Plan, Repair, Execute, Explain – How Planning Helps to Assemble your Home Theater

Pascal Bercher, Susanne Biundo, Thomas Geier
Thilo Hoernle, Felix Richter, Bernd Schattenberg
Institute of Artificial Intelligence,
University of Ulm, Germany,
email: firstName.lastName@uni-ulm.de

Florian Nothdurft
Institute of Communications Engineering
University of Ulm, Germany
email: florian.nothdurft@uni-ulm.de

Abstract

In various social, work-related, or educational contexts, an increasing demand for intelligent assistance systems can be observed. In this paper, we present a domain-independent approach that combines a number of planning and interaction components to realize advanced user assistance. Based on a hybrid planning formalism, the components provide facilities including the generation, execution, and repair as well as the presentation and explanation of plans. We demonstrate the feasibility of our approach by means of a system that aims to assist users in the assembly of their home theater. An empirical evaluation shows the benefit of such a supportive system, in particular for persons with a lack of domain expertise.

Introduction

Today's rapid technological progress periodically provides us with technical systems, services and devices of increasingly complex, "intelligent", functionality. Those systems include household appliances, web services, cars, service robots, electronic devices such as smartphones, and much more. However, it is not always easy to operate these systems and, even less so, to actually exploit the whole range of their potential. In many cases, users are overwhelmed or stressed out by awkward operation menus, inappropriate interaction modes, or inconvenient or even missing instruction manuals. Moreover, the demand for assistance systems that support users in their every-day life by helping them to accomplish certain tasks or to handle systems and devices appropriately becomes more and more important, in particular in view of the aging society.

We show how AI planning technology can be employed to perform user assistance in a smart and valuable fashion. We introduce a system architecture composed of a number of planning components, a knowledge base, as well as components to manage the system's dialog and interaction with the user. The planning components include a hybrid planner that combines hierarchical concepts with those of partial-order causal-link planning. In addition, a plan execution system, as well as plan repair and explanation components are parts of the architecture. User assistance is based on automatically generated plans. Such a plan of actions is passed to the user in a stepwise manner. The user's execution of these actions is monitored. This way, the system is able to detect execution failures or deviations from the proposed plan and to initiate the repair of the original plan accordingly. Furthermore, explanations can be given, which elucidate why a certain action is part of the plan, for example. Plans and explanations need to be communicated to users in an adequate manner, however. Therefore, the system's facilities of establishing dialogs with users and interacting through various modalities are controlled by advanced dialog and interaction management components. They enable the system to give instructions and explanations via displayed text and speech, as well as to understand users' spoken responses and input via touch screen. We used the planning and interaction components and their orchestration within the overall architecture to implement a prototype system that assists users in the assembly of their home theater. Finally, we carried out an empirical user study to evaluate the acceptance and benefit of such an assistance system. It showed that the system's support was perceived very well, in particular by non-experts.

Planning technology has been recently used in several human computer interaction-related contexts. One line of research aims at supporting persons with various impairments, such as children with autism (Bernardini and Porayska-Pomsta 2013) or persons suffering from dementia (Boger et al. 2005). Another line investigates systems that realize assistance functionality by performing certain interaction-related tasks presently done by humans, such as household chores (Beetz et al. 2012) or serving drinks (Petrick and Foster 2013). These approaches do not incorporate facilities for revising or explaining the decisions of the system, however. Approaches that include some form of plan repair exist, e.g., the O-Plan architecture (Tate, Drabble, and Kirby 1994), or in the context of interactive storytelling (Porteous, Cavazza, and Charles 2010) or robotics (Lemai and Ingrand 2004).

In the next section, we present the system architecture and the interplay of its major constituents. The following sections describe these constituents in more detail, focusing on the planning components. After that, we present the user study together with its results and conclude the paper.

System Architecture

We have developed an architecture which integrates essential planning capabilities (plan generation, execution/moni-

toring, repair, explanation) with components to provide advanced human-computer interaction (dialog and interaction management). The architecture and the domain-independent components enable the implementation of concrete assistance systems in a variety of application areas. Such systems are capable of multi-modally interacting with a human user and to provide intelligent assistance based on the various capabilities provided by the planning and interaction components. The architecture is depicted in Fig. 2. We will shortly explain how the different components interact, before we describe their particular functionality in more detail.

Assistance functionality is provided through the interplay of various planning capabilities; the most basic one being that the system can give a course of actions to a user which solves his current problem at hand. We assume that these problems are formalized in terms of a hybrid planning domain. Knowledge about the specific domain is stored in the *Knowledge Base component*, which manages and processes the information necessary for the other components to work.

As soon as a user requests assistance in terms of a course of action to solve a given problem, the Knowledge Base component passes the corresponding hybrid planning problem to the *Plan Generation component* (arrow 1 in Fig. 2). That component generates a solution, i.e., a plan, and passes it to both the *Plan Execution component* and the *Plan Explanation component* (arrow 2). The latter is done to be prepared in case the user wants to ask questions about the current plan. The plan is executed step by step, and several components (arrows 3 to 10) are involved for each such step.

