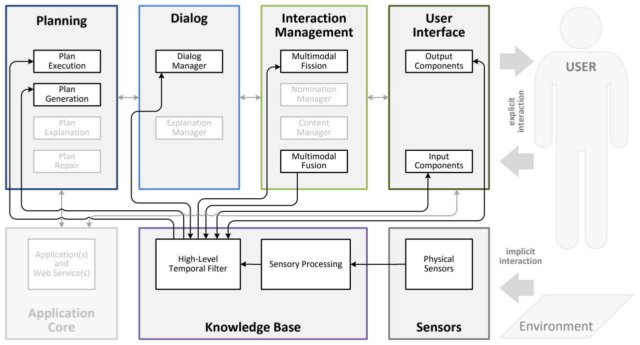


Figure 4: Architectural view and modules with focus on the knowledge base.

The knowledge base component serves as a central hub for the knowledge and beliefs of the whole system. Its main functionality is the maintenance of a time-dependent, probabilistic state representation which includes the state of the user and the system's general environment. As the knowledge base interfaces with many other modules, it has to support a large portion of the respective modeling paradigms. While there are approaches in which one common representation language for all the involved components is chosen [44, 26], here we use the individual models and ensure that they are used consistently. For example, the model used by the planning components (hybrid planning, cf. Section V) is time-dependent and based on a deterministic first-order logic. The components concerned with interaction management, on the other hand, use a flexible, deterministic model that allows specification of text fragments, images, or even general files as values [83, 61] to describe time-independent information about dialogue. Finally, the models used to process the sensor data are usually time-dependent and probabilistic. In addition, they usually make heavy use of continuous variables, such as the user's position, which is required to determine adequate output modalities (see also Sections B and A).

Figure 4 provides an overview on the components of the architecture that communicate with the knowledge base. The knowledge base itself consists of a high-level temporal filter (see Section A) as well as the sensory processing module (see Section B), which provides a dynamic model of the system's environment. Based on the data of physical sensors, the module generates a probabilistic output, which represents the uncertainty in the number of objects as well as the uncertainty about the individual object states and delivers this information to the high-level temporal filter. In addition to the sensory processing, the high-level temporal filter of the knowledge base obtains input from the dialog, the interaction management, and the user interface. The multimodal fusion of the interaction management delivers information about the user's interaction history to the knowledge base, which facilitates the adaptivity to the user's preferences in succeeding interactions. The input and output components deliver their capabilities as well as so-called heartbeat messages to the knowledge base. Further, the usage of the input components enables the knowledge base to draw conclusions about the user's position. In addition to the aggregation of all available inputs, the knowledge base provides the planning problem to the plan generation component. The context knowledge about the system's environment as well as the user's position are delivered to the multimodal fission module in order to choose suitable devices to present the system output.

### A Temporal high-level filter

Because of the diversity of the different modeling approaches, it is important to be selective in the choice of supported features. An adequate treatment of uncertainty and temporal modeling aspects appeared to be essential to facilitate a proper integration of sensory input, and to honor the fact that the system has to act within a dynamically changing and only partially observable environment. This is achieved by having the knowledge base component maintain a probability distribution $p(x_t|Z_{0:t})$ over the current state $x_t$ at time $t$, given all past observations $Z_{0:t}$. This task is usually called *filtering*. To facilitate a seamless integration, e. g., with the logical planning formalism, and to support modeling by human experts, we finally settled with expressing the central model of the system using *Markov Logic* (ML) [139]. ML has been used to integrate low-level and high-level data in a probabilistic setting on many different occasions [36, 57, 101, 102, 105, 121].

ML is an instance of a first-order probabilistic language [120]. As such, it allows the concise specification of probability distributions over a relational object structure. Since the semantics of ML are based on undirected graphical models [108], the integration of arbitrary logical constraints is straight-forward, and ML appears as a generalization of a restricted version of first-order predicate logic [139]. The ML model consists of a set of weighted first-order logical formulas, usually containing free variables.

To illustrate the use of ML for the described architecture, we present the following simplified fragment from a model [81, 82] that is able to integrate object localization information (and Section B) with respect to a discretized localization scheme (Figure 5):

$$\infty \quad \texttt{at}(t,p,l) \wedge \texttt{at}(t+1,p,l') \Rightarrow \texttt{adj}(l,l') \tag{1}$$
$$2 \quad \texttt{at}(t,p,l) \Rightarrow \texttt{at}(t+1,p,l) \tag{2}$$

To obtain the probabilistic semantics, the formulas are instantiated (grounded) by assigning a time-step to $t$, a discrete location (i. e., a colored region as in Figure 5) to $l, l'$, and a person to $p$. The groundings of formulas with higher weights are more probable to be true, while formulas with infinite weight must not be violated at all. Equation 1 then describes that a person can only move from location $l$ to location $l'$ within one time-step, if both locations are adjacent. Equation 2 models the fact that if a person $p$ is at some location $l$ at time $t$, then this condition will likely still hold at time $t+1$; while a negative weight would state that it is unlikely for a person to remain within the same location. Potentially uncertain observations of the user-position can then be integrated in the model through observations on the `at` atoms [81]. Given a set of such weighted formulas $\{(w_k, \phi_k)\}_{1 \le k \le n}$, and writing $n_k(x)$ for the number of true groundings of formula $\phi_k(x)$ under interpretation $x$, the joint probability of the model is defined as

$$p(x) = \frac{1}{Z} \prod_k \exp(w_k \cdot n_k(x)). \tag{3}$$

The partition function $Z$ must be chosen such that $p$ is normalized. Equation 3 states that an interpretation has a higher probability of being the true interpretation, when it satisfies more positively weighted groundings of formulas. With respect to our example, an interpretation in which a person stands still for a longer duration is more likely, because it satisfies more groundings of formula 2.

Inference within a ML model can be performed either *grounded* or *lifted*. Grounded inference works by converting the ML model to a Markov network, and applying conventional inference algorithms [108]. Because the grounding can cause an exponential blow-up

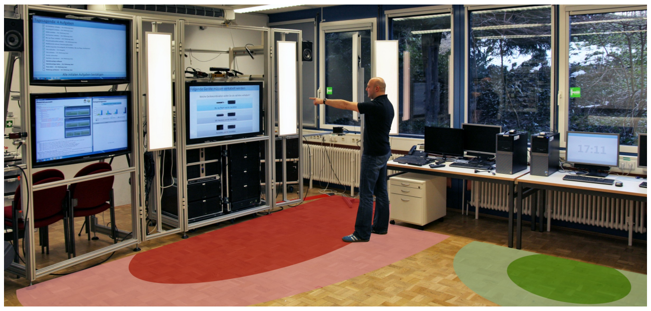


Figure 5: The figure shows the environment in which our prototype system was set up. There is a user making a pointing gesture towards the main terminal of the system. On the left and right side of this terminal, there are laser range finders for user localization (Only the one on the right is part of the picture). On the right is another desktop computer that can be used for user interaction as well. The colored areas depict the discrete locations of the MLN model that is used for user localization.

in model size, the more complex lifted approaches to inference try to avoid this step and work directly on the first-order representation [125], exploiting potential symmetries. As most symmetries are destroyed by observations, we followed a grounded approach to inference [92].

Interfacing to the other modules of the system is restricted to (potentially uncertain) observations of current variables, and queries to the marginal distribution over current variables. Observations are supplied by sensor-based modules such as user localization (see Section B), or modules related to explicit user input. An example of the latter is the observation of the current user location based on touch interaction with the system. A further stream of observations comes from the dialog management component at the end of the human-computer interaction cycle. It verbalizes some of the dialog steps (see Section VI) to the user and then receives a confirmation of their execution. We treat this confirmation as an assertion that the effects associated with the action have occurred, and can thus be observed.

Other modules query the marginal distribution over the random variables of the present time step. Among the consumers of this information are modules that are concerned with decision making, namely the plan generation component, the dialog management component, and the fission component. Here an important limitation becomes apparent, caused by the modular approach and its separation of the different components for inference and decision making. Within our implementation, only marginal distributions over single variables were queried, despite the distribution represented by the knowledge base component can represent dependencies between variables. The assumption of independence greatly simplifies reporting the results of queries as the representation of multi-variate distributions can become exponentially large in the number of variables, but it may also cause an over-simplification. A solution to this problem would either be more sophisticated result representations (for example through variational approximations [122]), or the unification of probabilistic inference and decision making within the same module.

Another limitation concerns the restriction to discrete random variables. A model containing continuous variables is more appropriate in many situations (for example when modeling user location, or emotional states). The extension of ML models by continuous random variables has been examined in the literature [123].

### B Sensory Processing: User Localization

The data that originates from physical sensors is not directly processed by the high-level filter, as it is often of a quite different nature (high-dimensional, continuous state, high-frequency). For this purpose there exist dedicated probabilistic models that abstract the sensory information to make it compatible with the knowledge base, which supports models with discrete-valued variables that often exhibit a combinatorial component. We integrated one such sensory pre-processing module in the prototypical implementation. The purpose of this module was multi-object tracking, based on the measurements of a laser-range-finder. A reliable method of acquiring the user's current location is important in the context of ambient intelligence [144]. The current position of the user obviously restricts the interaction modalities, e. g., a touch-interaction is only possible if the user is located close to the screen. Thus, the continuous estimation of the user's kinematic state is required. This is typically performed using a *recursive Bayes filter* which estimates the probability density function $p(x|Z_{0:k})$ of the object's kinematic state using the measurement history $Z_{0:k}$, i. e., all measurements up to the current time $k$. The aim of the user localization module is to transmit the dynamic state of all persons in the proximity of the system to the knowledge base.

Since the considered scenario comprises several persons in the proximity of the system as well as occlusions due to static or dynamic obstacles, more advanced tracking algorithms are required to obtain the required accuracy of the state estimates. Mahler (2007) [130] proposed the multi-object Bayes filter which tackles the problem of estimating the number of objects in the environment as well as their individual states in a mathematically rigorous way. The multi-object state is represented by a random finite set (RFS), which comprises a random number of (unordered) points whose individual kinematic states are random. Thus, an RFS naturally represents the uncertainty of the multi-object state since the number of objects as well as their individual positions are unknown.

In this work, a sequential Monte Carlo (SMC) implementation of the full multi-object Bayes filter is used to track the persons in the system's environment. Compared to the approximations introduced in the previous paragraph, the proposed SMC implementation facilitates the incorporation of object interactions, based on physical constraints [71, 63], i. e., the prediction step ensures that objects may not overlap. The interactions are modeled using the principles of the social force model [193] which avoids collisions of objects using force vectors. Instead of applying the force vectors to the motion of the persons, the weights of the particles in the SMC implementation are adapted, i. e., the weight of invalid predicted particles is decreased while the weight of valid particles is unchanged. The SMC implementation further allows for the modeling of state-dependent detection probabilities which is essential in the investigated scenarios due to occlusions by static or dynamic obstacles.

Popular approximations of the multi-object Bayes filter are the Probability Hypothesis Density (PHD) filter [162], the Cardinalized PHD (CPHD) Filter [129], and the Cardinality Balanced Multi-Object Multi-Bernoulli (CB-MB) filter [114]. However, these approximations require the removal of the set representation and do not facilitate the modeling of object interactions. Recently, the class of labeled RFS and an implementation of the multi-object Bayes filter using Generalized Labeled-Multi-Bernoulli (GLMB) distributions were proposed by Vo and Vo (2013) [74]. Since the GLMB representation keeps the set representation, the proposed methods for modeling object interactions could also be integrated into the GLMB filter.

### C Evaluation

Since the scenario of setting up a home theater provides several interaction possibilities, the information about the user's position is essential for choosing the best output device and modality for the current time step (cf. Section 5). The choice of the sensor setup for detecting and tracking all persons or other obstacles in the proximity of a system strongly depends on the conditions of the scenario, e. g., the required range or resolution. In our example, two laser range finders are used for the tracking system. The sensors are mounted at a height of approximately 1 m in two corners of the room (cf. Figure 5)). Obviously, this sensor setup leads to a high possibility of occlusions in scenarios with several persons in the room, which emulates the restricted mounting positions for arbitrary types of sensors in real systems.

The SMC implementation of the multi-object Bayes filter, as presented in Section B, successfully estimates the dynamic states of the persons in challenging scenarios with short-term occlusions due to other persons or static obstacles. In order to keep track of a person during occlusion, a state-dependent detection probability is required, which may, e. g., be obtained using a grid map [97]. The benefit of using a state-dependent detection probability is the significantly reduced number of lost tracks in occlusion scenarios [63]. Further, modeling object interactions stabilizes the tracking results and facilitates principled approximations during the calculation of the multi-object likelihood function [63]. The filter delivers highly accurate estimates with position errors of less than 10 cm. In case of short-term occlusions, the position uncertainty obviously increases, but the sophisticated modeling of the detection probability ensures that the uncertainty about the user's position is restricted to the occluded area. Since the complexity of the filter increases exponentially in the number of tracked objects, real-time performance is only possible for a restricted number of objects. Hence, approximative solutions like the Labeled Multi-Bernoulli (LMB) filter [64] are required for scenarios with more than 10 to 15 persons. However, the structure of the LMB filter enables the usage of the SMC multi-object Bayes filter for small groups of objects which are currently in a challenging situation.

As discussed above, the SMC multi-object Bayes filter provides excellent results in situations with short-time occlusions. However, the absence of context knowledge prevents the continuous tracking of persons in case of long-term occlusions. Hence, the results of the multi-object tracking system can be improved in these scenarios by a probabilistic knowledge base. The feasibility and benefit of the integration between a probabilistic knowledge base and a multi-object tracking filter is demonstrated by Geier et al. (2012, 2012) [81, 82] via an empirical evaluation. The results of the probabilistic object tracking approach are integrated into a Markov logic (ML)-based model providing information about people's destinations. The aim of the experiment was to improve the tracking performance by enriching the model with additional background information. For evaluation purposes, three increasingly sophisticated ML models have been employed. The first ML model only encodes a floor plan of the room and a basic movement model, the second one additionally adds information about static occlusions and the third one also uses the information about the possible destination of the current user. The results showed that the track continuity significantly increased with the complexity of the ML model. Since an ML model can only represent discrete-valued random variables, the floor plan has to be discretized with a specific resolution. The experiments by Geier et al. (2012) [82] showed that the discretization has to be chosen with care, because both overly coarse and overly fine-grained discretizations can be unfavorable.

## V Planning

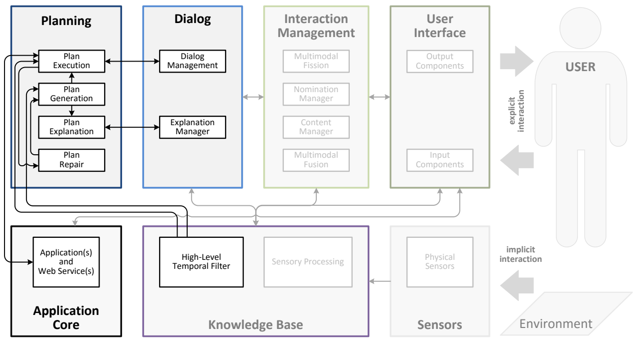


Figure 6: Architectural view and modules with focus on the planning part.

Advanced user assistance is based, among others, on AI planning capabilities. The most fundamental of these capabilities is to come up with a plan, i. e., a course of actions, that solves the user's goals. Such goals can be specified both in terms of state properties that must hold after the execution of a plan and in terms of abstract tasks that need to be achieved. That is, the plan generation component is provided with some abstract tasks that specify the high-level activities the user wants to have achieved, such as setting up the home theater in our assembly task example. Our planning framework includes the generation of plans, plan execution, plan repair, and plan explanation (see Figure 6).

The interplay of the plan generation component with other components of the architecture (cf. Figures 3 and 6) works as follows: The input of the plan generation component is a planning problem (which encodes the initial state, the user's goals, and the planning model). There are two possible types of planning problems that come from different sources.

The first type is the initial planning problem, which, in the prototype, is specified beforehand and stored in the knowledge base. In general, it should be elicited from the user, e. g., by letting her or him select a predefined domain of interest and initiating a domain- and user-specific dialog for specifying the problem instance. Such problems will, in our architecture, be generated by the knowledge base and passed on to the planning component. The second type of planning problem occurs when the system needs to repair the current plan. In that case, a new planning problem is generated by the plan repair component and passed on to the plan generation component (Section C).

The plan generation component creates a solution for the given planning problem. The plan is passed on to two components. On the one hand, the plan execution component (Section B) needs it for execution; on the other hand, the plan explanation component (Section D) receives it to be able to answer questions about it that may arise on the user's side.

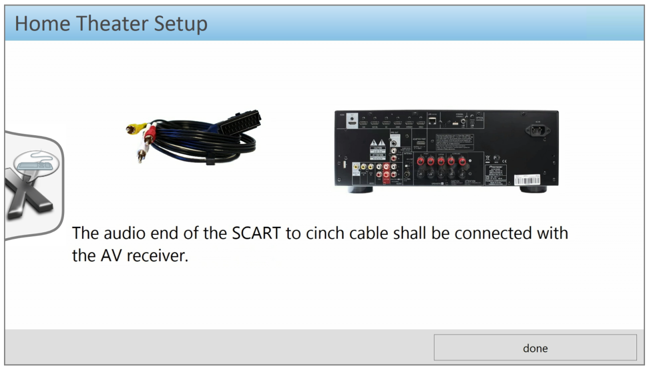
There are mainly two purposes of a solution: It may determine the behavior of the system itself (enabling flexible system behavior) and thus be executed by the system. In that case, the plan execution component passes the steps that need to be done to reach the goal on to the application core component. A second purpose is to serve as advice for a human user. Then the steps are passed on to the dialog component for execution. In any case, the plan execution monitors the execution as described in Section B. When problems occur during execution, it triggers the plan repair component to generate a plan repair problem.

When planning-based advice is provided to the user, questions might arise on the user's side. Questions might be about *declarative knowledge* (like "What is a SCART cable?") – these questions are answered by the dialog component (cf. Section B). There might also be questions about *procedural knowledge* (like "Why shall I do that?"). If this is the case, then the dialog component calls the plan explanation component (Section D) and presents the answer to the user in form of natural language text.

**Preliminaries** The deployed planning framework, *Hybrid Planning* [186, 171, 89, 33], fuses *Hierarchical Task Network (HTN) planning* [191, 156] with concepts from *Partial-Order Causal-Link (POCL) planning* [201, 200, 157]. Both paradigms seem to be well suited for our endeavor to assist human users in real-world tasks for several reasons. Most notably, HTN and POCL planning correspond to two prevalent methods of human planning behavior: top-down and bottom-up planning, respectively [153][2]. In top-down planning, higher-level actions "guide" the decision making on the more primitive and elemental actions. This is achieved by generating a higher-level plan first, and then (e. g., using predefined "recipes") refine this abstract solution into a concrete one. HTN planning uses the same hierarchical structure of the domain to define valid problem-solving strategies, and is as such well suited to capture this behavior. In contrast, in bottom-up planning humans make decisions based on currently available options and opportunities, rather than based on predefined refinement structures. Such opportunistic behavior can be captured by POCL planning, which is driven by currently unfulfilled preconditions of actions. As argued by Ward and Morris (2005) [153], humans do not seem to use one of these ways to plan alone, but rather combine them into a coherent planning behavior, where the choice of the planning methodology at each decision step depends on external influences (e. g., difficulty of the problem, familiarity of the human with that problem, size of the problem, etc). Hybrid Planning does the same as humans do: combining two approaches to best fit the human's cognitive process of planning. Using a planning procedure that also refines abstract tasks into more primitive ones thus resembles that way of problem solving and can further be exploited by incorporating a human user directly in the planning process [51, 31, 11, 12]. Even before the actual planning starts, the hierarchy can be exploited, as it allows to model the given application in a hierarchical manner (e. g., rules as "In order to carry out task X, one must first do Y, then Z" can be expressed). Several real-world planning applications have therefore been modeled in that way [149, 118, 89, 44]; in particular, many planning-based assistance systems are based upon a hierarchical planning approach [178, 163, 145, 137, 76, 22], see Section B. The hierarchical dependencies can further be exploited when generating plan explanations, as elaborated in Section D. Plan explanations further base on the explicit representation of causality (by means on causal links, which are inherited from POCL planning). They can further be exploited for plan linearization and plan repair (see Sections B and C, respectively).

We now shorty explain the hybrid planning concepts that are relevant for the purpose of this report. For a more detailed explanation we refer to previous work and, in particular for our enterprise of applying the technology to human users, to the book chapter on *user-centered planning* [15]. Planning is centered around the execution of tasks. In hybrid planning, there are two kinds of tasks. Abstract tasks and primitive ones; the latter are also referred to as actions. Actions are regarded directly executable by the user, so they can be communicated to him or her in an adequate way by relying on the dialog and interaction components (Sections VI, VII, and VIII, respectively). Abstract tasks, on the other hand, are high-level descriptions of several tasks, so they are not directly communicated to a user but refined into courses of actions first. For this purpose, the so-called planning domain model contains one to several decomposition methods for each of the abstract tasks. Such a method specifies in which way an abstract task can be refined. That is, each method is a mapping from an abstract

[2]There are other approaches that try to explain human planning behavior. Notably Newell, Simon, et al. (1972) [204] interpreted it as a search process through a space of states – which resembles forward search in classical planning. Empirically this behavior seems to be limited to well-defined and artificial planning problems, like Towers of Hanoi [147].


Home Theater Setup

The audio end of the SCART to cinch cable shall be connected with the AV receiver.

Figure 7: This figure shows how an action may be presented to the user [45, Fig. 1]. The $X$ on the left side allows the user to provide user-initiative input via text input, as for example: "The cable is broken."

task to a pre-defined partial plan consisting of more primitive tasks that is regarded an implementation of that abstract task [171, 33]. Tasks, both primitive and abstract, specify preconditions and effects, which in turn define the circumstances under which the tasks can be executed and how they change the current state of the world. For instance, the action $plugIn(\textsc{scartCinchCable}, \text{PORT}_{audio}, \textsc{amp}, \text{PORT}_{audio})$ represents the instruction to plug the audio end of the SCART to cinch cable into the respective audio port of the amplifier. The actions' parameters, such as $\textsc{scartCinchCable}$, are constants representing the actual devices. These actions are then passed on to the dialog manager, which decides how they are communicated to the user (cf. Section VI). How this can look like is illustrated in Figure 7. It is worth mentioning that the level of abstraction of these actions is closely coupled with the level of abstraction of the respective dialog structure behind: As will be explained in Section VI, each action has a hierarchical dialog structure associated to be able to present the respective action in a user-specific way. The more elementary an action is modeled, the less hierarchical can this dialog structure be. So, the level of abstraction needs to be considered with care. A further alternative would be to compute plans in which some of the plan steps are still abstract. In this case the downward solution property [202, 197] should hold so it is guaranteed that these partially abstract plans can be refined into solutions.