The Plan Execution component identifies the plan step to be executed next and sends it to the *Dialog Management component* (arrow 3). This way, it can be presented to the user. The component loads the dialog model associated with the plan step and calculates the most-appropriate *dialog goals* to achieve the effects of the given plan step. The chosen dialog goals are passed one by one to the *Interaction Management component* (arrow 4), which builds the actual user interface to be presented to the user (arrow 5). The input of the user is sent back over the Interaction Management component to the Dialog Management component (arrow 8). This constitutes both explicit input entered via the user interface (arrow 6), and implicit input (e.g., facial expression), which is registered by the sensors (arrow 7).

Once a dialog goal is finished, its effects are sent to the Knowledge Base component (arrow 9). A common pattern is to associate a dialog goal to a plan step, and interpret the user's input as an acknowledgement of having achieved the effects of the plan step. The effects registered by the Interaction Management component are then combined with sensor inputs within the Knowledge Base component to calculate a belief about the current world state. This information is then passed back to the Plan Execution component (arrow 10). That component compares the believed state with the expected state, using the effects of the last plan step. If they match, the next plan step is executed in the same way. If a deviation from the expected state is detected, the Plan Execution component triggers the *Plan Repair component* (arrow 11). Repair is initiated by constructing an *augmented planning problem*, which incorporates the found discrepancies together with the execution state of the currently enacted plan. The augmented problem is then sent to the Plan Generation component (arrow 12), which finds a new solution replacing the old plan, s.t. a new cycle can start over.

Figure 1: The figure shows the back panel of the amplifier used in our domain model and user study.

At any point in the interaction, the user may ask why he had to execute any of the presented plan steps. If he does so, the Dialog Management component requests the explanation of the current plan step from the Plan Explanation component (arrow 13), which generates a formal explanation stating why this plan step is necessary to solve the overall problem. This explanation is sent back to the Dialog Management component, which initiates its presentation to the user (arrows 14 and 4 to 8).

To illustrate how such an architecture works in detail, we implemented a prototypical system that assists a person in the task of setting up a complex home theater. The interaction between the various components is realized with the message-oriented middleware Semaine (Schröder 2010). The main physical features of the prototype are a multi-device user interface (one touch-screen, one remote terminal) and user-localization based on a laser-range-finder (Honold et al. 2013, Figure 10). No additional sensory modules were used, in particular there were no means of emotion recognition, or means of automatically recognizing the state of the home theater setup and cables. Progress on the task was solely reported through the dialog system.

The task of the user is to connect several devices of his new home theater. They comprise a satellite receiver, a blu-ray player, an amplifier, and a television. Finally, the user wants to be able to use both the blu-ray player and the satellite receiver. While this task may seem easy for an expert, it is quite complicated for non-experts: Fig. 1 shows the back panel of the modeled amplifier (an audio/video receiver, which we also used in the evaluation) and illustrates the high number of different ports and thus the difficulty of the task. In addition to these devices, we modeled several different cables, adapters, and adapter cables (such as a HDMI-to-DVI adapter cable, for example). The information about available devices, cables, and other information relevant for the task (e.g., about the user) is stored in the Knowledge Base component and sent to the respective components if requested.

Knowledge Base

Besides storing all the domain specific information, the Knowledge Base component serves as the central hub for

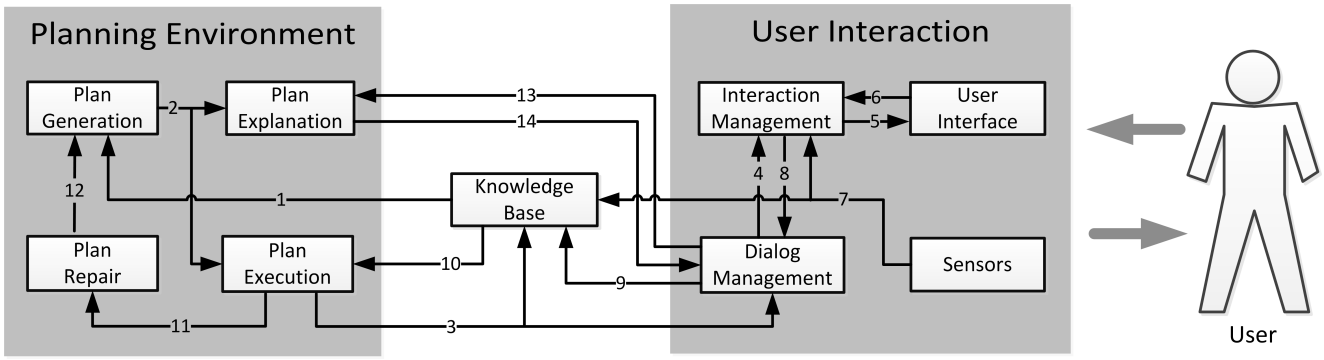

Figure 2: This figure shows the components of our architecture and how they interact with the user and with each other. It is based on the one presented by Honold, Schüssel, and Weber (2012), with emphasis on the planning part instead of the user interaction part. The numbers represent the actual data sent from one component to another: 1: planning problem, 2: generated solution plan, 3: next plan step, 4: next dialog goal, 5: interaction output, 6: explicit interaction input, 7: implicit sensor input, 8: fusion result, 9: interaction effects, 10: actual world state, 11: repair request, 12: plan repair problem, 13: explanation request, 14: formal plan explanation.