As mentioned before, the action's preconditions and effects specify when they are applicable and how the states in which they are applied change. More precisely, they are conjunctions of literals, which in turn base on a many-sorted first-order logic [171], in which the relevant information about the application domain is modeled. For instance, the amplifier is represented by the constant $\textsc{amp}$ of sort DEVICE. Each device has several ports, such as HDMI or coaxial. These are represented by constants as well. Their properties, such as *signal in*/*signal out*, *video*/*audio*/*both*, *used*/*unused*, and *male*/*female* are expressed via relations between these constants. For example, the predicate $used(\textsc{amp}, \text{PORT}_{audio})$ expresses that the audio port of the amplifier is already occupied. The so-called *initial state* then specifies all of these facts which are true prior to the execution of any action. For a more detailed description, we refer to previous work [55].

The actual problem that the user wants to get assisted by is specified in terms of a *hybrid planning problem* [89, 33]. Such a problem description consists of the *planning domain* (the description of all ingredients that were explained before: abstract tasks and their decomposition methods as well as primitive tasks) and by the actual problem description, i. e., an initial partial plan that contains some initial abstract tasks as well as an encoding of the initial state and the user's goals. Please note that we do not employ a user model for *plan generation*. So far, we exploit such a model only for the components that *directly* involve the human user, i. e., the dialog management (see Section VI) and the interaction management (see Section VII). This is in contrast to very recent work in which such models are used for planning as well [29, 10]. Here, in addition to the planning problem and domain itself, an additional model is maintained that specifies the user's mental model of that task. That way, plans can be generated that are closer to the expectations of the human user. This form of plan generation is often referred to as *human-aware planning* [6] or *human-in-the-loop planning* [50].

Partial plans are partially ordered sets of tasks. Due to the partial order, tasks need to be identified in a unique way. So, the partial order is defined upon a set of so-called plan steps, where each plan step is simply a uniquely labeled task. The causal dependencies between the plan steps is explicitly represented by relying on so-called *causal links*. A causal link $ps \rightarrow_{\varphi} ps'$ denotes that the precondition literal $\varphi$ is provided by (an effect of) the plan step $ps$. Causal links are thus the basic means in hybrid planning to ensure executability of plans. A partial plan is called a solution (or plan) to a hybrid planning problem if it is executable and if it is obtained from the initial partial plan by decomposition of abstract tasks and by the insertion of tasks, causal links, ordering constraints, and variable constraints. Executability is defined in terms of the causal links as ordinarily done in POCL planning. For a more detailed description of the solution criteria we refer to previous work [89, 33].

### *A Plan Generation*

Hybrid planning is in general quite difficult: Even the verification whether a given plan is a solution is often NP-complete [43, 33]. The question whether there exists a solution to a hybrid planning problem (the so-called plan existence problem) is in general undecidable [33]. However, in many cases this worst case complexity is too pessimistic and so the plan existence problem can be decided more efficiently [191, 91, 54, 41, 42].

There is a wide variety of planning systems capable of solving hierarchical planning problems. Some of the best known ones are SHOP2 [164] and our recent improvement thereof to allow the incorporation of heuristics [8], the ones for angelic hierarchical planning [119], GoDel [73], the plan space-based PANDA [113, 56], and an approach that relies on an encoding to SAT [3, 4, 187] or to non-hierarchical planning [30]. For an overview of the most early (and influential) hierarchical planning systems we refer to the work by Georgievski and Aiello (2015) [47] and for more recent and details overviews to the related work section of our most recent works on heuristics in HTN planning [14, 8]. In our prototype system we employ search using the plan space-based planning system PANDA [56].

PANDA searches in the space of partial plans starting with the initial partial plan $P_{init}$. That partial plan is successively refined until a solution is generated. That is, PANDA's search procedure [56, Algorithm 1] mimics the solution criteria of hybrid planning by decomposing abstract tasks and by inserting tasks, causal links, ordering and variable constraints. Te search process can be guided by informed search strategies and heuristics, which are designed to incorporate the task hierarchy [80, 56, 14], as well as causal relationships in the absence of hierarchical tasks, i. e., in case of standard POCL problems [175, 166, 67].

The runtime of the system heavily depends on several factors, such as the size of the problem, its combinatorial complexity (which in turn depends on how the problem is modeled), and the deployed search strategy or algorithm and heuristics. For the example application that we illustrate here, we can report that runtime was never an issue as plans can be found within at most seconds.

### *B Plan Execution and Monitoring*

To enable flexible plan execution, solutions generated by the plan generation component are partially ordered. When a human user is involved, in particular when he or she is executing the plan, this is often done sequentially. Thus, a linearization has to be found that is suitable for a human user [58]. This task, to linearize a partially ordered plan in a user-friendly way, is the first objective of the plan execution component. The second task is to monitor whether the outcome of the execution of an action matches the outcome given in the planning model and to trigger the plan repair component if necessary.

**Plan Linearization** Though all possible linearizations of the plan's partial order are valid (i. e., their execution will realize the goal) there may be orderings that are more intuitive for a human user than others. Consider the assembly problem in Figure 8, a receiver and a television have to be connected using a SCART to cinch cable. To solve it, the three cinch plugs have to be connected to the television and the SCART plug has to be connected to the receiver. The ordering of this four plug actions has no effect on the success of the plan, but it might be confusing if the system's linearization starts with the video cinch cable, followed by the SCART to cinch cable and ending with the two audio cinch cables – or even worse: if it starts to establish some completely different connections in between.

Figure 8: The Figure shows the devices and their ports that are involved in the task of connecting a receiver and a television via a SCART to cinch cable.

Thus, in interactive systems, techniques should be applied to find orderings that are appropriate for human-computer interaction. This approach can be seen as using a two-part model: hard constraints that are necessary to reach a goal are given in the planning domain, while soft constraints on the ordering are established afterwards. It preserves the flexibility of the plan, but adapts the ordering as much as possible to the needs of a user. Though it might be necessary to adapt the ordering strategies to a specific domain of application, there are several ways to re-use knowledge that is present in the planning problem or generated during the planning process to realize domain-independent approaches.

We identified three sources of information that form the basis for the following domain-independent linearization strategies [58]. As future work, we want to empirically evaluate if they make plan execution more intuitive for human users.

1. *Parameter similarity* – In the home theater domain, it seems reasonable to complete steps that include similar devices sequentially. Since these devices are represented as constants in the planning problem, they form the parameter set of the plan steps. A linearization component may optimize the similarity between parameter sets of plan steps and may search for a linearization that maximizes the similarity of subsequent steps. In the example given

above, this may result in a linearization where all television-related steps are close to each other.

2. *Causal link structure* – Causal links represent the causal structure in the plan and indicate which plan step is necessary to fulfill a precondition of another one. Since the planning process is problem-driven, there is no causal link in the plan that is not necessary to support a precondition. The second strategy searches a linearization where the producer and the consumer of a causal link are as closely presented to each other as possible. If a strategy suits a given problem or not heavily depends on the way a domain is modeled, of course. The requirement that the signal reaches its sink might, e. g., be modeled via preconditions and effects; in this case, this strategy could work well. However, it might also be fully modeled via the hierarchy, so that the next strategy might select better linearizations.

3. *Decomposition structure* – This strategy relies on the assumption that plan steps decomposed by the same method semantically belong together. Generalizing this assumption, the strategy is based on the distance of two plan steps in the decomposition tree, i. e., the tree of decompositions that transfer the initial plan into the applicable solution. Steps that are closer in the decomposition tree should also be ordered closer in the linearization. If you consider the example given above, the plugin task of the overall cable might be decomposed into two tasks – one for plugging in each of its two ends – and the one of it belonging to the cinch end might further be decomposed into two tasks – one for plugging in the audio end and one for the video end.

The given strategies can be applied directly, but they can also be the basis for domain-specific linearization approaches. A greedy implementation chooses the *next step* from a set of possible next plan steps, (i. e., those steps where all predecessors have already been executed), or the given criteria can be optimized over the *whole linearization*.

It is worth noting that research on the problem of finding good plan linearizations was first inspired by working on our prototype system: In early versions, the next step to execute was chosen arbitrarily. The results were often quite confusing. In our prototypical system, we hence implemented a strategy that corresponds somehow to the parameter similarity: In the given application scenario, constants and thus parameters represent devices of the real world – it thus seems reasonable to finish steps regarding one specific device before starting to work on the next one. We then generalized this idea and developed the further ones we have introduced here [58]. To our knowledge, there is no other approach for optimizing linearizations for human understandability in the literature so far.

**Monitoring** Once a plan step has been selected to be executed, it is passed on to the dialog management component for execution. After the execution has been finished, the dialog management component passes its result information on to the knowledge base component and triggers the plan execution component to check if the intended results have been achieved. This is done by querying the current state from the knowledge base component and comparing it to the expected state, i. e., the state that results from applying the plan step to the last known state. If these states are identical, the plan execution component is free to start the next plan step.

If the current state differs from the intended state, the plan execution component has to decide whether the detected difference is crucial for the execution of the overall plan or not. This is done based on the set of active causal links. A causal link is active if its producer has already been executed while its consumer has not. If there is no active causal link on some literal, it is either not important for the remaining plan steps to succeed, or it will be (re-)established by another plan step before it is needed. Therefore, if there have not been unexpected outcomes on literals with active causal links, the plan execution is, again, safe to start the next plan step. Otherwise, it is not ensured that all preconditions of the remaining plan steps are fulfilled and the plan repair component is triggered to find an adapted plan (cf. Section C). Once this plan has been generated, plan execution is resumed.

Another capability that is closely related to monitoring the execution of plans is *plan and goal recognition*: Given an observation of the user's actions performed so far, plan and goal recognition can be used to infer both the overall plan (i. e., outstanding actions) that the user is currently pursuing, as well as the goals and tasks the user wants to achieve by it. We have done some theoretical investigations on the complexity of this task [43, 37]. Very recently we have developed the first approach for plan recognition in a hierarchical setting [7], but it is not integrated into our prototype system. Plan and goal recognition might help to decide when the system should offer help to the user, e. g., when he or she changes his or her goals or when execution of the plan is not continued. It might also enable more flexible execution monitoring, since the user might try to reach a goal on another way than suggested by the system. That way, the system might even learn about preferences of the user.

## C Plan Repair

There are several reasons for a plan to fail. The environment may be complex and its representation in the planning domain may abstract from many details. Thus, changes in the environment or the outcome of some action may be unforeseen by the planning system. There may also be parts of the environment's state that are not observable for the system.

Though the performance of deterministic planning systems makes their use in real-world environments promising, they have to be able to deal with the given reasons for a plan to fail. Therefore, the plan execution component keeps track of the environment, detects unexpected changes and judges if they threaten the execution of the currently executed plan.

There are two main approaches to deal with unexpected changes in the environment [15]. First, re-planning is an approach that discards the old plan and searches for a new plan from the current state. A drawback of re-planning is the lack of plan stability when users are involved in the execution process: an entirely new way of solving the problem at hand may confuse the (human) executor. Also, resources may have already been blocked and appointments been made to execute the original plan. The user might not accept to take a taxi after having bought a bus ticket. The second option is to adapt the existing plan to fit the new situation. This might be possible with only minor changes, enabling plan stability. However, as Nebel and Koehler (1995) [194] showed for the setting of classical planning, modifying a plan at hand may in the worst case be harder than planning from scratch due to the additional complexity of trying to reuse as many steps from the original plan as possible [194].

Work on plan repair in a special setting of totally ordered HTN planning (here, the initial plan as well as all plans in the decomposition methods are totally ordered) was done by Warfield et al. (2007) [133] and Ayan et al. (2007) [124]. However, both approaches do not take the following aspect into account: in HTN planning, a plan resulting from some reparation process (either re-planning or repair) is not necessarily a refinement of the initial plan, which is one of the solution criteria for both HTN planning [191, 91] and hybrid planning [89, 33]. Consider the case of re-planning: the new overall plan consists of a prefix that is the executed sequence of the original plan and of a postfix that is the solution to the newly generated plan. This combination may not fulfill the constraints to a valid solution that have been represented in the hierarchy of the problem.

Therefore, we use a special plan repair approach that meets the criteria of hierarchical planning [116, 89, 55]. A full description of the approach is given in Section 4 by Bercher et al. (**.**) [**.**] In a more recent publication we discuss the differences and similarities of plan repair and replanning in a hierarchical problem setting in more detail and give further pointers to related work [9]. The approach guarantees the repaired plan to be a refinement of the original initial partial plan and includes all actions that have already been executed. This offers basic stability, although it does not guarantee minimal modifications to the old plan. It does however enable the use of sophisticated planning techniques and leaves the system free to find a solution that deals with the reason why the original plan failed. From a computational point of view, we found it to be similar to planning from scratch, though there are also cases where it becomes harder or easier for our planning system.

Based on the planning domain, the initial problem instance, and the failed plan, a new *plan repair problem* is created. It includes an additional set of so-called *obligations*. These are elements of the original plan that have to be present in the repair plan. It contains plan steps that have already been executed. These steps might have been partially ordered or even unordered in the original solution, but have been executed in a distinct total order. So this ordering must also be enforced in the repair plan using ordering obligations. Do deal with the unexpected environment change that caused the repair, a new primitive action is defined that is called *process* and causes an effect that reflects the detected change. It is also included as an obligation into the problem and placed after the finished actions. The planning algorithm is slightly adapted to find solutions that fulfill the obligations [55].

If the planning system generates a solution to the plan repair problem, it is a decomposition from the initial partial plan. It includes the parts of the original plan that have already been executed and can deal with the cause for re-planning. There might be situations where the new problem is not solvable, in which case the system informs the user accordingly.

Integration of the plan repair component is done by letting the plan execution component trigger repair when it detects an unforeseen state change that may threaten the execution of the given plan. The repair mechanism needs as input the original planning problem, the sequence of plan steps that have already been executed, and the unforeseen state change. This enables the component to create a plan repair problem that is then passed on to the plan generation component to find a solution.

## D Plan Explanation

The previously described plan repair process may lead to a plan which is different from the plan the *Companion*-System has initially presented to the user. These repaired plans are usually more complex in terms of size and causal structure, too, and may thus be difficult for a human user to comprehend. A similar situation can arise if the user is asked to execute a plan step while not understanding why he or she should do it, i. e., why performing the action helps him or her to achieve his or her objective. In both cases the user does not understand the behavior of the system. This problem needs to be properly addressed in a *Companion*-System, since unexpected or non-understandable behavior has a negative impact on the user's trust in a human-computer relationship [198]. Studies have shown that if the user does not trust the system, the interaction with the system itself may suffer [189]. This includes a reduced frequency of interactions, less engaged interactions, and in the worst case the complete abort of future interaction.

A *Companion*-System accommodates for these needs by providing suitable plan explanations. Previous research of Lim et al. (2009) [110] demonstrated that different kinds of explanations in context-aware intelligent systems like *Companion*-Systems are suitable to improve the trust of the user in the system and his or her understanding thereof. For planning-based intelligent systems in particular, explanations can improve the user's trust in the generated solution [32, 38]. As a general point Fox et al. (2017) [20] also argue that planning systems should be able to explain their behavior and their plans. Chakraborti et al. (2017, 2018, 2018) [29, 10, 6] argue that a model of the user's mental model of the planning task should be maintained so that generated plans and explanations are more expectation-conforming. For domains in which there are large differences between a planner's model and the human's mental model of it, they propose a plan explanation approach based on "model reconciliation": The assumption is that certain solutions presented to a human might not be comprehensible by him or her because of a different underlying model (e. g., the user assumes different preconditions or effects for certain actions). The approach then identifies and explains these differences to the user [19].

The question arises which types of explanations should be used by a *Companion*-System. Lim et al. (2009) [110] determined that *Why* and *Why-not* explanations can be used to increase the user's comprehension of the system, while only *Why* explanations increase the trust in human computer interaction. Similar correlations have also been observed by Nothdurft et al. (2014) [62]. Consequently, the approach for plan explanations used in our *Companion*-System focuses on *Why* explanations. Such explanations describe, e. g., why a certain plan step is part of the plan the users executes or why a certain ordering between two plan steps must be observed. In addition to these plan explanations, our system can also (even pro-actively) provide the explanation about certain concepts (e. g., about an HDMI cable) to establish a common ground between the parties. These explanations will be presented in Section B.

For the sake of brevity, we will only provide a short description of the algorithm which we applied in our *Companion*-System to generate explanations for the presence of a plan step in a plan, albeit our plan explanation system also being capable of providing explanations for orderings and the choice of action parameters. A complete description of the employed explanation technique was given by Seegebarth et al. (2012) [87]. Explainable planning has attracted more attention in recent past Fox et al. (2017) [20]. Still, there are only a few approaches in the literature in that general area. One approach focuses on explaining why a given classical planning problem is *not solvable* by providing an alternative initial state in which it would be possible [100]. We do not consider this approach as it cannot provide the *Why* explanations sought here. As mentioned above, there is a recent approach concerned with *Why* explanations that requires a formal specification of the user's mental model of the planning task. Given a solution, the involved differences are presented, i. e., explained, to the user [19]. Some work also considers *plans* as explanations of observed behavior [98], similar to the task of plan recognition. In previous work we have developed a technique for generating *Why* explanations [87] and evaluated their impact and usefulness empirically [55] – this is the approach pursued in our system; we will outline it below. In addition to such pure plan explanations, we have also developed a technique to combine them with explanations for background knowledge stored in an ontology [44, 13]. Such extended explanations increase the system's transparency as they not only explain the plan itself, but also the restrictions placed by the planning domain onto it. In the future, we want to develop a technique to generate *Why-not* explanations. Further arising challenges are outlined in more detail in recent work [11]. Using such why-not explanations, one could, e. g., explain why it is not possible to include a specific action in a plan; this will become especially interesting for mixed-initiative systems, where the user can actively participate in the process of generating plans and ask for specific changes.

The most paramount objective when generating explanations is that they are coherent with the current plan and their arguments are sound. To ensure this, we use a technique based on formal proofs in first order logic where an explanation is a proof for a certain logical fact [87]. Based on a plan, the list of modifications applied to generate it and the plan step, whose necessity has to be explained, a set of first order axioms is constructed. We use a predicate $N(ps)$ to denote that the task $ps$ is necessary in a given plan. Please note that the notion of necessity we employ does not imply that the execution of the plan cannot be successful without it, but rather that it serves some purpose in the respective plan. An explanation is consequently a formal proof for the necessity of the plan step the user has inquired about and can be generated by an arbitrary theorem prover. The structure of the axioms ensures that the proof is always a linear list of derivations, which are the arguments of the generated explanation, and that proving the theorem will always terminate in a finite number of steps. In practice all possible explanations (often several thousands) can be generated within milliseconds, due to the fact that every axiom is a Horn Clause [87].

The generated axioms can be split into two groups. First, the plan-dependent axioms encode the causal and decompositional structure of the plan, the initial partial plan and the goal. For example, a causal link is encoded by the fact $CR(ps, \varphi, ps')$. Similarly the plan steps contained in the initial partial plan and the plan step representing the goal are defined as necessary. This is due to the solution criteria for hybrid planning, which require that the plan steps in the initial partial plan must be decomposed into primitive actions in the solution and after executing all actions a goal state must be reached [89, 33]. Second, the plan-independent axioms encode the inference rules used to determine why a task is necessary. Such an inference can, e. g., be based on the causal links contained in the plan. Suppose there is a causal link $ps \rightarrow_\varphi ps'$ between the plan steps $ps$ and $ps'$, which ensures that the precondition $\varphi$ of $ps'$ is fulfilled. Then the plan step $ps$ is necessary, provided that $ps'$ is also necessary, since without it $ps'$ could not be executed. This relationship can be encoded in the axiom given in Equation 4.

$$\forall ps, ps', \varphi . [[CR(ps, \varphi, ps') \wedge N(ps')] \Rightarrow N(ps)] \tag{4}$$

Similar rules can be created based on the decomposition methods that have been applied to generate the plan and others based on the interaction between causal links and decompositions.

Figure 9 depicts the graphical representation of an example for a formal plan explanation that can be generated by our approach.

As a last step, the formal explanation needs to be transformed so that it can be conveyed to the user. In our architecture, the result of this transformation is a natural language text describing the formal explanations. The interaction management component subsequently determines the modality in which the explanation will be presented to the user, e. g., whether it will be displayed as text or read by a text-to-speech component. This process is transparent to the plan explanation component. The system described in Section II uses a pattern-based approach to generate natural language

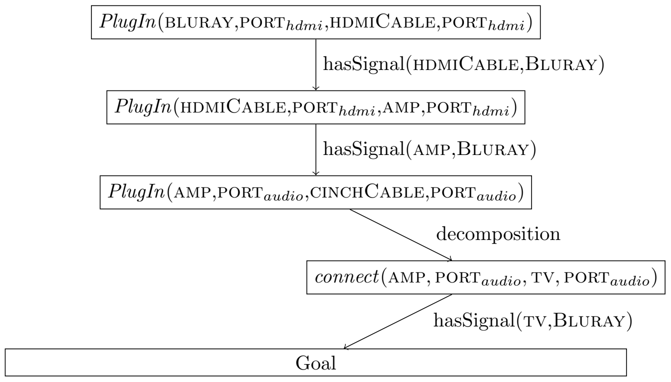

PlugIn(BLURAY,PORT_hdmi,HDMICABLE,PORT_hdmi)
hasSignal(HDMICABLE,BLURAY)
PlugIn(HDMICABLE,PORT_hdmi,AMP,PORT_hdmi)
hasSignal(AMP,BLURAY)
PlugIn(AMP,PORT_audio,CINCHCABLE,PORT_audio)
decomposition
connect(AMP, PORT_audio, TV, PORT_audio)
hasSignal(TV,BLURAY)
Goal

Figure 9: An example depicting a formal plan explanation, explaining why it is necessary to plug an HDMI cable in the respective port of the Blu-ray player. Actions occurring in the plan are depicted by boxes, while the arrows connecting them display the arguments of the explanation. Here, the given action is necessary, as it provides the effect that the HDMI carries the signal from the Blu-ray. Likewise, the third action is necessary, as it is part of the decomposition of the *connect* action.

given the formal proof. This technique is commonly used to present proofs generated by automated theorem provers [190, 183, 172]. One could also use a system similar to the Interactive Derivation Viewer [132] to make both verbal and visual explanations available.

Concerning the integration of plan explanations in our *Companion* architecture, plan explanations are only used if a user has explicitly requested them. Recognizing such a request is the objective of the explanation manager, which also determines the plan step the user wants to have explained. Using the middleware this information is transferred to the plan explanation component, which subsequently computes a plan explanation. Since there is a multitude of valid formal explanations of which only a single one can be presented to the user, the system has to select a most suitable one. Currently, we employ the “shorter explanations are better” scheme, as we assume that shorter explanations are easier to comprehend. If two (or more) explanations have the same length one of them is selected randomly. The generated and selected plan explanation is sent back to the explanation manager, which takes care of verbalizing them.

We have conducted a thorough investigation of the usefulness of generated plan explanations in our usecase system (see Section E). However, we are also aware of a few peculiarities which are to be investigated in future research. To us, the selection of a single explanation out of all possible ones for presentation to the user seems most crucial [11]. Despite the logical soundness of all presented explanations, some might be more comprehensible to a user. Imagine a situation where both the audio and the video signal are transmitted trough several cables, i. e., plug-in actions. Here, a possible explanation would state that the audio signal is necessary for action 1, which produces the video signal for action 2, which in turn produces the audio signal for action 3. Such an explanation might be confusing for a human user, compared to an explanation solely talking about audio signals and should thus not be presented. In the future, we want to investigate methods to select suitable explanations, e. g., based on constants contained in causal links or action parameters, the alternation of causal and decompositional arguments, or domain-specific knowledge. Similar problems arise also if linearizations have to be determined (see Section B), and thus similar techniques could be used. Formal explanations, especially in long plans, can be quite complicated. As such, techniques to produce shorter and more abstract explanations should be developed in order to ensure easy understandability. A similar issue arises from explaining certain relationships (like entailments) in ontologies. The before-mentioned integration of plan explanations with explanations based on background knowledge stemming from an ontology [44, 13] will also be further pursued. These techniques should take the user's knowledge into account to make sure that he or she will be presented an explanation that is suited for his or her level of expertise in the given domain.

### *E Evaluation*

There are many components of the planning framework that can be evaluated. The most fundamental one is the plan generation. In the given setting, runtimes of the plan generation component (i. e., the planning system PANDA) was never an issue. So, PANDA was only evaluated systematically in a variety of standard benchmarks using different search strategies and heuristics [113, 80, 56, 14]. Concerning the example scenario of setting up a home theater, we did an empirical evaluation that should show how well such a system is perceived in general and, more specifically, what impact plan explanations have on the users' confidence in the presented solutions [55].

The task of the test subjects was to set up a complex home theater as described in Section II. The test subjects did not have to figure out a solution to that task on their own, however. Instead, they were presented a detailed sequence of instructions that tells them which cable is to be plugged into which specific port of which device. These instructions are based upon a solution to the planning problem that encodes the respective assembly task. The presented instructions show these facts both using animated pictures (where the ports are flashing) and natural language text. Figure 7 depicts such an example instruction. The test subjects only had to follow these instructions – afterwards, the task was successfully completed (assuming they correctly followed the instructions). The system that showed the subjects how to perform the task was a seemingly interactive HTML5 slide show that corresponded to fixed course from the prototype system. We did so to have full control of the experiment and to ensure reproducibility. Here, we do not give any further details on how the experiment was performed and refer to our previous work for that purpose [55]. Instead, we just want to mention our main findings.

Our main hypothesis was that plan explanations foster the users' confidence in the correctness of the presented solution. Our data did not reveal any statistically significant effect related to the plan explanations. However, we believe this was mainly due to two reasons. First, the experimental setting seemed to be suboptimal, as there was no intrinsic need for the subjects to actually read and understand the explanations. They were not intended to be optional, but still some subjects proceeded by clicking "okay" very shortly after they were displayed. Second, the overall impression of the system was already very high for all of the subjects thereby limiting the possible impact of showing explanations: We constructed a summary variable summing up all questions that rate various aspects of the system (ignoring explanation aspects). These include trust, patronization, appeal, and utility. According to these results, the system was very well perceived in general with 26.63/3.67 points (mean/sd) out of 30 points. Some critical comments were mentioning the artificial voice reading the instructions. Most free-text comments were positive, however. Many of them highlighted the general appeal of the system. Others referred to its principal assistance functionality by comments like "assists in a useful way", "this assistance system is very useful, as it allows people without expertise to follow the instructions successfully", or "I would prefer this kind of instruction manual to all previous". While we could not support our main hypothesis that plan explanations foster the users' confidence in the correctness of the presented solution, we received mainly positive comments about the explanations such as "explanations were good and useful [...]" [55]. Thus, in conclusion, the evaluation revealed that the system was very well perceived by the subjects – even more by non-experts.

## VI Dialog

The task of the Dialog components is to control the user-adaptive structure, content, and flow of the dialog between the user and the system. It communicates with the Planning components, the Interaction Management, and the Knowledge Base to gather from as well as to provide all necessary information in a user-adaptive

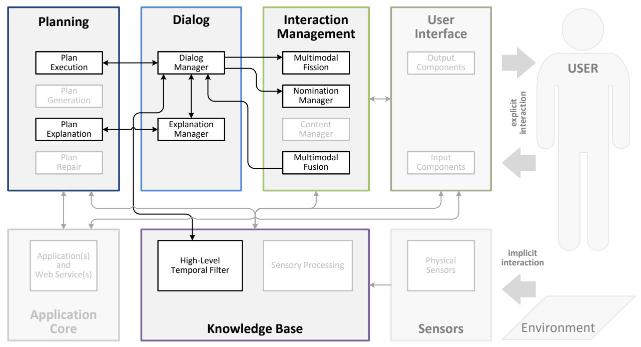


Figure 10: Architectural view and modules with focus on the dialog management.

fashion, which is needed to accomplish a specific task in cooperation with a technical system. In this *Companion*-System the Dialog module comprises the Dialog Manager and the Explanation Manager components [24]. The former is responsible for the user-driven adaptation of the dialog structure (e. g., using scalar knowledge models [173]), while the latter controls and initiates all proactive behavior related to explanation capabilities (e. g., using a structural knowledge model).

At the start the Dialog Manager receives plan steps[3] from the plan execution component to generate a user-adaptive dialog. This dialog is potentially augmented by the explanation manager before it is passed on as modality-independent dialog output to the multimodal fission. However, if the dialog designer wants to specify preferred modalities for a dialog, these preferences can be sent to the nomination manager to influence the modality arbitration process in the interaction management. After the user interaction, the dialog manager receives the fused interaction results from the multimodal fusion and maps these results to action effects, which are passed back to the plan execution component as well as to the High-Level Temporal Filter of the Knowledge Base. In addition, the explanation manager is also responsible for requesting as well as receiving plan explanations from the plan explanation component and then transferring them into text using techniques for natural language generation.

## A *Dialog Manager*

In the past decades several approaches to the dialog management task have been developed, which can be classified in four basic categories. First of all, basic *finite-state-machine approaches*, where a set of states is defined, and, for each state, a set of moves that transition into a new state in the automaton. Second, *frame-based approaches*, where the dialog management monitors the current so-called frame, which is specified by a set of needed information (slots), the context for the utterance interpretation, and the context for the dialog progress. This is more advanced, since it allows for mixed-initiative interaction and allows multiple paths to acquire the information. Third, *stochastic-based approaches* (e. g., by Williams and Young (2007) [134]), which apply reinforcement learning techniques to dialog management by determining the best policy or choice of actions, from all available actions a system can take in a dialogue. Reinforcement learning will optimize the system's performance as measured by a utility function, such as the user's evaluation of the system [169]. The last main group of approaches are *agent-based approaches*, where the dialog is controlled by several intelligent agents capable of reasoning about themselves (e. g., in the BDI (beliefs, desires, and intentions) approach [196]) using artificial intelligence techniques. The agent-based approach subsumes plan-based dialog models, where preconditions, actions, and effects are used to control the ongoing dialogue. Here, the dialog flow is determined at runtime by the different agents using the current world state and the goals left to achieve. Thus, the approach used here falls into this category.

In the architecture, our system bases upon, the dialog management components are coupled closely with the planning components (cf. Section V). Hence, a dialog management approach related to the employed planning approach was chosen. As described in Section B about plan execution, the course of plan steps representing the solution of the given planning problem is sequentially executed, i. e., passed on from the planning framework to the dialog management component. The main purpose of this component is to decompose and refine, if necessary, the plan step into a user-adaptive dialog. If the dialog for the passed plan step requires adaptation to the individual user, the structure and the content of the dialog can be adapted. Therefore, our combination of dialog management and planning is closely related to the split into task level and dialog level in agent-based approaches.

The provided plan step is decomposed into a, if necessary, hierarchical dialog structure which consists of so-called dialog goals [103, 77]. As discussed in Section V, this depends on the level of abstraction of the respective action. For example, if the action is modeled in a very elementary way, then the degree of freedom for presenting it may be limited and only the presentation of the dialogue goal may differ, but a hierarchical structure may not be needed. Each dialog goal represents a single interaction step between the user and the technical system (e. g., one screen on a device filled with specific information). For example, to execute the plan

[3]Please recall that plan steps are simply uniquely labeled actions, so the terms *plan step* and *action* can be considered equivalent here (cf. Section V).

step which checks whether the setup of a home theater was successful, the first dialog goal is to select a video signal to test the system with. A resulting interaction step can, e. g., look as depicted in Figure 14b. Depending on the user's choice further dialog goals need to be established. If the user selects the Blu-ray player as the source of the video signal, he or she has to be instructed to insert a disc, while if testing the satellite receiver is selected, he or she has to be instructed how to do so. The planning component abstracts from this procedure by regarding the test action as primitive and ignoring its internal procedure. Using this scheme, all decisions which do not influence the execution of the plan can be taken by the user via the dialog management.

The term *dialog goal* arises from the fact that every step in the interaction pursues a goal. The goal is, in this case, to achieve one or several of the desired action effects of the respective action. Therefore, the term *dialog goal* is to be distinguished from the term *goal* used in planning. This means that each action can be decomposed into several dialog goals and that for every of its effects a set of corresponding dialog goals may exist. These similar dialog goals usually have so-called *guards*, which formulate conditions that need to be fulfilled in order for the dialog goal to be entered at runtime. These guards take into account user characteristics (e. g., knowledge or emotions), and therefore help to adapt the dialog to the user. These user characteristics, stored in the knowledge base, either come from a default user model that evolves over time using the interaction history (see Section B) or from classification modules that estimate, for example, the user's emotions.

Goals can be arranged in a vertical as well as in a horizontal structure. In a vertical structure each goal may yield several subgoals. In a horizontal structure each goal may have a fixed successive goal that is next in the dialog. This implies that the dialog may be roughly structured like a finite state machine, but there is enough room left to dynamically arrange the subgoals to the user's needs. Such an arrangement is handled in the way the guards for the goals are defined. Typical guards in our model are the requirements for generally needed user knowledge. For example, in our used domain of connecting devices of a home cinema, this means that if the user's overall knowledge regarding this domain is low, he or she will be considered a novice, resulting in an adapted composition and arrangement of subgoals. Those roughly made dialog structure adaptations are later augmented by specific fine-grained user knowledge adaptations during runtime, which will be explained in Section B. Note that the guards mechanism only considers the most probable world state. An interesting extension would be to consider a mechanism capable of more fully leveraging information about world state uncertainty contained in the knowledge base.

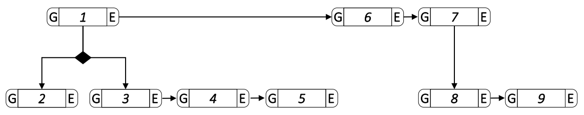


Figure 11: This graphic illustrates the dialog flow [24, Fig. 9.6]. Dialog goals are represented by the rounded rectangles. Inside those, guards and effects are responsible for the dialog flow. Vertical arrows represent levels of abstraction, and diamonds decisions based on constraints. Horizontal arrows represent predefined dialog sequences. This shows that the content and flow may change if, e. g., two different users interact with the system.

During interaction, the dialog management component traverses through its dialog structure to select the path of dialog goals most suitable for the current user. For example, in Figure 11, if a user has expert knowledge on a topic, only dialog step 2 would be included, presenting high-level instructions. Contrary to that, a novice user would receive a more detailed as well as longer sequence of dialog steps (3,4,5) with additional extent of assistance. The selection of the next dialog goal is therefore made in a user-adaptive manner and leads to an individual dialog appropriate for the current user. In order to conduct the selection of the next appropriate dialog goal, a constraint solving algorithm is used. Constraint programming [179] has proven to be especially useful in problems on finite domains where many conditions limit the possible variable value configurations [128]. It is a technique to find solutions to problems by backtracking and efficient reasoning. We consider the conditions in the guards as constraints and based on the current values of the variables the constraint solver tells which guard conditions can be fulfilled. Based on this variable configuration we can select a number of dialog goals that can be executed at a certain time in the dialog. As a dialog in our case is limited to a certain number of possibilities how the system can traverse through the dialog structure, it is reasonable to use a finite domain for the variables that constrain the execution of the dialog goals. In Addition, there are no real-time issues affecting this dialog selection process. This is especially important, since a delay in interaction would influence the user experience negatively. Hence, These conditions make the dialog model suited for applying a finite domain constraint solving mechanism.

Before each dialog step is passed on to the fission component, the explanation manager checks whether the user has the required knowledge to accomplish the proposed dialog step. If necessary, an additional dialog step explaining the missing knowledge is included in the course of the dialog, which will be explained in more detail in Subsection B. The dialogue step is then transformed into a so-called *dialog output*. Each dialog output represents a modality-independent description of an abstract user interface, and can be hierarchically structured using the following elements (cf. Figure 14a): one topic, one dialog act, an optional sequence of navigation items, plus an optional sequence of listen items. The listen items are used to allow user-initiative grammar-based inputs. The dialog act can contain multiple information items, selections, and widgets. Selections consist of multiple information items that act as selection items.

After the user interaction, the dialog management component receives the *interaction results* from the multimodal fusion. The results are analyzed to check whether the results of the dialog step are related to the desired action's effects. If this is the case, these effects are transmitted to the knowledge base as observations.

However, the user could as well search for additional clarification by requesting *plan* or *dialog explanations*. These explanation requests are sent to the explanation manager, which will, in case of a user-initiative dialog explanation request (e. g., "What is HDMICABLE?" or "How can I connect AMP and HDMICABLE?"), handle the request itself. In case of plan explanation requests (e. g., "Why do I have to perform this?"), it will sent the request to the plan explanation component (cf. Section D).

## B *Explanation Manager*

In general, explanations are given to clarify, change, or impart knowledge. Usually the implicit idea consists of aligning the mental models of the participating parties. The mental model is the perceived representation of the real world, or in our case of the technical system and its underlying processes. In this context, explanations try to establish a common ground between the parties in the sense that the technical system tries to convey its actual model to the user. This is the attempt of aligning the user's mental model to the actual system. However, explanations do not always have the goal of aligning mental models, but can be used for other purposes as well. Analogous to human-human interaction, in human-computer interaction the sender of the explanation pursues a certain goal, with respect to the addressee (see Table 1 for a taxonomy of explanation goals after Sørmo et al. (2005) [151]).

*Why* explanations (e. g., plan explanations in Section D) may increase the user's understanding in system decisions, and may thereby counteract arising problems regarding the human-computer trust relationship [110]. However, the most fundamental factors, influencing a successful task completion and healthy human-computer interaction, are still the user's capabilities, and especially the user knowledge, which can be influenced by explanations with the goals of *Learning* and *Conceptualization*. Therefore, the course of the dialog that accompanies the interaction steps is adapted or extended by additional explanation dialogs to increase the individual user's knowledge at runtime. Each of these dialogue steps is represented as a relation with the name of the task and its appendant concepts and arguments. As our main goal is to prevent task failure, we have to ensure that the upcoming or current tasks and appendant concepts will not exceed the user's knowledge.

| Goals | Details |
|---|---|
| Transparency | How was the system's answer reached? |
| Justification | Explain the motives of the answer |
| Relevance | Why is the answer a relevant answer? |
| Conceptualization | Clarify the meaning of concepts |
| Learning | Learn something about the domain |

Table 1: A taxonomy of explanation goals. It subsumes different kinds of explanation as, e. g., why, why-not, what-if, and how-to explanations.

One countermeasure is employing plan explanations as presented in the last section upon explicit user request. For this, formal plan explanations are translated by an *explanation manager* using template-based natural language generation into human-understandable text. Another important countermeasure directly concerns the user's understanding of the task at hand: prior to sending the dialog steps for presentation to the fission component, they are first sent to the explanation manager [62]. Here, the content of the predefined interaction dialogs is analyzed and compared to the user's knowledge model, which is stored in the knowledge base. If the user's knowledge is not sufficient, either the content of the predefined dialog is changed, or the course of the dialog is updated by including additional dialog steps to fit better the user's knowledge model. Those additional dialog steps are meant to explain missing knowledge to the user.

In our knowledge model, we distinguish between declarative knowledge and procedural knowledge [85]. The former can be used to describe the being of things (i. e., appearance and purpose). Possessing declarative knowledge about something does not necessarily mean to be able to use this knowledge for a task or action. In comparison to that, procedural knowledge can be applied to a task. Procedural knowledge provides the knowledge on how to execute a task or on how to solve a problem. For example, in the domain used in the demonstrator, the task represented as relation *connect(*TV, AMP, HDMICABLE*)*, contains the knowledge of the action *connect* as well as the concepts TV,

```xml
<proceduralGoalKnowledge goalName="connect" goalID="3.2">
  <knowledgeValue value="novice" probability="0.5"/>
  <knowledgeValue value="advanced novice" probability="0.3"/>
  <knowledgeValue value="intermediate" probability="0.1"/>
  <knowledgeValue value="advanced intermediate" probability="0.1"/>
  <knowledgeValue value="expert" probability="0"/>
</proceduralGoalKnowledge>
```

```xml
<declarativeConceptKnowledge conceptName="hdmi">
  <knowledgeValue value="novice" probability="0.5"/>
  <knowledgeValue value="advanced novice" probability="0.3"/>
  <knowledgeValue value="intermediate" probability="0.1"/>
  <knowledgeValue value="advanced intermediate" probability="0.1"/>
  <knowledgeValue value="expert" probability="0"/>
</declarativeConceptKnowledge>
```

Figure 12: In this excerpt of the user's knowledge model the procedural knowledge of the action *connect* and the declarative knowledge of the concept *HDMI* as well as their respective probability distributions over the knowledge levels are listed.

amp, and hdmiCable. Those knowledge constituents are modeled as probability distribution over a five-step knowledge scale ranging from *novice* to *expert* (see Figure 12). This means that compared to other systems (e. g., by Beaumont (1998) [185]) the knowledge is modeled in small independent information pieces instead of using only one general level. The user's knowledge levels are based on observations made during the interaction and on past interaction episodes. Therefore, the knowledge levels are system-made assumptions about the user, requiring a probabilistic representation. Events occurring during the human-machine interaction influence the probability distribution of relevant data objects. These events may be, for example, given explanations, failed actions, or plainly elapsed time. Contained is, for example, not only which tasks the user did execute, but which entities were used for this task and whether the task completion was successful.

Before each output, we check whether it is sufficiently likely that the user's knowledge level for the current dialog step is sufficiently high. If it is too likely that the user's knowledge level is low, the explanation manager generates an additional explanation dialog, which tries to impart the missing knowledge for the user to execute the dialog step successfully. The explanation manager selects which type of explanation is appropriate for the current lack of knowledge. This explanation may consist of several parts. Each planned task and its concepts are analyzed to generate a summarized explanation. The explanation manager sends the content of the explanation to the knowledge base to be stored in the information model. This explanation may consist of pictures, text or text meant to be spoken. Afterwards, a dialog step is generated, which references the content of the explanation stored in the information model and sent to the dialog management to be included in the course of the interaction cycle.

Taking a closer look at the implemented approach, a possible extension of the presented approach is a more thorough treatment of uncertainty: though the modeling of the user's knowledge is made in a realistic fashion, considering the uncertainty of one's knowledge distribution, the process of updating the knowledge over time is not. While presenting an explanation to the user will increase the chances of understanding, it will not guarantee it. Therefore, the update process of users' knowledge should integrate uncertainty as well [126]. It should not only include using a probabilistic knowledge modeling approach, but also the integration of information indicating understanding (e. g., user affective states like engagement, interest, disposition). However, contrary to the approach used here, real-time issues might become relevant, as one would have to be careful about the increasing complexity of the probabilistic model. Why this treatment of affective states is of utmost relevance for the design of user-adaptive dialogs will be elucidated in the following section, which reviews the various evaluations conducted for both individual components and component combinations.

## C Evaluation

In the last section we showed how different types of explanations are integrated into our *Companion*-System, but it is still unclear how this influences the user-experience in detail. Hence, we conducted several evaluations to test the design of different explanation strategies and their effects on the user-experience.

These evaluations can be structured into two main research questions covering two distinct situations in human-computer dialog: On the one hand situations where explanations are provided to treat unexpected or incomprehensible system behavior (e. g., incomprehensible generated plans or plan repair). On the other hand situations where learning or plain conceptual explanations are necessary for assisting the user in solving a task due to an estimated lack of user knowledge. For both situations we aimed to investigate how and when explanations are best provided to foster optimal user experience.

**Incomprehensible System Behavior** Incomprehensible situations bear the danger of influencing the relationship between human and technical system negatively [198]. They may lead to a change or complete abortion of the interaction [189]. Those situations do usually occur due to incongruent models: during interaction the user builds a mental model of the system and its underlying processes determining system actions and output. However, if this perceived mental model and the actual system model do not match, the situation is perceived as incomprehensible. This system behavior is likely to occur in intelligent and complex systems like *Companion*-Systems, where proposed plans or upcoming actions may be unexpected or not understandable for the user. Analogous to human-human interaction providing explanations in these incomprehensible situations in HCI can reduce the loss of trust [117]. For human-computer trust (HCT) Madsen and Gregor

(2000) [177] constructed a hierarchical model, where personal attachment and faith constitute *affect-based* trust; and understandability, technical competence, and reliability constitute *cognitive-based* trust. Previous research concentrated on showing that explanations can influence HCT in general [110]. Hence, what is lacking currently is which explanations do influence which components of human-computer trust. Therefore, the goal was to change undirected strategies to handle HCT issues into directed and well-founded ones, substantiating the choice and goal of explanation.

For that we conducted a web-based study [62], where unexpected, and incongruent to the user's mental model, system events were influencing pro-actively the user's decisions. Here, without warning, the user was overruled by the system and either simply informed by this change, or was presented an additional justification or transparency explanation. Justifications are the most obvious goal an explanation can pursue. The main idea of this goal is to provide support for and increase confidence in given system advices or actions. The goal of transparency is to increase the user's understanding in how the system works and reasons. This can help the user to change his perception of the system from a black-box to a system the user can comprehend. Thereby, the user can build a mental model of the system and its underlying reasoning processes. The main objective of the participants was to organize four parties for friends or relatives in a web-based environment.

The results showed that explanations can significantly help to reduce the negative effects of trust loss regarding the user's perceived understandability and reliability of the system in incomprehensible and unexpected situations. Especially for the perceived understandability, meaning the prediction of future outcomes, transparency explanations fulfill their purpose in a good way. Additionally, they seem to help with the perception of a reliable, consistent system. We also found that the use of explanations in incomprehensible and not expected situations can help to keep the human-computer interaction running, as there was a significant reduction in drop-out rate when providing explanations (about 12%).