our architecture. It integrates information from various sources like sensor data (arrow 7, usually pre-processed by machine learning modules, not depicted), and explicit user input via the Dialog Management component (arrow 9). A major part of this information is of an uncertain nature due to sensor noise, partial observability or model imperfection.

The requirement of a consistent treatment of uncertainty throughout the whole system is addressed by choosing a probabilistic approach. The knowledge of the system is reflected in a probability distribution over the possible world states. Advances in the field over the past 20 years have led us to a point where the treatment of probability distributions over many random variables slowly becomes feasible, e.g., by the use of probabilistic graphical models (Koller and Friedman 2009). To facilitate the concise representation of large graphical models, and in order to ease the integration with other modules like automated planning, we use a logic-like, probabilistic, first-order modeling language called Markov Logic (Richardson and Domingos 2006).

The integration of the knowledge base with the planning components mainly focuses on the provision of a world state. Initially, the Knowledge Base component is queried for all state-features relevant to the current planning problem (arrow 1). In the case of the assembly task, this encompasses knowledge about available devices and cables. During plan execution, relevant state features are then constantly monitored (arrow 10).

To make the probabilistic view of the knowledge base compatible with the deterministic state representation of the planner, one could infer the most probable world state according to the current beliefs and report it as *actual* world state. But for models with many random variables it is very likely that the most probable state still has a vanishingly small probability, and the chance that the true world state differs from the reported one is huge. To handle this, we declare the part of the global model that comes from the planning domain as quasi-deterministic. In this part, upon each new time-step, variables that have not received an explicit observation (e.g., from the dialog) are automatically observed to the most probable state they had during the last time step. This makes the quasi-deterministic part behave like a database, where values only change upon a write request, offering semantics that are compatible with assumptions of deterministic planning. Still, the probabilistic part can transparently use information from the quasi-deterministic part for inferences. The usefulness of such a connection has been demonstrated by its exploitation for improved user identification (Geier et al. 2012).

Plan Generation

In order to facilitate the provision of a suitable plan, the Plan Generation component queries the current world state from the Knowledge Base component. A valid plan will then consist of a course of actions, whose execution will transform the current world state into a desired state. For the assembly task, actions are usually of the form "Connect port A of cable X to port B of device Y.", and the goal conditions consist of the transmission of certain signals (audio or video) from source devices to their respective sink devices.

In our system, such tasks are modeled by means of a hybrid planning problem (Biundo and Schattenberg 2001; Biundo et al. 2011; Geier and Bercher 2011). Hybrid planning combines hierarchical planning (Erol, Hendler, and Nau 1994; Marthi, Russell, and Wolfe 2008) with concepts known from Partial-Order Causal-Link (POCL) planning (McAllester and Rosenblitt 1991; Younes and Simmons 2003). Both paradigms seem to be well-suited for our enterprise of assisting human users: The hierarchical aspect of the domain captures the hierarchical structure many real-life problems show and the explicit representation of causality using causal links allows for the explanation of plans (Seegebarth et al. 2012). These explanations may further benefit from the hierarchical structure, as it allows more flexibility w.r.t. the level of detail of explanations. We believe that the capability of systems to explain their own behavior is essential when assisting human users, since explanations

can improve the trust a user has in the system and might lead to higher acceptance rates (Lim, Dey, and Avrahami 2009).