In another experiment the results were proven to hold across domains [51]. Here, we showed as well that increasing the user's perceived system transparency by including valid explanations on incomprehensible system behaviors may mitigate the negative effects, thus increasing the potential areas of application for these complex *Companion*-Systems.

**Missing Knowledge** Because of the increasing capabilities and functionalities of *Companion*-Systems, they also become increasingly complex to operate, and less intelligible for the user. Hence, it is more likely that the interaction between human and *Companion*-System may exceed the user's knowledge or capabilities. Therefore, such systems should adapt their content and course of interaction to the user's knowledge and explanations are vital and appropriate instruments for that. Besides the well-researched area of modeling and appropriate selection of knowledge, of importance is also how it is presented to the user.

On this account, we conducted a study to test how temporal and spatial distances of providing explanations in a cooperative decision-making process affect the user experience [39]. We aimed at gathering insights into how individual users perceive different explanation strategies to help to derive layout criteria, select appropriate media types, and structure the dialog in future cooperative *Companion*-Systems. For the presentation of the explanations we used the model-driven user interface generation provided by the Interaction Management (see Section VII). A model-driven approach is on the one hand required for the universal and not specified application domain of *Companion*-Systems, but on the other hand restricts the presentation form. For example, in challenging situations, in which an extensive explanation in combination with an underlying selection is needed, the size of the screen may be exceeded. This results in a UI either in form of a sequence of multiple screens (explanations plus selection) or in one (scrollable) UI, hence varying the spatial and temporal distances between explanation and selection task. To test the different effects of these conditions we also assessed UX during a selection task without explanations as baseline.

In general we found that both temporal and spatial distances of the presentation of explanations relative to decision-making (i. e., a selection) influences user experience. Specifically, the results showed that providing explanations separately in advance makes sense when the amount of content would impair the presentation form. However, if a convenient method for presenting the explanation content on the same dialog is possible without impairing the modality choice, this is the best option. In addition, we found correlations that indicate that extraverted participants seem to profit from the presentation of graphics, whereas neurotic persons seem to suffer more from a low quality of explanation dialogs because for the jointly, only-textual, presentation, neuroticism correlated negatively with perceived system attractiveness and hedonic system qualities [39].

## VII Interaction Management

Mark Weiser's vision of ubiquitous computing [184] is in the process of becoming reality. The current rapid technological progress provides us with a plenitude of technical systems, services, and personal interaction devices. The increasing number of such interaction devices

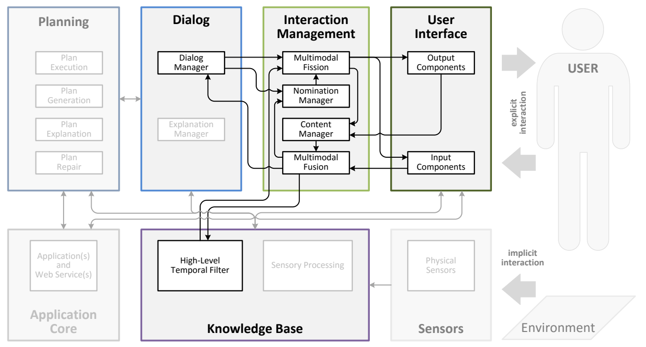


Figure 13: Architectural view and modules with focus on the interaction management.

and the idea of ubiquitous computing in combination with multimodal interaction concepts lead to complex interactive systems. Today, the interweaving of consumer electronics and devices is still in its infancy as it is rather hard-coded and does not respect individual preferences.

In the context of individual *Companion*-Technologies, the interaction management addresses the requirement of presenting an individualized user interface. The current section focuses on adaptive multimodal interaction. Subsection A describes how multimodal fission can be used to realize an individualized user interface (UI) at runtime with respect to ongoing changes in the context of use (CoU) [182]. Then, Subsection B discusses a possible realization for multimodal input fusion, and Subsection C shows how the interaction management's two major components can interact with and benefit from each other.

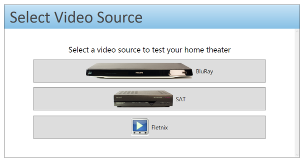
The modules, which are used to realize such processes, are depicted in Figure 13. As an example, the dialog management module provides a modality-independent dialog output (Figure 14 (a)) for a selection. Thereby included optional nominations can influence the dialog output's way of representation and are thus kept in memory by the nomination manager. Triggered by the dialog output, the multimodal fission reasons about the most adequate UI's configuration. Thereby, the reasoning process fetches additional knowledge from the knowledge base. As a result, the multimodal fission forwards the interface configuration as a so-called interaction output to the addressed input and output components for rendering (cf. Figure 14 (b+c)). This configuration, as well as the output components' final layout descriptions are passed on to the content manager. With such knowledge, the content manager can configure the multimodal fusion module. In that way, cross-modal inputs (e. g., speech plus pointing gesture) can be fused to perform an input for the offered selection. Identified inputs are passed back to the dialog management component for further processing. In addition, the multimodal fusion can also identify user-given nominations, which are passed on to the nomination manager. For a more detailed view on multimodal fission and fusion we refer to the work by Schüssel et al. (2017) [27].

## A *Multimodal Fission*

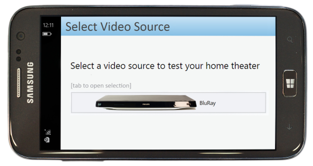
The multimodal fission component has to reason about a suitable UI in a scenario, in which the CoU as well as the device topology can change unpredictably. Also, influencing data is based on sensory information, which is affected by uncertainty. In the following, we describe how multiple input and output modalities can be utilized, and how the decision process of modality arbitration can work successfully, even with uncertain data.

```xml
<?xml version="1.0" encoding="utf-8"?>
<dialogOutput dialogID="video_source_selection">
  <topic>
    <abstractInformation objectID="topic" informationID="video_source_selection_topic"/>
  </topic>
  <dialogAct>
    <!-- optional nomination -->
    <desiredOutputChannel>visual</desiredOutputChannel>
    <!-- regular content -->
    <selection objectID="video_source_selection_container"
               informationID="video_source_selection_prompt">
      <abstractInformation objectID="BluRay" informationID="BluRay_information"/>
      <abstractInformation objectID="SAT" informationID="SAT_information"/>
      <abstractInformation objectID="fletnix" informationID="fletnix_information"/>
    </selection>
  </dialogAct>
  <listen>
    <abstractInformation objectID="video_source_problem"
                         informationID="video_source_problem"/>
  </listen>
</dialogOutput>
```

(a) Description of a modality-independent dialog output that contains six information items and serves as abstract user interface (AUI). The intended output contains a topic plus a selection offer. The selection comprises a selection prompt and three selection items.



(b) The final user interface (FUI) as a rendered selection offer on a desktop screen. The generated output is based on the given dialog description (a).



(c) The dialog from (a) as rendered on a smart phone. The rendering of the selection concept can be automatically adapted on the FUI level by using a pull-down list for selection, because of the small screen size.

Figure 14: A modality-independent dialog output (a) is sensed as AUI. This invokes the reasoning process for modality arbitration and results in individual user- and context-dependent user interfaces on the FUI level (see (b) and (c) as two possible examples for the visual channel). The intended listen item is realized with the use of automatic speech recognition. Cf. [2, Fig. 4.6, p. 122 and Fig. 4.8, p. 126].

As shown in Figure 13 and described in Section A, the interaction management's fission component gets activated whenever the dialog management provides a new output description as part of its current dialog strategy. This so-called *dialog output* (see Figure 14a) represents a modality-independent description of an abstract user interface (AUI), which originates from the currently active task from the planning execution component. Based on that and with regard to the CAMELEON reference framework (CRF) [160], the fission's reasoning process for modality arbitration has to solve the mapping problem from the AUI level to the concrete user interface (CUI) level. It individually decides about which concrete encoding concepts shall be used to communicate a specific information item. A CUI description is then passed on to the addressed device components, which are in charge of rendering the content as final user interface (FUI) (see Figure 14b, 14c, and Section VIII). The reasoning process depends on diverse knowledge items (see Figure 13). The interface to the knowledge base provides such knowledge in terms of probability distributions over different values (cf. Section IV). Different knowledge items can be clustered to build different models, e. g., a user model, an environment model, or a model representing user-to-device distances. These models comprehend dynamic variables, since their data stems from continuous sensory data acquisition. In addition, the knowledge base provides static or quasi static knowledge in terms of two other models. The one model describes devices and their linked components for input and output. In that way, each component can be described in terms of its supporting encoding or decoding concepts, its location, or other attributes. The other model provides possible information mappings from modality-independent information variables to modality-specific encoding or decoding concepts for text, picture, ASR grammars, text-to-speech templates, or other concepts [83, 61].

The idea of an intelligent and individual output configuration is addressed by many researchers. According to Costa and Duarte (2011) [90], systems that combine different output modalities like text and speech evolved since the early nineties. The allocation of output modalities of the early multimodal systems was hard-coded with very simple logic and not based on intelligent algorithms. Recent concepts that utilize multi-agent systems [93] are based on plan-based composing [176], or make use of rules to identify the most adequate combination for multimodal output [90, 83]. While some approaches tend to exploit all possible variants of multimodal representation [96], this might conflict with the guideline to "address privacy and security issues" [159]. Based on that, it looks promising to employ an ongoing adaptation to the context of use, as motivated by the last of the WWHT questions[4] by Rousseau et al. (2006) [140]. Based on the definition of context by Dey and Abowd (1999) [182], a model of the context of use (CoU) can include (amongst others) information about the environment, the user, and the available devices. So, talking about adaption to the CoU also includes strategies respecting user preferences, as identified by Pruvost et al. (2011) [96] and Schüssel et al. (2012) [86].

Today, rule-based approaches can be seen as established practice in order to react to changes in the CoU [96]. Zaguia et al. (2013) [75] apply an ontology applied for context-dependent modality arbitration. However, their approach lacks the opportunity to express uncertainty values, which is, for example, of great importance when fusing contradicting statements. Recent work goes together with model-driven UI generation as described in the CRF. This means that a model of a modality-independent output has to be transformed to a modality-specific kind of output via a certain mapping process. The mapping can be either hard-coded or an adaptive reasoning process. We will focus on the latter, as proposed by Honold et al. (2012) [83].

In the remainder of this section, we present our approach for modality arbitration. First, the dialog output gets analyzed and split up into its single abstract information fragments. In a second step, the fission component identifies all possible variants of concrete information encoding on each available device component. According to Nigay and Coutaz (1995) [195] such a "coupling of a device $d$ with an interaction language $L$" is called *interaction technique* (a.k.a. modality). Items of similar semantic meaning are clustered together. Afterwards, the modality arbitration is done for only one of them, and case-based reasoning (CBR) is applied for the others. Next, all available device models are inspected in order to identify the possible modalities for each information item. Therefore, all possible modalities, as well as their multimodal combinations have to be analyzed. A valid combination of modalities is called output configuration $oc$. For $n$ possible modalities, there are $2^n - 1$ possible $oc$s. If, for example, the abstract `BluRay_information` (as referenced in Figure 14 (a)) could be rendered as text, picture, or video on 6 possible displays (= 18 modalities), and could also be rendered using Text-to-Speech (TTS) via 2 different speakers (= 2 modalities) in combination with potential speech input via 3 different microphones (= 3 modalities), than this single information item could be realized via $2^{23} - 1 = 8,388,607$ different $oc$s.

Modality arbitration takes place in the next step. Re-

[4]WWHT questions: *What* is the information to present? *Which* modality(ies) should be used to present this information? *How* to present the information using this (these) modality (modalities)? and *Then*, how to handle the evolution of the resulting presentation?

ferring to Figure 14 (a), for each of the six applied abstract information items the most adequate one out of the $2^n - 1$ possible *oc*s has to be identified. The reasoner can rate each possible *oc* by using a given set of reward and punishment functions. Each function's reward is biased, based on the probability of their activating knowledge items, which are provided by the *Knowledge Base's* static and dynamic context models. The meaning of each function stems from domain-independent design rules or is influenced by study results, e. g., by Schüssel et al. (2012) [86]. Each identified optimal solution for a given mapping problem is kept in memory for similar problems in the future. The highest-rated output configuration represents the final candidate for rendering. A process of probabilistic fission is described in detail by Honold et al. (2012) [83].

Beyond that, desires or dislikes of developers or users can activate nomination-specific functions. This is done by so-called nominations (see Subsection C), which are managed by the system's *Nomination Manager*. Besides simple references on information items, the dialog output can also include abstract references to specialized widgets as, e. g., widgets that realize a calendar, a media player, or a web browser (see Section VIII). The reasoning about the concrete widget realization is realized analogous to the reasoning about the information items.

Finally, the highest-rated output configurations for information items and/or widgets are assembled to form the modality-specific interaction output. This output is passed on as CUI to the involved device components for rendering (see Figure 13 and Section VIII). It is also passed on to the content manager in order to configure the fusion module for possible interaction inputs (see Section C).

Based on our prototypical implementation, we identified two points for improvement. First, the use of simple 1-dimensional user-to-device distances does not allow respecting device-to-device distances in the reasoning process. The use of 3-dimensional representations for object- and user locations, as well as knowledge about the user's gaze direction could help to identify modalities, which are out of the user's sight. For interfaces with multiple GUIs, this could also help to prevent scattered UIs, where one part of the UI is displayed in front of the user and another part is displayed behind him or her. The second improvement concerns the process of determining the highest-rated output configuration. In the current implementation, it is determined by simply enumerating all configurations. As the number of possible output configurations increases exponentially in the number of modalities, it would be preferable to use a combinatorial optimization algorithm instead, such as simulated annealing.

## B Multimodal Fusion

Once the fission component has decided on how to present an interface, the fusion component needs to be informed about the resulting interaction possibilities. Based on that, the fusion is able to decide if different user inputs are ambiguous, conflicting, or reinforce each other. Our current approach [72, 27] applies evidential reasoning as a generalization of probabilities to provide robust fusion results. Compared to other approaches of decision level fusion for HCI [188], which can be classified as performing a procedural, frame-based, unification, or hybrid/statistical type of fusion [109, 107, 46], the approach can be categorized as hybrid fusion, which merges a frame-based data model with a unification operation. While not providing the power of describing complex sequences of inputs that span multiple interactions, as this is the task of the dialog management in a *Companion*-System, the approach comes with the benefit of providing a formally sound model for ambiguity for the tasks of reinforcement and disambiguation of multimodal inputs. The often rule-based approaches found in literature either ignore the presence of uncertainty (like the approaches by Dumas et al. (2012) [79] and Olmedo et al. (2015) [52]) or rely on simple n-best lists for making a decision (like the one by Cohen et al. (1999) [181], Bouchet et al. (2004) [154], Russ et al. (2005) [150], and Sun et al. (2006) [142]).

In order to perform its task, the fusion component needs to be provided with an Abstract Interaction Model (AIM) that states all actions in the domain at hand the user could possibly trigger via the available input device components. In addition, the AIM must contain all domain-specific knowledge on how different inputs should be semantically combined. The AIM uses the concept of graphs with nodes (representing possible inputs from input device components) and edges (representing their combinations) to hold this information. The specification is done in GraphML syntax [167]; the details of which are out of scope here. Within our approach, the interaction management's content manager (cf. Figure 13) is responsible to provide this kind of information, as separately explained in the upcoming subsection. An exemplary AIM is visualized in Figure 15.

The AIM shows that the user can state *selections*, can make *references* to objects, and can perform *requests* for additional information in several ways (green boxes). To elucidate this, imagine the situation where the user states "give me more information about that" and at the same time points at the HDMI cable. In such a situation, the ASR component would raise an input, containing just a *request* of *type* 'explanation' (Section B explains how this type of explanation is handled). The gesture component would raise an input containing

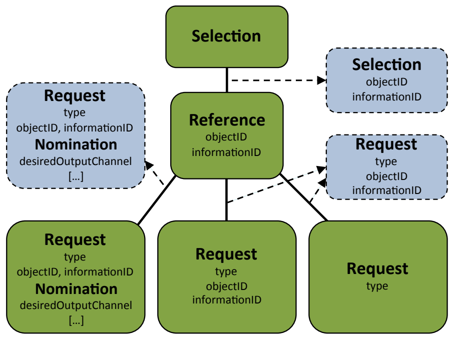


Figure 15: The AIM created by the Content Manager from the interaction output [60, Fig. 2] and [2, Fig. 4.16, p. 151]. The graph expresses all possible inputs (solid nodes) and their resulting combinations (dashed nodes). Some of these combinations are complementary (e. g., Selection + Reference), while others represent redundant inputs (e. g., Reference + Request/Nomination).

a *reference* to the *objectID* 'hdmi' and *informationID* 'hdmi_information'. As defined in the AIM via the edges between the input nodes, the fusion component is able to combine these two inputs and create a complete *request* (blue box) that contains the type 'explanation', as well as the objectID and informationID of the HDMI cable. In addition, the confidence values given by the input components are taken into account to make sure that only the most probable input results are forwarded to the dialog management.

A *request* can also occur together with a so-called *nomination*, i. e., the user requests additional information in a specific way. For example, the user could state "tell me more about the HDMI cable", which would result in an input as shown in the green box in the bottom-left corner of Figure 15. Since all needed information is already contained in the input, it can directly be transformed to a complete request with nomination, without the need of combining it with other modalities. Since the semantics of nominations are modality-specific, they are not relevant for the dialogue management but only for the fission component. Thus, such nominations are forwarded to a dedicated nomination manager component (cf. Figure 13), as explained separately in the next subsection.

Though the current implementation provides all necessary functionality like combination, disambiguation, reinforcement, and conflict detection of inputs, there may be situations where the system does not behave as intended by the user, as there might be recognition mistakes of particular modalities, e. g., a wrong speech recognition due to ambient sounds. This fact has already been acknowledged in the literature (like by Dumas et al. (2012) [79]), but has not been widely addressed yet. The abstract interaction model, which is the starting point for the multimodal input fusion, can also be used to detect and recover from such sensor errors. The idea is that past multimodal user behavior can be stored as interaction history in terms of temporal behavior as described by Schüssel et al. (2014) [66]. Given that a temporal behavior for a specific user in a specific situation is known, a new input can be detected as atypical and therefore be exposed as a sensor error. In addition, the interaction history can be used to resolve some of the exposed errors, thus avoiding the need for user feedback on ambiguities.

For this to work, a user's interaction history must be associated with a specific situation. Here, the AIM comes into play, as it allows identification of different cases. This can not only be done on a complete AIM basis, but on each individual edge of the graph as well. In this way, an interaction history can be applied on multiple AIMs. Figure 16 depicts this process for an exemplary generic case.

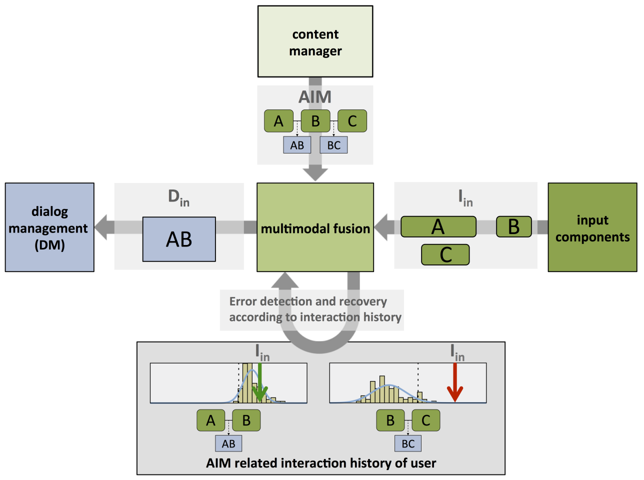


Figure 16: The figure illustrates the process of detecting and recovering from sensor errors using a stored interaction history. In the example, a false positive interaction input $C$ is detected and the correct input $AB$ is forwarded to the dialog management.

The input components provide the multimodal fusion component with an interaction input as time-based intervals ($I_{in}$). Within the application example of this report this could be a selection trigger ($A$), an object reference ($B$), and an information request ($C$), as described in Section B. Given the AIM from the content manager, there are two possible outcomes: either the user wants to select the object ($AB$) or requests more information on the object ($BC$).

In order to decide which is the most probable input, the multimodal fusion component checks the interaction history of the user for each edge and adjacent nodes of the AIM. As visualized in the lower part of Figure 16, the temporal relation of the selection ($AB$) much better fits the user's past behavior than that of the request for more information ($AC$). Thus, it is concluded that the information request ($C$) must have been a sensory fault (more precisely, a conflict triggered by a false positive) and that the selection ($AB$) must have been the correct input, i. e., the user most probably wanted to select the referred object. This correct input is then forwarded to the dialog management ($D_{in}$).

This approach for error detection could easily be integrated into the application scenario of this report as it also uses some kind of selection tasks, for which a considerably decreased error rate has already been shown by Schüssel et al. (2016) [40], where the error detection and recovery process is covered in greater detail.

### C *Interplay of Multimodal Fission and Fusion*

While fission and fusion are usually described and implemented in a self-sufficient way [195, 107], the two components for fission and fusion can both benefit from each other when an actual interplay is realized. Therefore, we introduce two additional components to realize the linkage: the content manager and the nomination manager (see Figure 3). This subsection summarizes findings of Honold et al. (2014) [60] and Schüssel et al. (2017) [27] and presents a novel explicit collaboration of fission and fusion.

#### Content Manager

As soon as the fission component's computation is done, and therefore the CUI is realized via device components in form of a FUI (cf. Section A), the input fusion component needs to be configured for all resulting interaction possibilities. This task is realized by the content manager. Although the abstract model of interactions that can occur is predefined (e. g., *references*, *selections*, etc., as described in Section B), it depends on the FUI which of them are actually available in the current dialog step.

Inspecting the CUI from the fission component and the FUI descriptions from the device components, the content manager creates the AIM in the form of an undirected graph (cf. Figure 15). Nodes of the graph represent all single events that can occur via the available input device components, while the edges contain information in the form of XSLT transformations on how to semantically combine multiple single events. This way complementary and redundant multimodal inputs [192] are modeled.

Using this approach, fission and fusion can both work on different models and abstraction levels that best fit their respective purposes. In addition, a dedicated component like the content manager allows the input fusion to be domain-independent and reusable in completely different applications.

#### Nomination Manager

As motivated in Section B, a user may demand a specific output configuration as final UI. To address the *Companion* characteristics of individuality, adaptability, and cooperativeness, the user shall be able to submit any desires and dislikes for possible channels (i. e., aural, visual, and tactile), encoding concepts (e. g., text, picture, video, and text-to-speech), or used device components. Each of such statements is encoded as a so-called *nomination* and can be linked with a specific dialog-, object-, or informationID. Nominations can also originate from the dialog manager in order to support an envisioned dialog strategy.