In hybrid planning, the task is modeled by means of a planning domain $\mathcal{D}$ and a problem instance $\mathcal{I}$. For our example application, the planning domain models the specific technical devices (ports male/female, audio/video/either/both signal in/out, etc.) and how they can be connected using the available cables. The problem instance specifies the actual task to solve. In hybrid planning, actions may be either primitive or abstract. In both cases, an action $a(\bar{\tau})$ has the form $\langle pre, \mathit{eff} \rangle$ with pre and eff being conjunctions of literals over the action parameters $\bar{\tau} = \tau_1, \ldots, \tau_n$. Preconditions specify in which states an action is applicable, and effects of actions specify how a state changes if the action is applied. States and action applicability and application are defined as usual. A partial plan is a structure $\langle PS, \prec, VC, CL \rangle$ consisting of a set of plan steps PS, which are uniquely identified actions, either primitive or abstract. The partial order between these plan steps is achieved by the relation $\prec$. The set VC contains the variable constraints, which co- or non-codesignate action parameters to other parameters or constants. We call an action ground if all of its parameters are codesignated to some constant taking the closure of VC. A plan is called ground if all its actions are ground. The set CL contains all causal links of the partial plan P. A causal link $ps \rightarrow_{\varphi} ps'$ denotes that the precondition literal φ of plan step ps' is supported (protected) by the plan step ps. While primitive actions can be executed if all their preconditions are supported by causal links, abstract actions must be further refined, even if their preconditions are all supported. To that end, the domain $\mathcal{D}$ contains at least one so-called *decomposition method* for every abstract action. A decomposition method $m = \langle a(\bar{\tau}), VC_m, P \rangle$ maps an abstract action $a(\bar{\tau})$ to a partial plan P, which "implements" the preconditions and effects of $a(\bar{\tau})$ (Biundo and Schattenberg 2001). Applying a method to a partial plan results in removing the abstract action and replacing it by the partial plan the method maps to, adding the variable constraints VC_m, which relate the action parameters to the variables in P, and to pass on the causal links of the abstract action to its new sub-actions.

Now, the domain $\mathcal{D} = \langle A, M \rangle$ contains the set of primitive and abstract actions A as well as the set of decomposition methods M. The problem instance $\mathcal{I}$ is given in terms of an initial partial plan P_{init}, which contains some abstract actions representing the high-level specification of the tasks to be achieved. It also contains two artificial actions $init$ and $goal$, which encode the initial state and the goal description, respectively. The action $init$ has no precondition and a complete description of the initial state as effect, including the facts initially false in terms of negative literals[1]. The action $goal$ has the goal description as precondition, which is a conjunction of (possibly negative) literals. The solution to a planning problem is a plan P, s.t.:

1. P is a refinement of P_{init}.
2. P does not contain abstract actions.
3. P does not show unprotected precondition literals.
4. P has no causal threats. A causal threat is the situation where the ordering constraints allow a plan step ps'' with effect φ to be ordered between two plan steps ps and ps' which share a causal link $ps \rightarrow_{\varphi'} ps'$, s.t. there is a unification σ with $\varphi = \neg\sigma(\varphi')$.

Criterion 1 is essential for relating any solution to the initial plan P_{init}. In contrast to PDDL, where the problem is given merely in terms of a goal description, the goals in hierarchical/hybrid planning are given in terms of P_{init}. Criterion 2 ensures that only primitive actions are present, as only those can be executed directly. Criteria 3 and 4 together ensure that every plan step linearization which is consistent with P's ordering constraints is an applicable action sequence and generates a state which satisfies the goal description.

Domain Model. We will now give an overview over how we modeled the problem of setting up a home theater. In our planning domain, the initial state specifies the available hardware and its compatibility to other devices. We differentiate between devices and cables, which both have an individual sort DEVICE and CABLE, respectively. Sorts correspond to the concept of types in PDDL. For every device and cable in the scenario, there is a constant of the respective sort, such as BLURAY of sort DEVICE to model the blu-ray player. We also introduce an abstract sort HARDWARE with sub sorts DEVICE and CABLE to be used by the actions. Every hardware in the scenario has several ports: Cables, for instance, ordinarily have two ports, one for each end of the cable (however, more complicated cables can have more than one port at the same end). We thus introduce a sort PORT. Every port of a specific hardware has certain properties: it may be either a *signal in or out*, either *used or unused*, either *male or female*, and it may be used for *video, audio, both* (in case of HDMI), *or either* (for instance, a cinch video cable may be used for either audio or video, but not for both at the same time as is the case for HDMI). The description of the hardware also specifies that certain devices are sources of audio and video signals. For instance, while the TV and the amplifier initially do not have any signal, the blu-ray player and the satellite receiver initially have both video as well as an audio signal. In our domain, we therefore have a predicate HASVIDEO(HARDWARE,DEVICE) expressing that the constant of sort HARDWARE has the video signal produced by the constant of sort DEVICE. In our scenario, the initial state thus contains the atoms HASVIDEO(BLURAY,BLURAY) and HASVIDEO(RECEIVER,RECEIVER) as well as the respective counter parts for the audio signal.

Actions are the means to connect cables to devices. More precisely, we modeled the actions in such a way that they can only be applied if the two ports to be connected are both unused and compatible with each other (w.r.t. gender, for instance). Furthermore, actions may only be applied in the direction of the signals in order to propagate that signal to the cable just plugged in. That is, a cable with an input port may only be connected to a cable or device with an output port if the other cable/device already has some signal. Then,

[1] In POCL planning, we need to specify the facts initially false to be able to support negative preconditions of actions.

the cable that has been plugged in will also have that signal. Note that we did not model *unplug* actions. Thus, in case of execution failures, our prototype might not always find a repaired solution in certain situations.

Hierarchy may be introduced to reduce the search space by specifying pre-defined solutions to sub-problems. For example, an abstract action CONNECT(BLURAY,AMPLIFIER) can be modeled using several decomposition methods, each corresponding to a valid solution transporting both audio and video from the blu-ray player to the amplifier. The initial plan may then contain that abstract action as well as CONNECT(RECEIVER,AMPLIFIER) and CONNECT(AMPLIFIER, TV).

Search Procedure. We search for solutions via search in the space of plans by step-wise refining the initial partial plan until a solution has been found. We base our search technique on the standard POCL planning paradigm (Younes and Simmons 2003), in which plan elements violating solution criteria induce so-called flaws. We extend this technique by being able to decompose abstract actions; we call the resulting system PANDA (Biundo and Schattenberg 2001; Elkawkagy et al. 2012). The search is a two-stage process. First, a most-promising plan is selected from a set of candidates. That selection is done using informed heuristic functions. Our system supports pure non-hierarchical heuristics (Nguyen and Kambhampati 2001; Younes and Simmons 2003; Bercher, Geier, and Biundo 2013; Bercher et al. 2013), as well as hierarchical ones (Elkawkagy et al. 2012). After a most-promising plan has been selected, one of its flaws is selected to be resolved. Resolving a flaw generates a set of successor plans, which can contain new flaws. For example, the insertion of an action adds one flaw for each of its precondition literals. That procedure is repeated until a solution has been created.

Plan Execution and Monitoring

Given a solution plan, the Plan Execution component executes it in a step by step way. When prompted to execute the next plan step, it first determines the set of executable plan steps. A plan step is executable if it has not yet been executed, whereas all plan steps which are ordered before it, have already been executed. The Plan Execution component then chooses a plan step to execute from this set. While any execution order will guarantee that the goal will be reached, some orders can be more intuitive for the user. The implemented system therefore employs a domain-specific approach to execute semantically connected tasks in direct succession, if possible. For example, several steps are needed to transport a video signal from the satellite receiver to the TV. If the last executed step was one of these steps, the component prefers executing another such step next. In future work, we would like to examine the extent to which this could be done domain-independently, for example by maximizing the number of common parameter values of two consecutively ordered tasks.

The chosen plan step is passed on to the Interaction Management component, where the actual execution is performed. Once the execution is completed, the interaction management passes the results of the execution to the Knowledge Base component and sends a signal to the Plan Execution component. The Plan Execution component then checks whether the action had the intended effects. This is done by comparing the expected state – as computed by applying the specified effects of the action to the last known world state – with the actual world state obtained by querying the knowledge base. When the expected and actual world states match, the next plan step is executed.

However, it might happen that the actual world state deviates from the expected state. For example, when plugging an HDMI cable into the blu-ray player fails because the HDMI cable is broken, the actual world state will be that the cable is unusable and that it is not connected to the blu-ray player. In this case, it has to be checked whether the new world state interferes with the yet-to-be-executed parts of the plan. This can be done by examining the presently *active* causal links of the plan, i.e., the causal links whose producer task has been executed while its consumer task has not.

If there is no active causal link for a given literal then it is either irrelevant for executing the remainder of the plan or a yet-to-be-executed task will reestablish it at a later point. Therefore, if the current state agrees with the expected state on all literals for which there is an active causal link, the plan is sure to still be a valid solution. Otherwise, the plan contains tasks (that might be executed far in the future) whose preconditions are no longer supported by valid causal links and the plan needs to be repaired.

Plan Repair

When the Plan Execution component detects that the plan at hand cannot be executed as intended, that plan, together with the problems detected during execution, is passed on to the Plan Repair component, which generates a "repair problem". The Plan Generation component uses this repair problem to generate a repaired solution, if possible.

There might be several reasons for a plan to fail. Often, the reason lies in the non-deterministic and partially observable nature of the environment. For example, an action might not be applicable (now or in the future), because one of its precondition literals does not hold at the current world state while the system predicted that precondition to hold. In our domain, for instance, a cable might be defective or a device port might not be working against expectation.

In these cases we need to find another – working – solution to the original planning problem, which can cope with the unpredicted changes of the world. If the new problem turns out to be unsolvable, we at least need to inform the user that it is no longer possible to set up the home theater system with the remaining set of cables and adapters. While in standard state-based planning, *replanning* seems most attractive (Nebel and Koehler 1995), it is not that easy in hierarchical approaches like we follow, since solutions need not only be executable, but also be a refinement of the initial plan (cf. solution criterion 1). We thereby follow a *plan repair* approach, which is suited for this additional constraint (Bidot, Biundo, and Schattenberg 2008; Biundo et al. 2011).