Nominations are passed on to the nomination manager. On each arrival of a new nomination, and with knowledge about the actual output, the nomination manager decides about a necessary repetition of the fission process. The fission in turn is able to respect all matching nominations with each reasoning process. Nomination-specific reward and punishment functions are used for biasing the rewards of the analyzed output configurations in the intended way (cf. Section A). User-given nominations do have a higher influence than the ones that stem from the dialog manager.

### D *Evaluation*

As motivated by Schaub (2014) [65], human-computer trust can be increased if a system respects its user's privacy concerns. We analyzed the reasoning results of the fission with respect to their compliance with given demands for privacy.

The reasoning process for modality arbitration is a problem, for which no polynomial time algorithm is known. With the use of $n$ modalities, the fission has to decide which of $2^n - 1$ possible output configurations ($oc$) shall be used for realizing an output item in the final UI. Accordingly, a problem with $n = 20$ available modalities provides more than one million possible solutions. That is why we expect higher rewards in cases where the system can pick its optimal $oc$s from a large pool of possibilities than in situations where only a few modalities are available.

For further insights, we conducted a performance analysis to measure different aspects of the fission's reasoning process. In this section we focus on the so-called reward factor $\langle oc \rangle$. The factor represents the ratio of an $oc$'s gained reward to a theoretical reward maximum. This maximum can be derived as an a-priori estimation from all available evaluation functions in a given context of use (CoU) without further knowledge of a par-

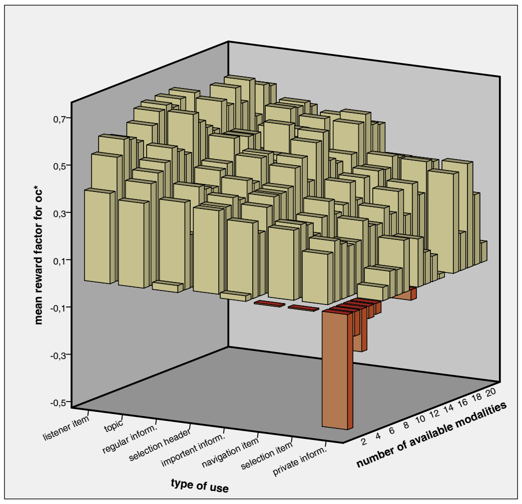


Figure 17: Achieved mean reward factors in modality arbitration. The mean of the achieved reward factors tend to increase with an increasing number of available modalities. If an object represents private information, the gained reward significantly drops in some cases (colored in red). This is because even the highest rated output configuration *oc** does not comply with the required privacy recommendations as employed in the activated privacy-specific evaluation functions.

ticular *oc*. In the experimental setup we used two dialog outputs of equal structure, but referencing 28 different information items (14 items per dialog output). Each dialog output contained one item representing private information (cf. Figure 17). We asked nine different persons (3 female, 6 male) to provide their individual user model instances as part of the CoU. We provided two different environment models (home vs. office) and simulated different devices. The probability distributions for the user-to-device distances were set randomly, according to the distributions as provided by our real-world sensors. The setup allowed us to simulate test cases with up to 20 randomized modalities. In total we evaluated all possible output configurations for 80,640 output items from 5,670 dialog outputs. Case-based reasoning (CBR) was applied in 28.7 % of all cases. We clustered the output objects according to their type of use in order to gain insights in the distribution of reward factors for different problem sizes (see chart in Figure 17). Objects for which CBR was activated were excluded from the analysis and do not affect the chart.

Figure 17 shows the distribution of mean reward factors for output items from different types, as expected. An increasing problem size (number of available modalities) offers the ability to identify an optimal output configuration with a higher reward. The visualized reward factors in Figure 17 may also decrease because of the randomized modalities and the randomized distance distributions (cf. results for type *private information* in the range of 12–15 available modalities). The values for the type of *private information* show that it is not always possible to provide a sufficiently optimal solution with respect to privacy, especially with a very limited set of possible modalities, e. g., with components like public displays and/or speakers.

To address the *Companion* characteristics of cooperativeness and trustworthiness even in these situations, and to overcome the problems caused by particular device components, we plan to add a privacy enforcement module to the output components on the level of the final UI as motivated by Honold et al. (2012) [83]. A possible integration of a privacy decision engine that reasons about the correct privacy policy is described by Schaub (2014) [65] on page 223.

A further evaluation of multimodal interaction is described by Bubalo et al. (2017) [18]. Said work also shows how the use of a so-called *interaction history* can be used for error detection and recovery by the multimodal fusion process.

## VIII User Interface

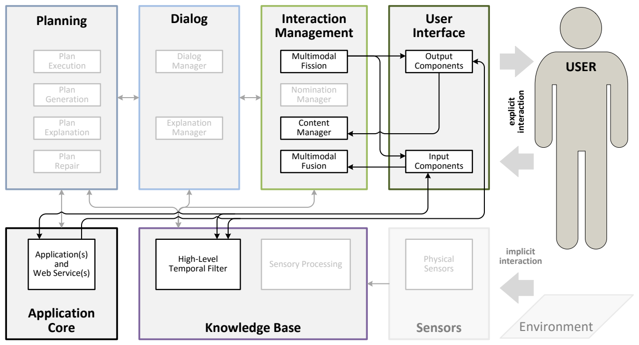


Figure 18: Architectural view and modules with focus on the user interface.

To implement the *Companion* characteristics of multimodality, individuality, adaptability, availability, cooperativeness, and trustworthiness our presented architecture for *Companion*-Technologies allows removing and integrating new devices and sensors at runtime that realize the user interface (UI). Such a UI consists of types of device components, namely output components and input components as depicted in Figure 18. Each *Companion*-compatible device component provides an XML Schema-based description of its capabilities for specific input decoding or output encoding. This knowledge is stored in the knowledge base and is used by the fission to determine the possible modalities. The knowledge base requests so-called heartbeats from the UI components via continuous broadcasts. The received responses allow to decide about each component's availability or inactivity. Each user interface component is in charge of rendering its assigned part of the final user interface assigned by the multimodal fission module. Input components are provided with their operational parameters (e. g., grammars for speech recognition), output components are provided with a concrete description of the intended output (e. g., semantic structure plus content for visual output). Devices are used to organize several device components for input and/or output. Visual output components provide the interaction management's content manager with information about their rendered items in terms of regions of interest. This allows the multimodal fusion module, e. g., to map pointing gestures to objects on the screen, even if the particular sensor is not connected with the screen's host device. Input components pass on their inputs to the multimodal fusion and send messages to the knowledge base whenever a user interacts with them, so the knowledge base can draw conclusions about a user's current position (cf. Section B). Widgets are used to wrap complex interaction concepts (e. g., a media center UI) in a single and coherent UI component. Widgets are available for input and Output components and allow to directly communicate with a linked application or web service, as motivated by Honold et al. (2011) [94].

Components for input, e. g., a speech recognizer, are provided with suitable configuration data. In that way, the *video_source_problem* listen item from the third last line in Figure 14a allows a user to take the floor and initiate a dialog, based on a context-specific ASR grammar. The grammar facilitates detecting cable problems based on verbal utterances, like, e. g.: "The cable is broken." or "The plug is detached from the cable." If no ASR component is available, the fission may instruct a graphical UI component with the listen task. This is realized as depicted in Figure 7. The "X" symbol on the left of the content window allows submitting text messages to the system. In this case text input can be analyzed in the same way as speech input. On the abstract semantic level, the dialog manager does not recognize any difference with respect to the user's applied input concept.

Adaptation on the FUI level allows, e. g., to apply different rendering concepts when providing a selection. As exemplified in Figure 14, size constraints as well as the amount of selectable items influence the applied concept. In this case, plain buttons are presented as selection offer for less than three items on smart phone screens. The concept of a pull-down selection list is used for options with more than two items (see Figure 14c). On other screens, plain buttons are used as selection offer for up to seven items.

In order to also support complex widget-based interactions (e. g., to include a fully functional map component, a media center application, or other already existing user controls), device components are able to instantiate external user controls at runtime via reflection, as described by Honold et al. (2011) [94]. Currently, this feature is only supported for GUI-based components. Whenever an intended widget is referenced on the abstract level in the dialog output, the fission can identify possible candidates by matching a referenced widget with a description of all known widgets (see Figure 19). Based on that, it is possible to identify all possible mappings to the CUI with the use of the descriptions about the known device components. This concept enables the *Companion* to integrate widgets even from remote locations at runtime.

```xml
<?xml version="1.0" encoding="utf-8"?>
<widgetSet>
  <widget widgetID="mediaCenter"
          description="a widget for media browsing and consumption">
    <cuiWidget location="http://www.myurl.com/pathTo/MyMediaCenter.dll"
               assemblyName="MediaCenter"
               typeName="MediaCenter.Controls.MediaCenter"
               outputEncoderMedium="gui"
               preferredHeightPx="800" />
  </widget>
  <widget widgetID="mapDemo"
          description="a dummy widget with a map application">
    <cuiWidget location="file:///C:/Somewhere/WidgetAssembly.dll"
               assemblyName="WidgetAssembly"
               typeName="WidgetAssembly.Controls.MapDemo"
               outputEncoderMedium="gui"
               preferredHeightPx="500" />
  </widget>
</widgetSet>
```

Figure 19: Mappings for possible widgets are used by the fission to identify possible implementations. On the abstract level widgets are described only with the use of their unique widgetID.

## IX Building an Assistance System

So far, we have seen how the various components of our system interact with each other in order to provide advanced assistance for a certain task that involves the execution of a series of actions. We have illustrated the application of the various technologies and components and their interaction in a specific application domain, where many people need or would appreciate assistance: the assembly of complex hardware components (see Section II). The deployed technologies, as well as the underlying system's architecture, is completely domain-independent, however. That is, when the various components are provided with an adequate description of the given problem at hand, the described system can provide support in many different application domains. In the previous sections, we have focused on describing the involved sub components and the actual models used in our prototype system. Here, we shift the focus towards practical issues that arise when one tries to apply our approach to a new application scenario and give an impression of the effort required for doing so.

**Knowledge Base** In our application scenario, the MLN models were not very complex, as sensory input was quite limited – it was only used for user localization with a limited number of areas where the user can interact with the system (cf. Figure 5). The model consisted of 7 rules, plus 24 static facts that represent the area adjacency graph. Concerning the difficulty of constructing MLN models in other application domains, Jain (2011) [95] give valuable pointers to pitfalls for knowledge engineering with MLNs. For the user localization, the choice of the sensors and their mounting positions are crucial for the system performance and strongly depend on the desired scenarios. Since the multi-object tracking system uses generic sensor interfaces, it is not necessary to change the filter core if an additional sensor is added to the perception system or a currently used sensor is replaced by, e. g., a new model with higher resolution. Consequently, only the sensor-specific measurement model has to be implemented for a sensor type, which has not been used before. If a sensor is replaced by a similar sensor with higher resolution, it is sufficient to adapt the parameters of the existing measurement model used for the sensor with lower resolution.

**Planning** Depending on the domain at hand, many steps of constructing a planning model can be automated. In the home theater scenario, for example, formal descriptions of the available ports, etc. of the home theater devices could be constructed by the manufacturer and shipped with the device or supplied online. In other cases, parts of the planning model can be inferred from readily available ontological knowledge, e. g., as demonstrated by Behnke et al. (2015) [44] in a fitness domain. We did, however, model and test the domain completely by hand. While domain modeling and its testing can be a complex task – in part due the sparse availability of intelligent tool support, in particular for hierarchical models [33] – it is still manageable in our example scenario, as there are only a few actions available. In principle, it suffices to have one single *plugIn* action (which makes use of several parameters, i. e., variables for devices and cables), as explained in Section V. However, this number heavily depends on several modeling choices, which in turn depend on the available planning language features. In one of our models of that domain, we used 16 tasks, which have up to 6 parameters. Each of the cables and adapters (in our scenario, there were 11 and 2 of them, respectively) as well as devices (4 in our scenario, see Section II) featured a number of ports, each one was modeled by an individual constant (each cable had 2 to 3 ports, for each device we modeled up to approx. 10 ports).

In any case, one needs to carefully consider the requirements of the given scenario. The hybrid planning formalism introduced here relies on a deterministic action model. Failures and unexpected behavior are handled via repairing plans. When the domain at hand requires handling uncertainty in action execution or sensory uncertainty at plan generation time, *Companion*-Systems can employ planning based on the framework of partially observable Markov decision processes (POMDPs [203]). In fact, controlling a *Companion*-System as a whole can be interpreted as solving an enormous POMDP problem [21, 53]. More practically, Richter and Biundo (2017) [25] describe how an approach that generalizes HTN planning to POMDPs [84] can be applied to an extended variant of the home theater assembly task that exhibits uncertainty in action execution as well as sensory uncertainty. Using this approach in a *Companion*-System alleviates the need for a Plan Repair component at the price of more limited plan explanation functionality, since the approach does not represent causal dependencies in plans.

A further limitation of our prototype system is that we currently do not handle time or action durations – only relative dependencies between actions can be represented and the execution of an action happens instantly. That is, we can model simple temporal relations such as "I have to shop for groceries before I go to the doctor", but handling time points such as "I have a doctor appointment at 10 am" is not possible. These features are conceptually integrated into our planning framework [113], but not yet implemented in our prototype system.

**Dialog** Providing the dialog components with all required data is closely coupled with modeling the planning domain. As explained in the planning and dialog sections (Sections V and VI, respectively), every action that needs user interaction has to have an associated dialog model, which allows its presentation to the user. Therefore, the level of abstraction of actions and the dialog model has to be chosen with care, as explained in both sections. If individualized dialogs are desired for the action, e. g., for different knowledge levels, this has to be predefined. This individualization may require variations of the content. For example, if two home theater components have to be connected via a cable, an expert instruction would require only pictures of the cable and the respective components and text indicating the instructions. However, if an individualized dialog for a novice user is desired, this action may be split into two more detailed ones, where each dialog highlights the port of the component a specific cable end has to be plugged into. As can be seen in Figure 7, which depicts a user instruction to plug the audio end of the SCART to cinch cable into the respective audio port of the amplifier, this requires the generation of two pictures, each highlighting the respective port of the component. This shows that individualization of the dialog may require extra effort for the designer and thus has to be handled with care. Further, this requires a user model for all steps that are involved in executing these actions. This user model is also used to decide whether explanations generated by the dialog component (cf. Section B) are presented proactively. In any case, these explanations require the upfront availability of the content of these explanations in the desired form (text, pictures, or videos). This content has to be specified by the domain modeler or gathered, e. g., using information retrieval and natural language processing techniques. This is in contrast to the plan explanations (cf. Section D), which are generated during runtime.

**Interaction Management** The interaction management cooperates with the UI components in a fully model-driven manner. The UI components are implemented as domain-independent interpreter of our specific XML-based user interface description language as CUI. As the description of the final look and feel is inferred from the dialog manager's modality-independent output at runtime, we only have to specify missing mappings for the possible information encodings. In that way the desired UIs can be generated on the fly.

In the current *assembly assistance* domain, the information model comprises 278 mappings from abstract information items to their possible concrete representations (i. e., text, picture, video, TTS, or ASR). Based on the connection topology of device-ports and cable-plugs, 98 of the 278 mappings have been automatically generated. These 98 mappings apply links to corresponding pictures and videos. For these cases we provided 26 pictures plus 26 animated videos. In total, we provided 84 media items. It is also possible to inject new mappings as runtime, as it is done by the dialog management for plan explanation. Right now, these on-the-fly generations are simple mappings to automatically generated textual descriptions, but it would also be possible to add visual encoding concepts (i. e., images, or videos).

In its current setting, the fission process utilizes 92 evaluation functions to reason about the most adequate UI, according to a given CoU. These functions are domain-independent, and there is no need to reconfigure them. Nevertheless, it might be possible that specific domains require additional functions in order to respect specific contextual situations. New reward or punishment functions can be added and are integrated at runtime via reflection. The evaluation functions are activated and weighted by contextual parameters and their level of uncertainty. The current implementation supports 6 parameters for the environment model and 29 user-specific parameters. Additional evaluation functions may require additional context parameters. Since XML serves as exchange format for the context models, these parameters can also be extended, if desired.

Instances of different device models form another class of input for the fission. Such model instances (a.k.a. profiles) are used to describe devices and their components' rendering capabilities. As devices can be re-used across different scenarios and domains, it is likewise with their profiles.

The fusion process is configured by the abstract interaction model (AIM). In our approach the AIM can be automatically generated using XSLT transformations in conjunction with knowledge about the CUI from the fission and the FUI from the utilized UI components. This work is done by the content manager. As its XSLT operation only depends on the semantic structure of the dialog output, it is domain-independent in itself and can remain unchanged in any other setting.

Albeit such a model-driven approach offers maximum flexibility in terms of domain independence, it should be mentioned that the implementation of generic model interpreters for different input and output modalities (including automatic layout mechanisms) are extensive tasks. If the scope of all UI variants is known at design-time, and if they are not large in number, for smaller projects, the linking of pre-compiled widgets at runtime might be a more cost-effective solution.

The adaptive components of the interaction management in cooperation with the UI components were already successfully applied to the domains of *fitness* by Nothdurft et al. (2016) [39] and to *interactive biology-course teaching* by Nothdurft et al. (2014) [61].

## X Related Work

In this section we first give a short overview of some of the most important *research projects* that are related to our motivation of creating truly user-friendly technical systems. In the second part, we get more "concrete" and give an overview about *research and systems* that are concerned – as is the approach described in this report – with providing assistance to humans based on AI planning as its core reasoning mechanism.

### *A Research Projects*

Various research projects address questions and issues that are related to our general enterprise of assisting human users in a natural and user-friendly way.

The *EMBASSI project* [174], for instance, aimed at providing intuitive interaction capabilities for devices of every-day life contexts. It specifically deals with providing assistance functionality through multimodal interaction with anthropomorphic user interfaces. Here, additional emphasis is put on psychological and ergonomic aspects.

A dialog shell for interactive systems was also proposed by the *SmartKom project* [138, 165], which focused on flexible, natural interaction.

The *REAL project* is concerned with assisting its users in different tasks, but focuses on instrumented environments [152]. Their focus lies on the implicit and explicit user interaction, whereas we propose a general system architecture that focuses on providing planning-based assistance in a broad variety of tasks.

Closely related are also the research projects that deal with cognitive technical systems, such as the *CoTeSys (Cognition for Technical Systems) Cluster of Excellence* [99] and the DFG[5] *excellence cluster CITEC (Cognitive Interaction Technology)* [104]. CoTeSys did a large scale investigation of cognition in technical systems with a particular focus on systems with explicit models of their own actions and perceptions. CITEC focuses on cognitive processes related to the interaction and communication between a cognitive technical system and its user.

While many of these projects are highly interdisciplinary, none of these focus on addressing the *implicit requirements* on cognitive technical systems, i. e., the *Companion* characteristics mentioned before (see Section I). For a more detailed discussion on related research projects we refer to the survey by Biundo et al. (2016) [34].

### B Planning-based Assistance Systems

Due to the importance of the topic for a modern society and the variety of related research topics, there is also a wide range of actual systems and general approaches that realize assistance functionality in a variety of different application areas such as robotics, space missions, health, and elderly care – to name just a few of the most active ones [34]. Here, we want to mention those approaches that are closely related to the one proposed here in the way that they base their decisions upon automatically generated plans of action. Please note that there are many practical applications in which the respective system relies upon an AI planning system (see the website `http://sig-aps.org/` of the *Special Interest Group for Applications of AI Planning and Scheduling (SIGAPS)* for many examples), so we here only list those in which a human user is directly involved by some sort of assistance system. Our approach distinguishes itself from the others in many aspects – for example the individualization to the individual user, the pursued planning framework, explanation and execution capabilities, and the advanced interaction capabilities – and we will discuss differences in the following. Our work differs from most of the following in the key aspect that it is completely domain-independent: provided with the required models, it can be used as a basis for a broad variety of different tasks. The works given below are all (but one) addressing a specific problem that they are solving and for which respective assumptions can be made. Further, only very few of the respective publications are precisely explaining the underlying system architecture.

Closely related to our approach is the system *RADAR*, a planning-based decision support system [28]. Like our approach, it is a domain-independent system for providing advice about how to carry out a certain task that is modeled in terms of a planning problem. Though the system shares several main ideas and capabilities of ours (like plan generation, plan repair, and plan explanation), there are several differences as well. One core difference is the underlying planning model: they rely on a non-hierarchical approach. The most prominent difference, however, is the design of the user interface and its capabilities. We decided to enable a detailed, multimodal presentation of the actions in a step-by-step fashion, so that a single user can just follow them. RADAR, in contrast, presents the entire plan as a list of action names. It allows its user to rearrange that list's order as well as to remove and add actions, similar to what mixed-initiative planning systems usually allow. The resulting plan can be evaluated and repair hints get generated in case of problems. Similar to our system, explanations can get generated. However, since RADAR assumes expert users, they do not follow our approach that generates explanations in natural language explaining the purpose of user-selected actions in the solution. Instead, they follow their approach of "model reconciliation", where they use a specification of the user's mental model of the planning task (which has to be available in the same language specification than the planning task itself) and explain differences to the actual model [19]. The user interface also features areas in which the available resources are presented as well as a further area to show additional domain-specific information. In their example scenario, RADAR supports in a fire-fighting scenario. Thus, an overview map of the relevant area is shown; the resources list the available fire engines and equipment per station, which can also be changed by the user.

The work by Yorke-Smith et al. (2007, 2009, 2012) [131, 115, 88] is only loosely related to ours, yet shows some noteworthy similarities. Their research is concerned with proactive behavior of personal task management assistants, such as used in office environments. They do not directly rely upon AI Planning, but instead build upon the BDI (Belief-Desire-Intention) framework SPARK [158]. BDI agents are a popular approach to realize intelligent agents that act in complex, dynamic environments. Though the BDI framework was devel-

[5] *D*eutsche *F*orschungs*g*emeinschaft, engl.: German Research Foundation

oped independently of planning, there is a close relationship: Since BDI agents also rely upon hierarchical control structures, they are expressive enough to capture the (undecidable) HTN planning framework [141] that we base upon, too. Yorke-Smith et al.'s research is concerned with the development of proactive personal assistants that support knowledge workers in managing time commitments and performing tasks (such as routine tasks in an office environment). To achieve this, they characterise helpful proactive behavior by the assistant and extend the BDI model such that it can act according to this behavior. From a technology point-of-view, Yorke-Smith et al.'s work is based on the *CALO (Cognitive Assistant that Learns and Organizes)* system [1]. It includes *Towel* [127] as one of its underlying components, an intelligent todo list manager for office environments, which can represent, among others, the duration of tasks and dependencies among them. In contrast to us, they focus on providing *proactive* support, i. e., deciding on when and how to interact with a user. The only proactive behavior that we are concerned with is the (proactive) provision of explanations in case certain concepts are not sufficiently known to the user, see Section VI. We, on the other hand, focus on how the integration of various planning and user-interaction facilities supports in the execution of various tasks via the provision of a detailed, multimodal *step-by-step* instruction. These aspects are not covered by this work, which instead presents clearly arranged lists of task descriptions.