The repair mechanism basically works as follows: Given the planning domain $\mathcal{D} = \langle A, M \rangle$, the initial problem instance $\mathcal{I}$ given by P_{init}, and the failed solution plan P, the Plan Execution component creates a repair problem instance $\mathcal{I} = \langle P_{init}, O \rangle$, which now contains a set of *obligations* O. This set contains an existence obligation for all plan steps which have already been executed. Satisfying these obligations will guarantee that executed plan steps are part of any new solution. This is important, as we require solutions which are refinements of the initial plan and the plan steps already executed may be essential to satisfy that criterion. Since these plan steps have been executed in one specific order (while the plan that was to be executed might only be *partially* ordered), O also contains ordering obligations which restrict these plan steps to be totally ordered and prevent any "new" action to be ordered within this sequence. Lastly, O contains an obligation that requires a so-called *process* to be inserted right behind the executed plan step sequence. A process is a primitive action with no precondition and the unpredicted world changes as effect.

The planning procedure is altered in such a way that unsatisfied obligations are represented as additional flaws. Then, plans with no flaws are solutions to the original planning problem, they contain the already executed plan steps, and can cope with the unforeseen changes caused by the environment.

Plan Explanation

So, for example, when the system learns about the fact that one of the cables is broken, it comes up with a new solution. This solution is usually more complicated than the original one, since the problem of a missing cable must be worked around. A broken cable could be addressed by using adapters and adapter cables as a replacement. However, the pro-active generation of a new plan, in particular a more complicated one, may confuse the user. Even worse, the system can produce changes in the plan that are not anticipated by the user, because the deviation was not detected via user interaction (e.g., the user reporting the failure of the cable), but it was detected via different means, e.g., computer vision. The explanation of such unexpected or otherwise opaque decisions is critical for the human-computer interaction, because especially unexpected or not understandable situations may have a negative impact on the human-computer trust relationship (Muir 1992). Studies have shown that if the user does not trust the system, the interaction may suffer. This includes reduced frequency or way of interaction, and in the worst case the complete abortion of future interaction (Parasuraman and Riley 1997). Of course, as we want technical systems to become intelligent assistants and help us in complex, as well as in critical situations, it is clear that this should be impeded.

In human-human interaction moments of unclear, not reasonable decisions by one party are often clarified by explaining the process of reasoning (i.e., increasing transparency and understandability). Analogous to that, research by Lim et al. (2009) showed that different kinds of explanations can improve the trust in and the understandability of context-aware intelligent systems. The results showed that *Why* and *Why-not* explanations were the best kind of explanation to increase the user's understanding in the system, though trust was only increased by providing *Why* explanations.

Therefore, our planning system can help improve trust and understandability by providing, if requested, plan explanations. The special case of plan explanation used in this setting is the task of explaining why a given plan step is present in a given plan. To generate explanations, an axiomatic system Σ comprising formalizations of basic arguments about a plan and their logical consequences is constructed (Seegebarth et al. 2012). The axioms are derived from the solution criteria, the plan at hand, and the problem specification. We use the atom $N(ps)$ for representing that a step ps is necessary for a given plan P to constitute a solution. For a formula ϕ encoding a reason that supports the presence of ps in P, Σ contains an axiom of the form $\forall ps.[\phi \Rightarrow N(ps)]$. Constructing an explanation for the presence of a plan step is equivalent to finding a sequence of "applications" of these axioms.

The reason for the presence of a plan step ps in a plan P, according to the solution criteria for hybrid plans, can be one of the following: First, ps is trivially either $init$, $goal$, or any plan step from the problem specification, i.e., Σ contains $N(init)$, $N(goal)$, and $N(ps)$ for all ps in P_{init}. Second, ps establishes a precondition of some other plan step ps'. For this, Σ contains an atom $CR(ps, \varphi, ps')$ for every causal link $ps \rightarrow_{\varphi} ps'$ present in P, representing the *causal relation* between ps and ps'. The necessity of further plan steps can be recursively derived from causal relations to necessary plan steps with the following axiom:

$$\forall ps.[[\exists ps', \varphi.[CR(ps, \varphi, ps') \wedge N(ps')]] \Rightarrow N(ps)] \quad (1)$$

The third reason for the presence of a plan step ps in P is that ps has been introduced by decomposing an abstract plan step ps' during the construction of P. In contrast to causal dependencies, this information is not represented explicitly in the solution plan and has to be (re-)constructed from the knowledge about the plan generation process: for each decomposition of an abstract plan step ps' via a method m, performed during the construction of P from P_{init}, Σ contains an atom $DR(ps, m, ps')$ for each plan step ps introduced by m, representing the *decomposition relation* between ps and ps'. Plan step necessity can again be recursively derived:

$$\forall ps.[[\exists ps', m.[DR(ps, m, ps') \wedge N(ps')]] \Rightarrow N(ps)] \quad (2)$$

The result of an explanation request is a formal proof that has to be conveyed to the user in a more suitable from, for instance verbally. The simplest way of achieving this is to provide a text template for each axiom and translating an explanation proof step by proof step.