González et al. (2017) [22] introduce a planning-based architecture for humanoid robots to be used in the context of rehabilitation therapies. Their approach allows a therapist to define exercises that the robot carries out together with the patient. The proposed architecture relies on different planning layers. The top-most layer is used to define the therapy. It is formalized using HTN planning due to the natural hierarchy of the problem and solved by the planning system JSHOP2 [164]. The lower levels are realized using a non-hierarchical planning approach; they are realizing the exercises, i. e., controlling the robot. The robot is equipped with a camera to monitor whether exercises are executed correctly. In case the currently observed state differs from the expected one (and this causes the current plan to fail), then a new plan is generated relying on replanning from the current state. The system does not feature explanation capabilities based on AI technology, but instead it is able to give an introduction to the respective exercises. Their approach is therefore only loosely related to ours in that it relies on a hierarchical action model to support human users, coupled with repair facilities. A core difference is therefore that it is not a domain-independent approach for supporting in a variety of complex tasks. Consequently, it it not about the individual multimodal presentation of instructions.

Bernardini and Porayska-Pomsta (2013) [68] introduce a system that helps children with Autism Spectrum Conditions to acquire social communication skills. One of its similarities to our system is that is also maintains a user model; this model of the child is used to reflect its current cognitive and affective state to adapt its behavior to the child's reactions. Thus, their system is, as ours, both user- and situation-adaptive. A core difference to our system is that it uses a virtual agent to interact with the child, whereas our system shows instructions that correspond to the underlying plan. The agent's behavior is also based upon such plans, created by a POCL planner. Our approach, hybrid planning, extends POCL planning by means of a hierarchy so that the planning tasks can also be modeled in a hierarchical way and to improve explanations about presented instructions. Since Bernardini and Porayska-Pomsta's system is not about explaining how to solve a problem and thus does not show a sequence of instructions, it also comes without explanation capabilities as proposed in this report. Similar to our system, theirs also allows user input via touch gesture and receives sensory information. For instance, their system can also recognize gaze and gestures. In contrast to our system, theirs runs on a single device, so they are not concerned with modality arbitration to the same extent as our system is. Concerning planning execution, monitoring, and repair, there is a big difference between our systems: Bernardini and Porayska-Pomsta's system interleaves planning and plan execution, whereas our system derives the entire plan before the user starts executing it. That way, they can always adapt the current plan according to the current situation and the child's reaction.

Beetz et al. (2012) [76] develop techniques to allow autonomous service robots achieving "home chore task intelligence" to assist elderly people by duties of their daily life. These robots are controlled by so-called "cognition-enabled robot control programs" with learning, reasoning, and planning capabilities. Similar to us, they use a hierarchical planning approach. While Beetz et al. also aim at assisting human users, a key difference is that their aim is to control robots so they do the tasks in the adequate way. Our approach, in contrast, has one of its strengths in presenting a sequence of instructions to its user thereby taking his or her knowledge and the current situation into account.

Related to the last system, the one by Pollack (2002) [170] is an assistant for elderly people and people with cognitive impairments. Its purpose is to assist them in routines of their daily life so they can remain their independence for a longer time and therefore also stay in their own homes longer than without such an assistant.

For this, the assistant is conceptualized for running on an autonomous mobile robot. The routines that it is assisting by include, for example, eating and drinking, using the bathroom, managing medicine, and housekeeping. Thus, one of the system's central capabilities is to allow its user to add activities and to remind about forgotten ones. For such activities, the assistant features a model of time allowing tasks to have a duration and time points where they are supposed to start. As discussed in Section IX, such expansions are currently not implemented in our system. The underlying planning procedure is again a POCL system. The respective system also features some sort of plan repair Pollack (called *plan update*), which is an important feature of their assistant, because forgotten or changed routines can have an influence of the remainder of the plan.

A closely related endeavor is also pursued by Boger et al. (2005, 2006) [143, 135]. Their assistant assists people with dementia in daily routine tasks. While Pollack's approach focuses on complying to the user's schedule (i. e., tasks can be regarded atomic), Boger et al.'s approach focuses on assisting to *carry out* the respective tasks. For this, the system is equipped with a computer vision system to observe the user's currently performed task. The system ensures that no actions were forgotten or carried out in an incorrect order. If this is detected, the system prompts – using audio feedback – an adequate advice. As an example (and for empirical evaluations) they use the task of washing hands. It consists of various steps (to be performed in a useful order), such as turning water on and off, taking and using the soap, and removing the soap from the hands under running water. As underlying planning model, they use Markov Decision Processes (MDPs) and the generalization thereof to Partially Observable MDPs (POMDPs). The authors argue that they need such a formalism because the problems they are assisting by are inherently relying on observations (which are practically always uncertain). Further, also the effects of their system's actions are uncertain. Finally, they argue that they have to satisfy several different objective criteria that might even conflict with each other, thereby preventing goal achievement with certainty.

The SIADEX system provides decision support in the application area of crisis management [137]. The example application in which they have tested their system is forest fire fighting. They introduce a system architecture that features components for plan execution and monitoring (and, consequently, plan repair). It also comprises a component for displaying plans. Similar to our system, it also allows non-AI-experts to operate their system, i. e., technical staff. They do so by showing the respective plans via Microsoft Excel (as a chronogram) or Microsoft Project (as a Gantt chart). The underlying planning framework is HTN planning with temporal information (which is obviously particularly important in crisis management). In the respective setting, there is a large amount of data stemming from heterogeneous sources that is of importance for carrying out the respective tasks. For this, the architecture features a central knowledge base, in which the information is stored and processed in form of an ontology. The system does not feature components for explaining the plans, nor are advanced input and output facilities in the focus of the project.

Peintner et al. (2009) [112] introduce a task management tool for military environments. It is basically an intelligent, AI-enhanced, todo list tailored to the specific requirements for the military. The task assistant builds upon the before-mentioned *Towel* system [127], an intelligent todo list manager. Note that Towel also bases upon the before-mentioned BDI framework SPARK [158] and that it was also created in the context of the CALO project. The assistant allows to create and manage task lists, showing which tasks depend on each other and how much time is needed to carry them out. Tasks can be shared between multiple users, a feature that is also currently not explicitly discussed in our assistant. AI technology was primarily used for providing recommendations on how to modify task lists based on previous experiences (based on Machine Learning). One of the core differences to our approach is therefore the general problem setting: We enable to automatically solve a given problem (like the operation of a technical device) due to the underlying formal model of the world, whereas todo lists contain a list of tasks with only limited interaction between each other.

Tate et al. (2000) [178] introduce a prototype system to provide decision aid to support US Army small unit operations in urban terrain. The focus of their paper lies on giving an overview on how AI technology can be deployed in the entire application scenario, from domain knowledge elicitation to the actual plan execution. The main purpose of their approach and prototype is to provide the soldiers with information about the environment or battlespace. Thus, it is especially important for the (soldier-borne) user interface to give an overview of the current terrain and objectives (many illustrations are provided in the paper). As such, it is not concerned with a direct illustration of actions, nor does it support advanced input or output capabilities, but it is rather tailored to the special requirements of the soldiers. The system does not incorporate user knowledge, but it can be configured to specialize itself according to user roles. The underlying planning approach is a hierarchical one and, similar to our approach, causal links are used to explicitly represent causality. The system used for the prototype's various capabilities is O-PLAN [199]. The system can repair its plans in the light of unforeseen changes to the current situation. For the sake of the prototype, a simulation creates such changes; in the real system, sensors are responsible for this instead. Explanation capabilities are not incorporated.

Further assistance systems that directly interact with their users and base their recommendations upon AI planning are basically *all* mixed-initiative planning (MIP) systems. Since an overview about existing MIP systems is out of the focus of this report and would be worth an article on its own, we here only mention a few systems and approaches that are closest to our approach. We also want to mention that it is also not always perfectly clear whether a planning-based system can be regarded a MIP system. In particular in real-world applications, at least the problem itself needs to be defined by a system's user, so such systems could be referred to as a MIP system as well. For example, also the previously introduced systems by Tate et al. (2000) [178] and Fernández-Olivares et al. (2006) [137] could be referred to as a MIP system for this reason. At the time being, there does not exist a survey on current MIP systems; instead we refer to the work by Behnke et al. (2017) [12] for a discussion of some of the most recent approaches. In contrast to our approach (which focuses on the assistance during the execution of a task), the core functionality of MIP systems is the *generation* of plans and therefore do not have plan execution-related facilities such as plan monitoring or plan repair. Also the (user-specific) presentation of plans in a step-by-step fashion is not required by such systems and thus not incorporated. Further, the cooperative development of plans is often intended to be done by an expert (see examples below), whereas the deployment of our assistance system is also intended to be used by novices.

One of the best-known MIP systems is PASSAT (Plan-Authoring System based on Sketches, Advice, and Templates) by Myers et al. (2003) [163]. Similar to our approach, it relies on an HTN planning approach. Further, PASSAT is the only system mentioned in this overview that is, as our approach, completely domain-independent – only their example application in which they illustrate their approach is in a specific application domain. In fact, most of their paper is concerned with the domain-independent formalization and techniques for collaborative plan generation. In their application scenario they are, similar to Tate et al. (2000) [178], concerned with generating attack plans for military special operations groups; more specifically, they are concerned with a hostage rescue scenario. Since their system is a general MIP system, it is spending quiet some effort on their repair mechanisms. Because plans are developed prior to their execution, "repair" here means identifying and addressing deficiencies of plans that are not yet solutions rather than compensating for solution failures of complete and correct plans. As expected for a MIP system, it does not incorporate advanced input/output capabilities or plan explanations.

A further well-known MIP system is MAPGEN (Mixed-Initiative Activity Planning Generator) that was used by the operations staff for assisting in creating the daily activity plans for the Mars Exploration Rovers (MER) Spirit and Opportunity [145, 146, 32]. The planning language was a hierarchical one that also featured resources (such as energy) and temporal constraints. Because MAPGEN was used to *create* plans instead of also executing them, it does not make sense to be equipped with a plan repair mechanism. However, both related to plan repair and plan explanation, it was a critical issue to show the user in an easy-to-comprehend way any conflicts of the current plan with any constraints, such as energy levels. Any advanced input or output capabilities were out of the scope of the project. Individualization to a specific user was also not incorporated.

Lastly, we would like to mention that the technology presented in this report has already attracted attention for its application in an industrial setting. More specifically, we are currently involved in a project in which non-experienced users are assisted in a Do-It-Yourself home improvement project of their choice [5, 26], e. g., to build a keyrack, a bird house, or a cupboard. The objectives of this assistant are twofold: it provides instructions to the user on how to safely (work safety) and successfully complete the desired project, but it also aims at teaching the user how to use the electronic tools involved in the project, such that he or she will be able to use them without assistance in the future and thus become more self-reliant. Here, we employ the same architectural principles and many of the components presented in this report. We use a hierarchical planner to automatically derive appropriate instructions for the user that will lead to the completion of the respective project with those tools and materials the user has actually available. In addition to the pure sequential instructions, we have also show the user an abstraction of the current plan to guide him or her through the DIY project and to provide instant feedback on successful task-completion. Similarly, we use many of the dialog- and interaction-management principles. The major difference between the two systems is, however, that the DIY-assistant handles its media content based on ontology classification enabling us to find good-fitting content even if a perfect fitting one is not available. This application in a completely new environment emphasizes our claim that the presented methodology is independent of the concrete application at hand.

## XI Conclusion and Future Work

We presented an approach for realizing adaptive, individual, and trustworthy assistance systems for every-day life situations. We identified AI planning, probabilistic reasoning, dialog management, and interaction management as the key technologies in such *Companion*-Systems and proposed a flexible architecture that combines these technologies. Dialog management and interaction management adaptively and individually mediate between the user and the higher-level cognitive processes of the system provided by planning and reasoning. Transparency and therefore trustworthiness is achieved across multiple components by offering capabilities for explaining plans and relevant concepts. A common property of the involved technologies is the high degree of scalability achieved by using domain-independent, model-based approaches. That way implementing a system based on our architecture is straightforward: it only requires specifying the respective models for planning, dialog management, and so on (cf. Section IX) in addition to the required data, such as pictures or videos. The situation is similar on the sensor side: many components can process a multitude of information. For example, the dialog management component can modify its dialog strategies according to the user's emotional state. Therefore, sensing capabilities can be added transparently when their measurements concern variables already taken into account by existing user and environment models [48].

We implemented a prototype system based on the proposed architecture in the example domain of *setting up a complex home theater* and used it to illustrate the involved technologies. We gave pointers to previous and related work, including summaries for evaluations that were done for the respective system components and the prototype. In one of these studies we were particularly interested in the usefulness of our system and the acceptance of such assistance functionality by the test subjects. In summary, it was perceived very well, in particular by non-experts (cf. Section E).

We discussed possible directions of future work for each of the architecture's components. One of these directions, which is shared by several of the involved components, concerns the consistent consideration of uncertainty – in particular in the form of partial observability. Being able to handle partial observability allows, e. g., to deal with partial knowledge about the user or the current state of the environment. With so-called sensing actions, certain information can then be acquired by the system. This is also closely related to our currently pursued research direction of mixed-initiative planning [11, 12]. Here, several choices can be left to the user while the planner guarantees that for all these choices the plan still achieves all of the user's goals – no matter what decision he or she takes (cf. Section A). Further interesting directions in which our prototype can be extended involve the incorporation of plan and goal recognition, which would, e. g., allow the system to intervene when it detects that the user's currently deployed strategy will not achieve his or her goals or in a sub-optimal way (cf. Section B); or the realization of a dynamic configuration that combines the most promising components to the overall system (cf. Section B). Currently ongoing work incorporates an ontology to the knowledge base that stores and processes factual knowledge of the application domain. These currently developed concepts are deployed in another assistance system, which bases upon the technology described in this report and is developed together with an industry partner. It assists its users in a Do-It-Yourself (DIY) home improvement project [5, 26] (cf. also Section X for more details).

## Acknowledgements

This work was done within the Transregional Collaborative Research Center SFB/TRR 62 "*Companion*-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

We want to thank our former colleagues Gregor Bertrand, Peter Kurzok, Louisa Pragst, Bernd Schattenberg, and Bastian Seegebarth for their work for the demonstrated system. We also acknowledge Roland Barth and Matthias Rau for their contributions in producing a video [59] explaining the system.

## References

[1] S. International. *CALO: Cognitive Assistant that Learns and Organizes*. 2003–2009. URL: http://www.ai.sri.com/project/CALO.

[2] F. Honold. "Interaktionsmanagement und Modalitätsarbitrierung für eine adaptive und multimodale Mensch-Computer-Interaktion. Betrachtungen im Kontext des SFB/Transregio 62 *Eine Companion-Technologie für kognitive technische Systeme*". In Press. Dissertation. Ulm University, 2019.

[3] G. Behnke, D. Höller, and S. Biundo. "totSAT – Totally-Ordered Hierarchical Planning through SAT". In: *Proc. of the 32nd AAAI Conf. on Artificial Intelligence (AAAI 2018)*. AAAI Press, 2018, pp. 6110–6118.

[4] G. Behnke, D. Höller, and S. Biundo. "Tracking Branches in Trees – A Propositional Encoding for Solving Partially-Ordered HTN Planning Problems". In: *Proc. of the 30th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 2018)*. IEEE Press, 2018.

[5] G. Behnke, M. Schiller, M. Kraus, P. Bercher, M. Schmautz, M. Dorna, W. Minker, B. Glimm, and S. Biundo. "Instructing Novice Users on How to Use Tools in DIY Projects". In: *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence and the 23rd European Conf. on Artificial Intelligence (IJCAI-ECAI 2018)*. IJCAI, 2018, pp. 5805–5807. DOI: 10.24963/ijcai.2018/844.

[6] T. Chakraborti, S. Sreedharan, and S. Kambhampati. "Human-Aware Planning Revisited: A Tale of Three Models". In: *Proc. of the IJCAI/ECAI 2018 Workshop on EXplainable Artificial Intelligence (XAI)*. That paper was also published in the Proc. of the ICAPS 2018 Workshop on EXplainable AI Planning (XAIP). 2018, pp. 18–25.

[7] D. Höller, G. Behnke, P. Bercher, and S. Biundo. "Plan and Goal Recognition as HTN Planning". In: *Proc. of the 30th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 2018)*. IEEE, 2018.

[8] D. Höller, P. Bercher, G. Behnke, and S. Biundo. "A Generic Method to Guide HTN Progression Search with Classical Heuristics". In: *Proc. of the 28th Int. Conf. on Automated Planning and Scheduling (ICAPS 2018)*. AAAI Press, 2018, pp. 114–122.

[9] D. Höller, P. Bercher, G. Behnke, and S. Biundo. "HTN Plan Repair Using Unmodified Planning Systems". In: *Proc. of the First ICAPS Workshop on Hierarchical Planning*. 2018, pp. 26–30.

[10] S. K. Tathagata Chakrabroti Sarath Sreedharan. "Explicability versus Explanations in Human-Aware Planning". In: *Proc. of the 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012)*. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2018, pp. 2180–2182.

[11] G. Behnke, B. Leichtmann, P. Bercher, D. Höller, V. Nitsch, M. Baumann, and S. Biundo. "Help me make a dinner! Challenges when assisting humans in action planning". In: *Proc. of the 2nd Int. Conf. on Companion Technology (ICCT 2017)*. IEEE, 2017.

[12] G. Behnke, F. Nielsen, M. Schiller, P. Bercher, M. Kraus, B. Glimm, W. Minker, and S. Biundo. "SLOTH – the Interactive Workout Planner". In: *Proc. of the 2nd Int. Conf. on Companion Technology (ICCT 2017)*. IEEE, 2017.

[13] G. Behnke, F. Nielsen, M. Schiller, D. Ponomaryov, Pascal Bercher, B. Glimm, W. Minker, and S. Biundo. "To Plan for the User Is to Plan With the User – Integrating User Interaction Into the Planning Process". In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 7, pp. 123–144. DOI: 10.1007/978-3-319-43665-4_7.

[14] P. Bercher, G. Behnke, D. Höller, and S. Biundo. "An Admissible HTN Planning Heuristic". In: *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence (IJCAI 2017)*. IJCAI, 2017, pp. 480–488. DOI: 10.24963/ijcai.2017/68.

[15] P. Bercher, D. Höller, G. Behnke, and S. Biundo. "User-Centered Planning". In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 5, pp. 79–100. DOI: 10.1007/978-3-319-43665-4_5.

[16] P. Bercher, F. Richter, T. Hörnle, T. Geier, D. Höller, G. Behnke, F. Nielsen, F. Honold, F. Schüssel, S. Reuter, W. Minker, M. Weber, K. Dietmayer, and S. Biundo. "Advanced User Assistance for Setting Up a Home Theater". In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 24, pp. 485–492. DOI: 10.1007/978-3-319-43665-4_24.

[17] S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. DOI: 10.1007/978-3-319-43665-4.

[18] N. Bubalo, F. Schüssel, F. Honold, M. Weber, and A. Huckauf. "Interaction History in Adaptive Multimodal Interaction". In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 12, pp. 231–252. DOI: 10.1007/978-3-319-43665-4_12.

[19] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. "Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy". In: *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence (IJCAI 2017)*. IJCAI, 2017, pp. 156–163. DOI: 10.24963/ijcai.2017/23.

[20] M. Fox, D. Long, and D. Magazzeni. “Explainable Planning”. In: *Proc. of the IJCAI-17 Workshop on Explainable AI (XAI 2017)*. 2017, pp. 24–30.

[21] T. Geier and S. Biundo. “Multi-Level Knowledge Processing in Cognitive Technical Systems”. In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 2, pp. 17–36. DOI: 10.1007/978-3-319-43665-4_2.

[22] J. C. González, J. C. Pulido, and F. Fernández. “A three-layer planning architecture for the autonomous control of rehabilitation therapies based on social robots”. In: *Cognitive Systems Research* 43.Supplement C (2017), pp. 232–249. DOI: 10.1016/j.cogsys.2016.09.003.

[23] T. Hörnle, M. Tornow, F. Honold, R. Schwegler, R. Heinemann, S. Biundo, and A. Wendemuth. “*Companion*-Systems: A Reference Architecture”. In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 22, pp. 449–470. DOI: 10.1007/978-3-319-43665-4_22.

[24] F. Nielsen and W. Minker. “Assistive and Adaptive Dialog Management”. In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 9, pp. 167–186. DOI: 10.1007/978-3-319-43665-4_9.

[25] F. Richter and S. Biundo. “Addressing Uncertainty in Hierarchical User-Centered Planning”. In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 6, pp. 101–122. DOI: 10.1007/978-3-319-43665-4_6.

[26] M. Schiller, G. Behnke, M. Schmautz, P. Bercher, M. Kraus, M. Dorna, W. Minker, B. Glimm, and S. Biundo. “A Paradigm for Coupling Procedural and Conceptual Knowledge in Companion Systems”. In: *Proc. of the 2nd Int. Conf. on Companion Technology (ICCT 2017)*. IEEE, 2017. DOI: 10.1109/COMPANION.2017.8287072.

[27] F. Schüssel, F. Honold, N. Bubalo, M. Weber, and A. Huckauf. “Management of Multimodal User Interaction in *Companion*-Systems”. In: S. Biundo and A. Wendemuth. *Companion Technology – A Paradigm Shift in Human-Technology Interaction*. Cognitive Technologies. Springer, 2017. Chap. 10, pp. 187–208. DOI: 10.1007/978-3-319-43665-4_10.

[28] S. Sengupta, T. Chakraborti, S. Sreedharan, S. G. Vadlamudi, and S. Kambhampati. “RADAR – A Proactive Decision Support System for Human-in-the-Loop Planning”. In: *The 2017 AAAI Fall Symposium Series: Technical Reports*. AAAI Press, 2017, pp. 269–276.

[29] S. Sreedharan, T. Chakraborti, and S. Kambhampati. “Balancing Explicability and Explanation in Human-Aware Planning”. In: *The 2017 AAAI Fall Symposium Series: Technical Reports*. FS-17-01. AAAI Press, 2017, pp. 61–68.

[30] R. Alford, G. Behnke, D. Höller, P. Bercher, S. Biundo, and D. Aha. “Bound to Plan: Exploiting Classical Heuristics via Automatic Translations of Tail-Recursive HTN Problems”. In: *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS 2016)*. AAAI Press, 2016, pp. 20–28.

[31] G. Behnke, D. Höller, P. Bercher, and S. Biundo. “Change the Plan – How hard can that be?” In: *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS 2016)*. AAAI Press, 2016, pp. 38–46.

[32] P. Bercher and D. Höller. “Interview with David E. Smith”. In: *Künstliche Intelligenz* 30.1 (2016). Special Issue on Companion Technologies, pp. 101–105. DOI: 10.1007/s13218-015-0403-y.

[33] P. Bercher, D. Höller, G. Behnke, and S. Biundo. “More than a Name? On Implications of Preconditions and Effects of Compound HTN Planning Tasks”. In: *Proc. of the 22nd European Conf. on Artificial Intelligence (ECAI 2016)*. IOS Press, 2016, pp. 225–233. DOI: 10.3233/978-1-61499-672-9-225.

[34] S. Biundo, D. Höller, B. Schattenberg, and P. Bercher. “Companion-Technology: An Overview”. In: *Künstliche Intelligenz* 30.1 (2016). Special Issue on Companion Technologies, pp. 11–20. DOI: 10.1007/s13218-015-0419-3.

[35] S. Biundo and A. Wendemuth. “*Companion*-Technology for Cognitive Technical Systems”. In: *Künstliche Intelligenz* 30.1 (2016). Special Issue on Companion Technologies, pp. 71–75. DOI: 10.1007/s13218-015-0414-8.

[36] T. Geier, M. Glodek, G. Layher, H. Neumann, S. Biundo, and G. Palm. "On stacking probabilistic temporal models with bidirectional information flow". In: *Proc. of the 8th Int. Conf. on Probabilistic Graphical Models.* JMLR Workshop and Conf. Proceedings, 2016, pp. 195–206.

[37] D. Höller, G. Behnke, P. Bercher, and S. Biundo. "Assessing the Expressivity of Planning Formalisms through the Comparison to Formal Languages". In: *Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS 2016).* AAAI Press, 2016, pp. 158–165.

[38] J. B. Lyons, K. S. Koltai, N. T. Ho, W. B. Johnson, D. E. Smith, and R. J. Shively. "Engineering Trust in Complex Automated Systems". In: *Ergonomics in Design: The Quarterly of Human Factors Applications* 24.1 (2016), pp. 13–17. DOI: 10.1177/1064804615611272.

[39] F. Nothdurft, F. Honold, and W. Minker. "Temporal and Spatial Design of Explanations in a Multimodal System". In: *Human-Computer Interaction. Interaction Platforms and Techniques: 18th Int. Conf., HCI International 2016. Proceedings, Part II.* Springer International Publishing, 2016, pp. 201–210. DOI: 10.1007/978-3-319-39516-6_19.

[40] F. Schüssel, F. Honold, N. Bubalo, and M. Weber. "Increasing Robustness of Multimodal Interaction via Individual Interaction Histories". In: *Proc. of the Workshop on Multimodal Analyses Enabling Artificial Agents in Human-Machine Interaction.* MA3HMI '16. Tokyo, Japan: ACM, 2016, pp. 56–63. DOI: 10.1145/3011263.3011273.

[41] R. Alford, P. Bercher, and D. Aha. "Tight Bounds for HTN Planning". In: *Proc. of the 25th Int. Conf. on Automated Planning and Scheduling (ICAPS 2015).* AAAI Press, 2015, pp. 7–15.

[42] R. Alford, P. Bercher, and D. Aha. "Tight Bounds for HTN planning with Task Insertion". In: *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence (IJCAI 2015).* AAAI Press, 2015, pp. 1502–1508.

[43] G. Behnke, D. Höller, and S. Biundo. "On the Complexity of HTN Plan Verification and its Implications for Plan Recognition". In: *Proc. of the 25th Int. Conf. on Automated Planning and Scheduling (ICAPS 2015).* AAAI Press, 2015, pp. 25–33.

[44] G. Behnke, D. Ponomaryov, M. Schiller, P. Bercher, F. Nothdurft, B. Glimm, and S. Biundo. "Coherence Across Components in Cognitive Systems – One Ontology to Rule Them All". In: *Proc. of the 25th Int. Joint Conf. on Artificial Intelligence (IJCAI 2015).* AAAI Press, 2015, pp. 1442–1449.

[45] P. Bercher, F. Richter, T. Hörnle, T. Geier, D. Höller, G. Behnke, F. Nothdurft, F. Honold, W. Minker, M. Weber, and S. Biundo. "A Planning-based Assistance System for Setting Up a Home Theater". In: *Proc. of the 29th AAAI Conf. on Artificial Intelligence (AAAI 2015).* AAAI Press, 2015, pp. 4264–4265.

[46] M. C. Caschera, A. D'Ulizia, F. Ferri, and P. Grifoni. "On the Move to Meaningful Internet Systems: OTM 2015 Workshops". In: ed. by I. Ciuciu, H. Panetto, C. Debruyne, A. Aubry, P. Bollen, R. Valencia-García, A. Mishra, A. Fensel, and F. Ferri. Springer International Publishing, 2015. Chap. Multimodal Systems: An Excursus of the Main Research Questions, pp. 546–558. ISBN: 978-3-319-26138-6. DOI: 10.1007/978-3-31926138-6_59.

[47] I. Georgievski and M. Aiello. "HTN planning: Overview, comparison, and beyond". In: *Artificial Intelligence* 222.0 (2015), pp. 124–156. DOI: 10.1016/j.artint.2015.02.002.

[48] M. Glodek, F. Honold, T. Geier, G. Krell, F. Nothdurft, S. Reuter, F. Schüssel, T. Hörnle, K. Dietmayer, W. Minker, S. Biundo, M. Weber, G. Palm, and F. Schwenker. "Fusion paradigms in cognitive technical systems for human-computer interaction". In: *Neurocomputing* 161.0 (2015), pp. 17–37. DOI: 10.1016/j.neucom.2015.01.076.

[49] T. Hörnle and M. Tornow. "Reference Architecture Approach for Companion-Systems". In: *Proc. of the 1st Int. Symposium on Companion Technology (ISCT 2015).* 2015, pp. 155–160.

[50] S. Kambhampati and K. Talamadupula. *Human-in-the-Loop Planning and Decision Support.* AAAI 2015 Tutorial. 2015. URL: http://rakaposhi.eas.asu.edu/hilp-tutorial/.

[51] F. Nothdurft, G. Behnke, P. Bercher, S. Biundo, and W. Minker. "The Interplay of User-Centered Dialog Systems and AI Planning". In: *Proc. of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2015).* Association for Computational Linguistics, 2015, pp. 344–353.

[52] H. Olmedo, D. Escudero, and V. Cardeñoso. "Multimodal Interaction with Virtual Worlds XMMVR: eXtensible Language for Multimodal Interaction with Virtual Reality Worlds". In: *Journal on Multimodal User Interfaces* 9.3 (2015), pp. 153–172. DOI: 10.1007/s12193-015-0176-5.

[53] F. Richter, T. Geier, and S. Biundo. "Believing in POMDPs". In: *Proc. of the 1st Int. Symposium on Companion Technology (ISCT 2015)*. 2015, pp. 25–30.

[54] R. Alford, V. Shivashankar, U. Kuter, and D. Nau. "On the Feasibility of Planning Graph Style Heuristics for HTN Planning". In: *Proc. of the 24th Int. Conf. on Automated Planning and Scheduling (ICAPS 2014)*. AAAI Press, 2014, pp. 2–10.

[55] P. Bercher, S. Biundo, T. Geier, T. Hoernle, F. Nothdurft, F. Richter, and B. Schattenberg. "Plan, Repair, Execute, Explain – How Planning Helps to Assemble your Home Theater". In: *Proc. of the 24th Int. Conf. on Automated Planning and Scheduling (ICAPS 2014)*. AAAI Press, 2014, pp. 386–394.

[56] P. Bercher, S. Keen, and S. Biundo. "Hybrid Planning Heuristics Based on Task Decomposition Graphs". In: *Proc. of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*. AAAI Press, 2014, pp. 35–43.

[57] M. Glodek, T. Geier, S. Biundo, and G. Palm. "A layered architecture for probabilistic complex pattern recognition to detect user preferences". In: *Biologically Inspired Cognitive Architectures* 9 (July 2014). Neural-Symbolic Networks for Cognitive Capacities, pp. 46–56.

[58] D. Höller, P. Bercher, F. Richter, M. Schiller, T. Geier, and S. Biundo. "Finding User-friendly Linearizations of Partially Ordered Plans". In: *28th PuK Workshop "Planen, Scheduling und Konfigurieren, Entwerfen" (PuK 2014)*. 2014.

[59] F. Honold, P. Bercher, F. Richter, F. Nothdurft, T. Geier, R. Barth, T. Hörnle, F. Schüssel, S. Reuter, M. Rau, G. Bertrand, B. Seegebarth, P. Kurzok, B. Schattenberg, W. Minker, M. Weber, and S. Biundo. "Companion-Technology: Towards User- and Situation-Adaptive Functionality of Technical Systems". In: *Proc. of the 10th Int. Conf. on Intelligent Environments (IE 2014)*. IEEE, 2014, pp. 378–381. DOI: 10.1109/IE.2014.60.

[60] F. Honold, F. Schüssel, and M. Weber. "The Automated Interplay of Multimodal Fission and Fusion in Adaptive HCI". In: *Proc. of the 10th Int. Conf. on Intelligent Environments (IE 2014)*. IEEE, 2014, pp. 170–177. DOI: 10.1109/IE.2014.32.

[61] F. Nothdurft, F. Honold, K. Zablotskaya, A. Diab, and W. Minker. "Application of Verbal Intelligence in Dialog Systems for Multimodal Interaction". In: *Proc. of the 10th Int. Conf. on Intelligent Environments (IE 2014)*. IEEE, 2014, pp. 361–364. DOI: 10.1109/IE.2014.59.

[62] F. Nothdurft, F. Richter, and W. Minker. "Probabilistic Human-Computer Trust Handling". In: *Proc. of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2014)*. Association for Computational Linguistics, 2014, pp. 51–59.

[63] S. Reuter. "Multi-Object Tracking Using Random Finite Sets". PhD thesis. Ulm University, 2014. DOI: 10.18725/OPARU-3204.

[64] S. Reuter, B. Vo, B. Vo, and K. Dietmayer. "The Labeled Multi-Bernoulli Filter". In: *IEEE Transactions on Signal Processing* 62.12 (2014), pp. 3246–3260. DOI: 10.1109/TSP.2014.2323064.

[65] F. M. Schaub. "Dynamic privacy adaptation in ubiquitous computing". PhD thesis. Ulm University. Faculty of Engineering and Computer Science, 2014. DOI: 10.18725/OPARU-3188.

[66] F. Schüssel, F. Honold, M. Weber, M. Schmidt, N. Bubalo, and A. Huckauf. "Multimodal Interaction History and its use in Error Detection and Recovery". In: *Proc. of the 16th ACM Int. Conf. on Multimodal Interaction (ICMI 2014)*. ACM, 2014, pp. 164–171. DOI: 10.1145/2663204.2663255.

[67] P. Bercher, T. Geier, and S. Biundo. "Using State-Based Planning Heuristics for Partial-Order Causal-Link Planning". In: *Proc. of the 36th German Conf. on Artificial Intelligence (KI 2013)*. Springer, 2013, pp. 1–12. DOI: 10.1007/978-3-642-40942-4_1.

[68] S. Bernardini and K. Porayska-Pomsta. "Planning-Based Social Partners for Children with Autism". In: *Proc. of the 23rd Int. Conf. on Automated Planning and Scheduling (ICAPS 2013)*. AAAI Press, 2013, pp. 362–370.

[69] F. Honold, F. Schüssel, M. Weber, F. Nothdurft, G. Bertrand, and W. Minker. “Context Models for Adaptive Dialogs and Multimodal Interaction”. In: *Proc. of the 9th Int. Conf. on Intelligent Environments (IE 2013)*. IEEE, 2013, pp. 57–64. DOI: 10.1109/IE.2013.54.

[70] M. Knappmeyer, S. L. Kiani, E. S. Reetz, N. Baker, and R. Tonjes. “Survey of Context Provisioning Middleware”. In: *Communications Surveys Tutorials, IEEE* 15.3 (2013), pp. 1492–1519. DOI: 10.1109/SURV.2013.010413.00207.

[71] S. Reuter, B. Wilking, J. Wiest, M. Munz, and K. Dietmayer. “Real-Time Multi-Object Tracking using Random Finite Sets”. In: *IEEE Transactions on Aerospace and Electronic Systems* 49.4 (2013), pp. 2666–2678. DOI: 10.1109/TAES.2014.6619956.

[72] F. Schüssel, F. Honold, and M. Weber. “Using the Transferable Belief Model for Multimodal Input Fusion in Companion Systems”. In: *Multimodal Pattern Recognition of Social Signals in HCI*. Springer, 2013, pp. 100–115. DOI: 10.1007/978-3-642-37081-6_12.

[73] V. Shivashankar, R. Alford, U. Kuter, and D. Nau. “The GoDeL Planning System: A More Perfect Union of Domain-Independent and Hierarchical Planning”. In: *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI 2013)*. AAAI Press, 2013, pp. 2380–2386.

[74] B.-T. Vo and B.-N. Vo. “Labeled Random Finite Sets and Multi-Object Conjugate Priors”. In: *IEEE Transactions on Signal Processing* 61.13 (2013), pp. 3460–3475. DOI: 10.1109/TSP.2013.225982.

[75] A. Zaguia, A. Wahbi, C. Tadj, and A. Ramdane-Cherif. “Multimodal Fission For Interaction Architecture”. In: *Journal of Emerging Trends in Computing and Information Sciences* 4.1 (Jan. 2013), pp. 152–167.

[76] M. Beetz, D. Jain, L. Mosenlechner, M. Tenorth, L. Kunze, N. Blodow, and D. Pangercic. “Cognition-Enabled Autonomous Robot Control for the Realization of Home Chore Task Intelligence”. In: *Proc. of the IEEE* 100.8 (2012), pp. 2454–2471. DOI: 10.1109/JPROC.2012.220055.

[77] G. Bertrand, F. Nothdurft, and W. Minker. ““What Do You Want to Do Next?” Providing the User with More Freedom in Adaptive Spoken Dialogue Systems”. In: *Proc. of the 8th Int. Conf. on Intelligent Environments (IE 2012)*. IEEE, 2012, pp. 290–296.

[78] M. Bodell, D. Dahl, I. Kliche, J. Larson, B. Porter, D. Raggett, T. Raman, B. H. Rodriguez, M. Selvaraj, R. Tumulur, A. Wahbe, P. Wiechno, and M. Yudkowsky. *Multimodal Architecture and Interfaces*. W3C Recommendation. W3C, Oct. 2012.

[79] B. Dumas, B. Signer, and D. Lalanne. “Fusion in Multimodal Interactive Systems: An HMM-based Algorithm for User-induced Adaptation”. In: *Proc. of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2012)*. ACM, 2012, pp. 15–24. DOI: 10.1145/2305484.2305490.

[80] M. Elkawkagy, P. Bercher, B. Schattenberg, and S. Biundo. “Improving Hierarchical Planning Performance by the Use of Landmarks”. In: *Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI 2012)*. AAAI Press, 2012, pp. 1763–1769.

[81] T. Geier, S. Reuter, K. Dietmayer, and S. Biundo. “Goal-Based Person Tracking Using a First-Order Probabilistic Model”. In: *Proc. of the Ninth UAI Bayesian Modeling Applications Workshop (UAI-AW 2012)*. 2012.

[82] T. Geier, S. Reuter, K. Dietmayer, and S. Biundo. “Track-Person Association Using a First-Order Probabilistic Model”. In: *Proc. of the 24th Int. Conf. on Tools with Artificial Intelligence (ICTAI 2012)*. IEEE, 2012, pp. 844–851.

[83] F. Honold, F. Schüssel, and M. Weber. “Adaptive Probabilistic Fission for Multimodal Systems”. In: *Proc. of the 24th Australian Computer-Human Interaction Conf. (OzCHI 2012)*. ACM, 2012, pp. 222–231. DOI: 10.1145/2414536.2414575.

[84] F. Müller, C. Späth, T. Geier, and S. Biundo. “Exploiting Expert Knowledge in Factored POMDPs”. In: *Proc. of the 20th European Conf. on Artificial Intelligence (ECAI 2012)*. IOS Press, Aug. 2012, pp. 606–611. DOI: 10.3233/978-1-61499-098-7-606.

[85] F. Nothdurft and W. Minker. “Using Multimodal Resources for Explanation Approaches in Technical Systems”. In: *Proc. of the 8th Conf. on Int. Language Resources and Evaluation (LREC 2012)*. European Language Resources Association (ELRA), 2012, pp. 411–415.

[86] F. Schüssel, F. Honold, and M. Weber. “Influencing factors on multimodal interaction during selection tasks”. In: *Journal on Multimodal User Interfaces* (2012), pp. 1–12. DOI: 10.1007/s12193-012-0117-5.

[87] B. Seegebarth, F. Müller, B. Schattenberg, and S. Biundo. “Making Hybrid Plans More Clear to Human Users – A Formal Approach for Generating Sound Explanations”. In: *Proc. of the 22nd Int. Conf. on Automated Planning and Scheduling (ICAPS 2012)*. AAAI Press, 2012, pp. 225–233.

[88] N. Yorke-Smith, S. Saadati, K. L. Myers, and D. N. Morley. “The Design of a Proactive Personal Agent for Task Management”. In: *International Journal on Artificial Intelligence Tools* 21.1 (2012), pp. 1250004-1–1250004-30.

[89] S. Biundo, P. Bercher, T. Geier, F. Müller, and B. Schattenberg. “Advanced user assistance based on AI planning”. In: *Cognitive Systems Research* 12.3-4 (Apr. 2011). Special Issue on Complex Cognition, pp. 219–236. DOI: 10.1016/j.cogsys.2010.12.005.

[90] D. Costa and C. Duarte. “Adapting multimodal fission to user’s abilities”. In: *Proc. of the 6th Int. Conf. on Universal Access in Human-Computer Interaction: Design for all and eInclusion – Volume Part I (UAHCI 2011)*. Springer-Verlag, 2011, pp. 347–356. DOI: 10.1007/978-3-642-21672-5_38.

[91] T. Geier and P. Bercher. “On the Decidability of HTN Planning with Task Insertion”. In: *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011)*. AAAI Press, 2011, pp. 1955–1961.

[92] T. Geier and S. Biundo. “Approximate Online Inference for Dynamic Markov Logic Networks”. In: *Proc. of the 23rd Int. Conf. on Tools with Artificial Intelligence (ICTAI 2011)*. IEEE, 2011, pp. 764–768. DOI: 10.1109/ICTAI.2011.120.

[93] M. D. Hina, C. Tadj, A. Ramdane-Cherif, and N. Levy. “A Multi-Agent based Multimodal System Adaptive to the User’s Interaction Context”. In: *Multi-Agent Systems – Modeling, Interactions, Simulations and Case Studies*. InTech, 2011. Chap. 2, pp. 29–56.

[94] F. Honold, M. Poguntke, F. Schüssel, and M. Weber. “Adaptive Dialogue Management and UIDL-based Interactive Applications”. In: *Proc. of the Int. Workshop on Software Support for User Interface Description Language (UIDL 2011)*. Thales Research and Technology France, 2011.

[95] D. Jain. “Knowledge engineering with markov logic networks: A review”. In: *Evolving Knowledge in Theory and Applications* 16 (2011).

[96] G. Pruvost, T. Heinroth, Y. Bellik, and W. Minker. “User Interaction Adaptation within Ambient Environments”. In: *Next Generation Intelligent Environments*. Springer New York, 2011, pp. 153–194. DOI: 10.1007/978-1-4614-1299-1_5.

[97] S. Reuter and K. Dietmayer. “Pedestrian Tracking Using Random Finite Sets”. In: *Proc. of the 14th Int. Conf. on Information Fusion (FUSION 2011)*. IEEE, 2011, pp. 1–8.

[98] S. Sohrabi, orge A. Baier, and S. A. McIlraith. “Preferred Explanations: Theory and Generation via Planning”. In: *Proc. of the 25th AAAI Conf. on Artificial Intelligence (AAAI 2011)*. AAAI Press, 2011, pp. 261–267.

[99] M. Buss and M. Beetz. “CoTeSys – Cognition for Technical Systems”. In: *Künstliche Intelligenz* 24 (2010), pp. 323–327. DOI: 10.1007/s13218-010-0061-z.

[100] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel. “Coming up With Good Excuses: What to do When no Plan Can be Found”. In: *Proc. of the 20th Int. Conf. on Automated Planning and Scheduling (ICAPS 2010)*. AAAI Press, 2010, pp. 81–88.

[101] R. Helaoui, M. Niepert, and H. Stuckenschmidt. “A Statistical-Relational Activity Recognition Framework for Ambient Assisted Living Systems”. In: *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010)*. Springer, 2010, pp. 247–254. DOI: 10.1007/978-3-642-13268-1_34.

[102] A. Kembhavi, T. Yeh, and L. S. Davis. “Why Did the Person Cross the Road (There)? Scene Understanding Using Probabilistic Logic Models and Common Sense Reasoning”. In: *Computer Vision ECCV 2010*. Vol. 6312. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2010, pp. 693–706. DOI: 10.1007/978-3-642-15552-9_50.

[103] F. Nothdurft, G. Bertrand, T. Heinroth, and W. Minker. “GEEDI – Guards for Emotional and Explanatory DIalogues”. In: *Proc. of the 6th Int. Conf. on Intelligent Environments (IE 2010)*. IEEE, 2010, pp. 90–95. DOI: 10.1109/IE.2010.24.

[104] H. J. Ritter. “Cognitive Interaction Technology – Goals and Perspectives of Excellence Cluster CITEC”. In: *Künstliche Intelligenz* 24.4 (2010), pp. 319–322. DOI: 10.1007/s13218-010-0063-x.

[105] A. Sadilek and H. Kautz. "Recognizing Multi-Agent Activities from GPS Data". In: *Twenty-Fourth AAAI Conf. on Artificial Intelligence.* AAAI Press, 2010, pp. 1134–1139.

[106] M. Schröder. "The SEMAINE API: Towards a Standards-Based Framework for Building Emotion-Oriented Systems". In: *Advances in Human-Computer Interaction* 2010 (2010), 319406:1–319406:21. DOI: 10 . 1155 / 2010 / 319406.

[107] B. Dumas, D. Lalanne, and S. Oviatt. "Multimodal Interfaces: A Survey of Principles, Models and Frameworks". In: *Human Machine Interaction – Research Results of the MMI Program.* Springer-Verlag, 2009. Chap. 1, pp. 3–26. DOI: 10.1007/978-3-642-00437-7_1.

[108] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles And Techniques.* The MIT Press, 2009.

[109] D. Lalanne, L. Nigay, P. Palanque, P. Robinson, J. Vanderdonckt, and J.-F. Ladry. "Fusion engines for multimodal input: a survey". In: *Proc. of the 2009 Int. Conf. on Multimodal Interfaces (ICMI-MLMI 2009).* ACM, 2009, pp. 153–160. DOI: 10.1145/1647314.1647343.