As an example, consider the case where a video signal needs to be transported from the blu-ray player to the TV, reflected in the goal description via HASVIDEO(TV,BLURAY). The solution plan contains a sequence of plan steps connecting devices and cables. The tasks are connected with causal links successively providing the HASVIDEO(X,BLURAY) property to the next task in the sequence. If the user requests an explanation for the necessity of the plan step plugging a cable into the blu-ray

player, the Plan Explanation component uses the above axioms to successively determine the need to derive the necessity of all plan steps in the sequence, including the goal step. Since the goal step is necessary by definition, so are all the other plan steps in the sequence.

Explanations subsume several kinds of explanations like *Why*, *Why-not*, *How-to*, or *What*. Plan explanations in our system are meant to increase the understandability in the system's decisions by using *Why* explanations. Therefore, plan explanation is to be distinguished from explanation of declarative knowledge, which uses *What* and *How-to* explanations to impart knowledge (Nothdurft and Minker 2012).

Dialog and Interaction Management

In the case of required user-interaction, plan steps are transmitted to the Dialog Management component (cf. Fig. 2). Here, the provided plan step is decomposed into a hierarchical dialog structure which consists of so-called dialog goals (Nothdurft et al. 2010). Each dialog goal represents a single interaction step between human and technical system (e.g., one screen on a device filled with specific information). The term dialog "goal" arises from the fact that every step in the interaction pursues a goal. This goal is, in this case, to achieve one or several of the desired plan step effects. Therefore, the term *dialog goal* is to be distinguished from the term *goal* used in planning.

This means on the one hand that a plan step may be decomposed into several steps of interaction, and on the other hand that for every desired plan step effect a set of similar dialog goals may exist. These similar dialog goals may for example take into account user knowledge or emotions, and therefore differ in the complexity and kind of content presentation. The selection of the next dialog goal is made in a user-adaptive manner and leads to an individual dialog appropriate for the current user (Honold et al. 2013). For information presentation the dialog goal is passed on to the Interaction Management component (cf. Fig. 2) and by that transferred to a XML-based format called *Dialog Output*.

Hereby, the dialog content is composed of multiple information objects referencing so-called *information IDs* in the information model. Each information ID can consist of different types (e.g., text, audio, and pictures) which are selected and combined at runtime by a fission subcomponent to compose the user interface in a user- and situation-adaptive way. The reasoning process about the most-appropriate user interface is based on a set of evaluation functions. These functions rate the selection of possible output modalities and combinations by a reward function and propagate the best one for presentation (Honold, Schüssel, and Weber 2012).

Performed user input is perceived by the input devices and transmitted to the multimodal fusion (Schüssel, Honold, and Weber 2013). The interaction input is passed on back to the Dialog Management component, which analyzes how the interaction result influences the desired plan step effects. The effects of the user interaction are transferred to the Knowledge Base component and the Plan Execution component is notified that the current plan step has been processed by the dialog management.

Additionally to that, if the user requested an explanation why the current plan step has to be executed, the dialog management can transfer a plan explanation request to the Plan Explanation component.

Experimental Evaluation

We conducted an experiment to evaluate the benefit of providing plan explanation for the acceptance of an automated assistive system. The experiment is designed as a controlled, randomized trial. This allows us to investigate the effects of plan explanations in a principled way. For reasons of reproducibility, we recreated a fixed course from the prototype system as a seemingly interactive HTML5 slide show. Participants are confronted with the task of connecting several devices of a home theater system consisting of a satellite receiver, a blu-ray player, an amplifier, and a television, as explained in the beginning of the paper. Fig. 1 shows the back panel of the amplifier used in the study. The requirements to the solution were that both watching satellite, TV, and blu-ray disks is possible. The task was complicated by the fact that the enacted solution uses an adapter for one connection. Participants were given an electronic assistant that gave them instructions on which cable to connect to which device through written and spoken text, as well as by images of the devices with the concerned ports highlighted. Participants were randomly, and without them knowing, assigned into two groups. The control group ($n = 29$) was faced with the task as described. The treatment group ($n = 30$) was shown two plan explanations at fixed steps. The longer explanation was: "This step served the goal to transmit the video signal of the blu-ray player to the TV. To this end, the video signal of the blu-ray player is transmitted over the HDMI-to-DVI adapter and the HDMI-to-DVI cable to the amplifier. From there it is transmitted over the video-cinch cable to the TV.".

The outcome of the trial was captured by a self-report questionnaire administered right after the task. Most importantly we asked for the subjectively perceived certainty that the way how the devices are connected fulfills the stated requirements as a 5-point Likert question (e.g., five levels from "No Chance" to "Certain"). In addition to demographic variables, we asked about expertise and prior experience with similar tasks, several other questions about how the participant perceived the device, and questions about how the explanations were perceived (only for the treatment group).

Our main hypothesis was that the plan explanations have a positive effect on the subjectively perceived certainty in the validity of the setup solution.

We recorded 59 subjects in total. Concerning demographic variables, we have at 3 female and 7 male subjects aged over 30, and 19 female and 27 male subjects aged at most thirty; age is missing for three subjects; 26 subjects had a university degree. Nine participants alone were Ph.D. students in chemistry. Only 7 subjects' graduation was lesser than high school equivalent. In general the education level of the sample was overly high. Only 12 participants had a technical background (computer science, engineering); though this does not include natural sciences and Mathematics, to which another 18 subjects affiliated.