[110] B. Y. Lim, A. K. Dey, and D. Avrahami. "Why and why not explanations improve the intelligibility of context-aware intelligent systems". In: *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI 2009).* ACM, 2009, pp. 2119–2128. DOI: 10.1145/1518701.1519023.

[111] G. de Melo, F. Honold, M. Weber, M. Poguntke, and A. Berton. "Towards a Flexible UI Model for Automotive Human-machine Interaction". In: *Proc. of the 1st Int. Conf. on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2009).* ACM, 2009, pp. 47–50. DOI: 10.1145/1620509.1620518.

[112] B. Peintner, J. Dinger, A. Rodriguez, and K. Myers. "Task Assistant: Personalized Task Management for Military Environments". In: *Proc. of the 21st Innovative Applications of Artificial Intelligence Conf.* AAAI Press, 2009, pp. 128–134.

[113] B. Schattenberg. "Hybrid Planning & Scheduling". PhD thesis. University of Ulm, Germany, 2009. DOI: 10.18725/OPARU-1045.

[114] B.-T. Vo, B.-N. Vo, and A. Cantoni. "The Cardinality Balanced Multi-Target Multi-Bernoulli Filter and Its Implementations". In: *IEEE Transactions on Signal Processing* 57.2 (Feb. 2009), pp. 409–423. DOI: 10.1109/TSP.2008.2007924.

[115] N. Yorke-Smith, S. Saadati, K. L. Myers, and D. N. Morley. "Like an Intuitive and Courteous Butler: A Proactive Personal Agent for Task Management". In: *Proc. of the 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009).* International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2009, pp. 337–344.

[116] J. Bidot, S. Biundo, and B. Schattenberg. "Plan Repair in Hybrid Planning". In: *Proc. of the 31st German Conf. on Artificial Intelligence (KI 2008).* Springer, 2008, pp. 169–176. DOI: 10 . 1007/978-3-540-85845-4_21.

[117] A. Glass, D. L. McGuinness, and M. Wolverton. "Toward establishing trust in adaptive agents". In: *Proc. of the 13th Int. Conf. on Intelligent User Interfaces (IUI 2008).* ACM, 2008, pp. 227–236. DOI: 10.1145/1378773.1378804.

[118] N. Lin, U. Kuter, and E. Sirin. "Web Service Composition with User Preferences". In: *Proc. of the 5th European Semantic Web Conf. (ESWC 2008).* Springer, 2008, pp. 629–643. DOI: 10 . 1007/978-3-540-68234-9_46.

[119] B. Marthi, S. J. Russell, and J. Wolfe. "Angelic Hierarchical Planning: Optimal and Online Algorithms". In: *Proc. of the 18th Int. Conf. on Automated Planning and Scheduling (ICAPS 2008).* AAAI Press, 2008, pp. 222–231.

[120] R. de Salvo Braz, E. Amir, and D. Roth. "A Survey of First-Order Probabilistic Models". In: *Innovations in Bayesian Networks.* Studies in Computational Intelligence. Springer, 2008, pp. 289–317. DOI: 10.1007/978-3-540-85066-3_12.

[121] S. D. Tran and L. S. Davis. "Event Modeling and Recognition Using Markov Logic Networks". In: *Computer Vision–ECCV 2008* (2008), pp. 610–623. DOI: 10.1007/978-3-540-88688-4_45.

[122] M. J. Wainwright and M. I. Jordan. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1-2 (2008), pp. 1–305.

[123] J. Wang and P. Domingos. "Hybrid Markov Logic Networks". In: *Proc. of the 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008).* AAAI Press, 2008, pp. 1106–1111.

[124] N. F. Ayan, U. Kuter, F. Yaman, and R. P. Goldman. "HOTRiDE: Hierarchical Ordered Task Replanning in Dynamic Environments". In: *ICAPS Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices for Planning in Execution.* Providence, Rhode Island, USA, Sept. 2007.

[125] R. D. S. Braz, E. Amir, and D. Roth. "Lifted First-Order Probabilistic Inference". In: *Introduction to Statistical Relational Learning*. MIT Press, 2007. Chap. 15, pp. 433–452.

[126] P. Brusilovsky and E. Millán. "User models for adaptive hypermedia and adaptive educational systems". In: *The adaptive web*. Springer-Verlag, 2007, pp. 3–53. DOI: 10.1007/978-3-540-72079-9_1.

[127] K. Conley and J. Carpenter. "Towel: Towards an Intelligent To-Do List". In: *Proc. of the AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*. 2007, pp. 26–32.

[128] A. J. Fernandez, T. Hortala-Gonzalez, F. Saenz-Perez, and R. Del Vado-Virseda. "Constraint Functional Logic Programming over Finite Domains". In: *Theory and Practice of Logic Programming* 7.5 (Sept. 2007), pp. 537–582. DOI: 10.1017/S1471068406002924.

[129] R. Mahler. "PHD filters of higher order in target number". In: *IEEE Transactions on Aerospace and Electronic Systems* 43.4 (Oct. 2007), pp. 1523–1543. DOI: 10.1109/TAES.2007.4441756.

[130] R. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House Inc., Norwood, 2007.

[131] K. Myers, P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe. "An Intelligent Personal Assistant for Task and Time Management". In: *AI Magazine* 28.2 (2007), pp. 47–61.

[132] S. Trac, Y. Puzis, and G. Sutcliffe. "An Interactive Derivation Viewer". In: *Electronic Notes in Theoretical Computer Science* 174.2 (May 2007), pp. 109–123. DOI: 10.1016/j.entcs.2006.09.025.

[133] I. Warfield, C. Hogg, S. Lee-Urban, and H. Muñoz-Avila. "Adaptation of Hierarchical Task Network Plans". In: *Proc. of the 20th Int. Florida Artificial Intelligence Research Society Conf. (FLAIRS 2007)*. AAAI Press, 2007, pp. 429–434.

[134] J. D. Williams and S. Young. "Partially Observable Markov Decision Processes for Spoken Dialog Systems". In: *Computer Speech and Language* 21.2 (Apr. 2007), pp. 393–422.

[135] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis. "A planning system based on Markov decision processes to guide people with dementia through activities of daily living". In: *IEEE Transactions on Information Technology in Biomedicine* 10.2 (2006), pp. 323–333. DOI: 10.1109/TITB.2006.864480.

[136] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. "A Survey of Autonomic Communications". In: *ACM Transactions on Autonomous and Adaptive Systems* 1.2 (Dec. 2006), pp. 223–259. DOI: 10.1145/1186778.1186782.

[137] J. Fernández-Olivares, L. Castillo, Ó. García-Pérez, and F. Palao. "Bringing Users and Planning Technology Together. Experiences in SIADEX". In: *Proc. of the 16th Int. Conf. on Automated Planning and Scheduling (ICAPS 2006)*. AAAI Press, 2006, pp. 11–20.

[138] G. Herzog and N. Reithinger. "The SmartKom Architecture: A Framework for Multimodal Dialogue Systems". In: *SmartKom: Foundations of Multimodal Dialogue Systems*. Cognitive Technologies. Springer, 2006, pp. 55–70. DOI: 10.1007/3-540-36678-4_4.

[139] M. Richardson and P. Domingos. "Markov logic networks". In: *Machine Learning* 62.1-2 (2006), pp. 107–136. ISSN: 0885-6125. DOI: 10.1007/s10994-006-5833-1.

[140] C. Rousseau, Y. Bellik, F. Vernier, and D. Bazalgette. "A framework for the intelligent multimodal presentation of information". In: *Signal Processing – Special section: Multimodal human-computer interfaces* 86.12 (Dec. 2006), pp. 3696–3713. DOI: 10.1016/j.sigpro.2006.02.041.

[141] S. Sardina, L. de Silva, and L. Padgham. "Hierarchical Planning in BDI Agent Programming Languages: A Formal Approach". In: *Proc. of the 5th Int. Joint Conf. on Autonomous Agents and Multi-agent Systems (AAMAS 2006)*. ACM, 2006, pp. 1001–1008.

[142] Y. Sun, F. Chen, Y. Shi, and V. Chung. "A novel method for multi-sensory data fusion in multimodal human computer interaction". In: *Proc. of the 18th Australian Conf. on Computer-Human Interaction: Design: Activities, Artefacts and Environments (OZCHI 2006)*. ACM, 2006, pp. 401–404. DOI: 10.1145/1228175.1228257.

[143] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. "A decision-theoretic approach to task assistance for persons with dementia". In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Professional Book Center, 2005, pp. 1293–1299.

[144] B. Brandherm and T. Schwartz. "Geo referenced dynamic Bayesian networks for user positioning on mobile systems". In: *Int. Symposium on Location- and Context-Awareness (LoCA 2005)*. Springer, 2005, pp. 223–234. DOI: 10.1007/11426646_21.

[145] J. L. Bresina, A. K. Jónsson, P. H. Morris, and K. Rajan. "Activity Planning for the Mars Exploration Rovers". In: *Proc. of the 15th Int. Conf. on Automated Planning and Scheduling (ICAPS 2005)*. AAAI Press, 2005, pp. 40–49.

[146] J. L. Bresina, A. K. Jónsson, P. H. Morris, and K. Rajan. "Mixed-Initiative Activity Planning for Mars Rovers". In: *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Professional Book Center, 2005, pp. 1709–1710.

[147] S. P. Davies. "Planning and problem solving in well-defined domains". In: *The Cognitive Psychology of Planning*. Ed. by R. Morris and G. Ward. Chapter 2, pp. 35–52. Psychology Press, 2005. Chap. 2, pp. 35–52.

[148] R. Morris and G. Ward, eds. *The Cognitive Psychology of Planning*. Psychology Press, 2005.

[149] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, D. Wu, F. Yaman, H. Muñoz-Avila, and J. W. Murdock. "Applications of SHOP and SHOP2". In: *Intelligent Systems, IEEE* 20 (Mar. 2005), pp. 34–41. DOI: 10.1109/MIS.2005.20.

[150] G. Russ, B. Sallans, and H. Hareter. "Semantic Based Information Fusion in a Multimodal Interface". In: *Proc. of the 11th Int. Conf. on Human-Computer Interaction (HCI 2005)*. CSREA Press, 2005, pp. 94–102.

[151] F. Sørmo, J. Cassens, and A. Aamodt. "Explanation in case-based reasoning–perspectives and goals". In: *Artificial Intelligence Review* 24.2 (2005), pp. 109–143. DOI: 10.1007/s10462-005-4607-7.

[152] C. Stahl, J. Baus, B. Brandherm, M. Schmitz, and T. Schwartz. "Navigational - and shopping assistance on the basis of user interactions in intelligent environments". In: *Proc. of the IEE Int. Workshop on Intelligent Environments*. IET, 2005, pp. 182–191. DOI: 10.1049/ic:20050233.

[153] G. Ward and R. Morris. "Introduction to the psychology of planning". In: *The Cognitive Psychology of Planning*. Ed. by R. Morris and G. Ward. Chapter 1, pp. 1–34. Psychology Press, 2005. Chap. 1, pp. 1–34.

[154] J. Bouchet, L. Nigay, and T. Ganille. "ICARE software components for rapidly developing multimodal interfaces". In: *Proc. of the 6th Int. Conf. on Multimodal Interfaces (ICMI 2004)*. ACM, 2004, pp. 251–258. DOI: 10.1145/1027933.1027975.

[155] M. Ghallab, D. S. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Ed. by M. Ghallab, D. S. Nau, and P. Traverso. Morgan Kaufmann, 2004.

[156] M. Ghallab, D. S. Nau, and P. Traverso. "Hierarchical Task Network Planning". In: *Automated Planning: Theory and Practice*. Ed. by M. Ghallab, D. S. Nau, and P. Traverso. Morgan Kaufmann, 2004. Chap. 11, pp. 229–261.

[157] M. Ghallab, D. S. Nau, and P. Traverso. "Plan-Space Planning". In: *Automated Planning: Theory and Practice*. Ed. by M. Ghallab, D. S. Nau, and P. Traverso. Morgan Kaufmann, 2004. Chap. 5, pp. 85–109.

[158] D. Morley and K. Myers. "The SPARK Agent Framework". In: *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2004)*. IEEE Computer Society, 2004, pp. 714–721.

[159] L. M. Reeves, J. Lai, J. A. Larson, S. Oviatt, T. S. Balaji, S. Buisine, P. Collings, P. Cohen, B. Kraal, J.-C. Martin, M. McTear, T. Raman, K. M. Stanney, H. Su, and Q. Y. Wang. "Guidelines for Multimodal User Interface Design". In: *Commun. ACM* 47.1 (Jan. 2004), pp. 57–59. DOI: 10.1145/962081.962106.

[160] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. "A Unifying Reference Framework for Multi-Target User Interfaces". In: *Interacting with Computers* 15.3 (2003), pp. 289–308. DOI: 10.1016/S0953-5438(03)00010-9.

[161] J. M. Jishnu Mukerji. *MDA Guide Version 1.0.1*. OMG Consortium. June 2003.

[162] R. Mahler. "Multitarget Bayes filtering via first-order multitarget moments". In: *IEEE Transactions on Aerospace and Electronic Systems* 39.4 (Oct. 2003), pp. 1152–1178. DOI: 10.1109/TAES.2003.1261119.

[163] K. Myers, P. Jarvis, W. Tyson, and M. Wolverton. "A Mixed-initiative Framework for Robust Plan Sketching". In: *Proc. of the 13th Int. Conf. on Automated Planning and Scheduling (ICAPS 2003)*. AAAI Press, 2003, pp. 256–265.

[164] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. “SHOP2: An HTN Planning System”. In: *Journal of Artificial Intelligence Research (JAIR)* 20 (2003), pp. 379–404.

[165] W. Wahlster. “Smartkom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell”. In: *Proc. of the Human Computer Interaction Status Conference*. DLR, 2003, pp. 47–62.

[166] H. L. S. Younes and R. G. Simmons. “VHPOP: Versatile heuristic partial order planner”. In: *Journal of Artificial Intelligence Research (JAIR)* 20 (2003), pp. 405–430.

[167] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall. “GraphML Progress Report: Structural Layer Proposal”. In: *Graph Drawing*. Vol. 2265. Springer Berlin, 2002, pp. 501–512. ISBN: 978-3-540-43309-5. DOI: 10.1007/3-540-45848-4_59.

[168] G. Calvary, J. Coutaz, D. Thevenin, L. Bouillon, M. Florins, Q. Limbourg, N. Souchon, J. Vanderdonckt, L. Marucci, F. Paternò, and C. Santoro. *The CAMELEON Reference Framework*. Tech. rep. 1.1. CAMELEON Reference Framework Working Group, Sept. 2002.

[169] M. F. McTear. “Spoken Dialogue Technology: Enabling the Conversational User Interface”. In: *ACM Comput. Surv.* 34.1 (Mar. 2002), pp. 90–169. DOI: 10.1145/505282.505285.

[170] M. E. Pollack. “Planning Technology for Intelligent Cognitive Orthotics”. In: *Proc. of the 6th Int. Conf. on Artificial Intelligence Planning Systems (AIPS)*. AAAI Press, 2002, pp. 322–332.

[171] S. Biundo and B. Schattenberg. “From Abstract Crisis to Concrete Relief (A Preliminary Report on Combining State Abstraction and HTN Planning)”. In: *Proc. of the 6th European Conf. on Planning (ECP 2001)*. AAAI Press, 2001, pp. 157–168.

[172] A. Fiedler. “P.Rex: An Interactive Proof Explainer”. In: *Proc. of the First Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*. Springer, 2001, pp. 416–420. DOI: 10.1007/3-540-45744-5_33.

[173] G. Fischer. “User Modeling in Human-Computer Interaction”. In: *User Modeling and User-Adapted Interaction* 11.1-2 (2001), pp. 65–86. DOI: 10.1023/A:1011145532042.

[174] T. Herfet, T. Kirste, and M. Schnaider. “EMBASSI multimodal assistance for infotainment and service infrastructures”. In: *Computers & Graphics* 25.4 (2001), pp. 581–592. DOI: 10.1016/S0097-8493(01)00086-3.

[175] X. Nguyen and S. Kambhampati. “Reviving Partial Order Planning”. In: *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*. Vol. 17. Morgan Kaufmann, 2001, pp. 459–466.

[176] E. André. “The Generation of Multimedia Presentations”. In: *A Handbook of Natural Language Processing: techniques and applications for the processing of language as text*. Marcel Dekker Inc., 2000, pp. 305–327.

[177] M. Madsen and S. Gregor. “Measuring human-computer trust”. In: *Proc. of the 11th Australasian Conf. on Information Systems (ACIS 2000)*. Queensland University of Technology, 2000, pp. 6–8.

[178] A. Tate, J. Levine, P. Jarvis, and J. Dalton. “Using AI Planning Technology for Army Small Unit Operations”. In: *Proc. of the 5th Int. Conf. on Artificial Intelligence Planning Systems (AIPS 2000)*. AAAI Press, 2000, pp. 379–386.

[179] R. Barták. “Constraint Programming: In Pursuit of the Holy Grail”. In: *Proc. of Week of Doctoral Students (WDS 1999)*. 1999, pp. 555–564.

[180] R. Bastide and P. Palanque. “A visual and formal glue between application and interaction.” In: *Journal of Visual Languages and Computing* 10.5 (1999), pp. 481–507. DOI: 10.1006/jvlc.1999.0127.

[181] P. Cohen, D. McGee, S. Oviatt, L. Wu, J. Clow, R. King, S. Julier, and L. Rosenblum. “Multimodal Interaction for 2D and 3D Environments”. In: *IEEE Computer Graphics and Applications* 19 (4 July 1999), pp. 10–13. DOI: 10.1109/38.773958.

[182] A. K. Dey and G. D. Abowd. “Towards a better understanding of context and context-awareness”. In: *Proc. of the 1st Int. Symposium on Handheld and Ubiquitous Computing (HUC 1999)*. Springer-Verlag, 1999, pp. 304–307. DOI: 10.1007/3-540-48157-5_29.

[183] A. M. Holland-Minkley, R. Barzilay, and R. L. Constable. “Verbalization of High-level Formal Proofs”. In: *Proc. of the Sixteenth National Conf. on Artificial Intelligence (AAAI 1999)*. AAAI Press, 1999, pp. 277–284.

[184] M. Weiser. “The computer for the 21st century”. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 3.3 (1999). This article first appeared in Scientific America, Vol. 265, No. 3 (September 1991), pp 94-104, pp. 3–11. DOI: 10.1145/329124.329126.

[185] I. H. Beaumont. “User Modelling in the Interactive Anatomy Tutoring System ANATOM-TUTOR”. In: *Adaptive Hypertext and Hypermedia*. Springer Netherlands, 1998, pp. 91–115. DOI: 10.1007/978-94-017-0617-9_4.

[186] S. Kambhampati, A. Mali, and B. Srivastava. “Hybrid Planning for Partially Hierarchical Domains”. In: *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI 1998)*. AAAI Press, 1998, pp. 882–888.

[187] A. D. Mali and S. Kambhampati. “Encoding HTN Planning in Propositional Logic”. In: *Proc. of the 4th Int. Conf. on Artificial Intelligence Planning Systems (AIPS 1998)*. AAAI Press, 1998, pp. 190–198.

[188] R. Sharma, V. I. Pavlovic, and T. S. Huang. “Toward multimodal human-computer interface”. In: *Proc. of the IEEE* 86.5 (May 1998), pp. 853–869.

[189] R. Parasuraman and V. Riley. “Humans and Automation: Use, Misuse, Disuse, Abuse”. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 39.2 (June 1997), pp. 230–253. DOI: 10.1518/001872097778543886.

[190] M. Simons. “Proof Presentation for Isabelle”. In: *Proc. of the 10th Int. Conf. on Theorem Proving in Higher Order Logics (TPHOLs 1997)*. Springer, 1997, pp. 259–274. DOI: 10.1007/BFb0028399.

[191] K. Erol, J. A. Hendler, and D. S. Nau. “Complexity results for HTN planning”. In: *Annals of Mathematics and Artificial Intelligence* 18.1 (1996), pp. 69–93.

[192] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R. M. Young. “Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE Properties”. In: *Proc. of Human-Computer Interaction (INTERACT ’95)*. Springer US, 1995, pp. 115–120. DOI: 10.1007/978-1-5041-2896-4_19.

[193] D. Helbing and P. Molnar. “Social Force Model for Pedestrian Dynamics”. In: *Physical Review E* 51 (1995), pp. 4282–4286. DOI: 10.1103/PhysRevE.51.4282.

[194] B. Nebel and J. Koehler. “Plan reuse versus plan generation: A theoretical and empirical analysis”. In: *Artificial Intelligence* 76.1 (1995), pp. 427–454.

[195] L. Nigay and J. Coutaz. “A Generic Platform for Addressing the Multimodal Challenge”. In: *Proc. of the SIGCHI Conf. on Human factors in computing systems (CHI 1995)*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 98–105. DOI: h10.1145/223904.223917.

[196] A. S. Rao and M. P. Georgeff. “BDI Agents: From Theory to Practice”. In: *Proc. of the First Int. Conf. on Multi-Agent Systems (ICMAS 1995)*. MIT Press, 1995, pp. 312–319.

[197] F. Bacchus and Q. Yang. “Downward refinement and the efficiency of hierarchical problem solving”. In: *Artificial Intelligence* 71.1 (1994), pp. 43–100. DOI: 10.1016/0004-3702(94)90062-0.

[198] B. M. Muir. “Trust in automation: Part I. Theoretical issues in the study of trust and human intervention in automated systems”. In: *Ergonomics* 37.11 (1994), pp. 1905–1922. DOI: 10.1080/00140139408964957.

[199] A. Tate, B. Drabble, and R. Kirby. “O-Plan2: an Open Architecture for Command, Planning and Control”. In: *Intelligent Scheduling*. Ed. by M. Zweben and M. S. Fox. Morgan Kaufmann, 1994, pp. 213–239.

[200] J. S. Penberthy and D. S. Weld. “UCPOP: A Sound, Complete, Partial Order Planner for ADL”. In: *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 1992)*. Morgan Kaufmann, 1992, pp. 103–114.

[201] D. McAllester and D. Rosenblitt. “Systematic Nonlinear Planning”. In: *Proc. of the 9th National Conf. on Artificial Intelligence (AAAI 1991)*. AAAI Press, 1991, pp. 634–639.

[202] Q. Yang. “Formalizing Planning Knowledge for Hierarchical Planning”. In: *Computational Intelligence* 6.1 (1990), pp. 12–24. DOI: 10.1111/j.1467-8640.1990.tb00126.x.

[203] R. D. Smallwood and E. J. Sondik. “The Optimal Control of Partially Observable Markov Processes over a Finite Horizon”. In: *Operations Research* 21.5 (Oct. 1973), pp. 1071–1088. DOI: 10.1287/opre.21.5.1071.

[204] A. Newell, H. A. Simon, et al. *Human problem solving*. Vol. 104. 9. Prentice-Hall Englewood Cliffs, NJ, 1972.