We constructed a summary variable capturing the overall perception of the system by summing up all questions that rate aspects of the system (trust, patronization, appeal, utility, etc.; with good internal consistency $\alpha = 0.83$). According to this scale, the system was very well perceived in general (mean/sd: $26.63/3.67$ points out of 30). We found a major influence on this rating in the answer on a question (5-point Likert scale) asking people to judge their confidence to do the setup as required, but while only using the device manuals. The answer to that question significantly predicted the overall perception in a linear regression ($\beta = -0.37, t(55) = -2.99, p < .01$). Thus, people who consider themselves unskilled liked the system better. We also found that women rated the system better than men (mean/sd female $28.14/1.93$, male $25.67/4.19$, significant with $p < .01$ using Mann-Whitney-U, $Z = -2.81$).

Considering the differences between subjects in the treatment group and those in the control group, the main hypothesis that explanation fosters confidence in the implemented solution cannot be supported by the experimental results. In contrary, subjects of the treatment group had lower confidence on average (mean/sd: control $4.66/0.55$, treatment $4.50/0.82$, not significant). However, the confidence of the subjects in the correctness of their assembly was very high in general. Some subjects rated their confidence "certain", even though they forgot to connect some of the cables. Subjects within the treatment group also had lower overall perception scores compared to the control group (mean/sd: control $27.4/2.6$, treatment $25.8/4.4$, not significant).

We attribute the result concerning the explanation feature to several factors. First, there is no substantial risk involved in making a mistake while connecting the components. A simple experiment (turn the devices on and see if it works) can yield a reliable answer without much effort. So being pushed towards reasoning about the correctness may be perceived as annoying. Remember that the explanations were displayed without an explicit request by the user, to facilitate randomization. A second reason for the result within the treatment group is the possibility of an unsettling effect of the explanation, caused by the participants thinking they are to be fooled into believing something wrong — facilitated by the experimental condition.

When looking only at the treatment group, we found that people with a higher education level rated the explanation aspect[2] better (mean/sd: with German "Abitur" $14.9/5.38$, with university degree $17.9/5.43$; not significant).

In addition to the structured part, the questionnaire contained free questions, asking the participants about aspects they particularly liked or disliked. An aspect that was praised in most comments was that we provided photographs of the devices with highlighted ports. The negative counterpart were complaints about the text-to-speech engine used to read the instructions and the explanations, which were also raised by a vast majority of the participants. Comments about the system in general were all positive: "assists in a useful way", "this assistant would be great for my parents.", or "this assistance system is very useful, as it allows people without expertise to follow the instructions successfully". The comments also confirm our assessment that not the explanations themselves were perceived negatively, but the fact that they were mandatory: Some participants mentioned as positive that the explanations were completely optional, as one can simply proceed by clicking on "next" (they were not intended to be optional, but some people proceeded without reading them). Other comments suggested to present the explanation *before* the instruction it refers to, rather than afterwards. There were only five comments about the actual quality of the explanations. Four of them were positive, only one mentioned that explanations were complicated by mentioning the technical names for the cables. Some of the positive remarks were "explanations were good and useful, but the presented version was confusing due to the bad, automated voice" and "the explanations seem to be unnecessary at first glance, but they increase the understanding of what one does and strengthen the credibility of the system".

We conclude that a system that offers automated support is perceived very well, in particular by non-experts. Furthermore, explanation abilities appear to be an important and beneficial feature of assistance systems, while their proactive provision should be thoroughly considered.

[2] sum over 6 questions with good internal consistency of $\alpha = 0.84$

Conclusion

We presented a domain-independent architecture to implement plan-based assistance systems. It integrates planning capabilities with advanced dialog and interaction management components, which enable multi-modal communication skills of such systems. The approach offers a high degree of flexibility by involving plan repair mechanisms to assist users in cases where unexpected execution failures occur. Based on the architecture, we implemented a prototype system capable of assisting users in the task of setting up a complex home theater. We demonstrated the acceptance and usefulness of this system in an empirical evaluation.

Acknowledgment

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

References

Beetz, M.; Jain, D.; Mosenlechner, L.; Tenorth, M.; Kunze, L.; Blodow, N.; and Pangercic, D. 2012. Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proc. of the IEEE* 100(8):2454–2471.

Bercher, P.; Geier, T.; Richter, F.; and Biundo, S. 2013. On delete relaxation in partial-order causal-link planning. In *Proc. of the 25th IEEE Intl. Conf. on Tools with AI (ICTAI 2013)*, 674–681.

Bercher, P.; Geier, T.; and Biundo, S. 2013. Using state-based planning heuristics for partial-order causal-link planning. In *Proc. of the 36nd German Conf. on AI (KI 2013)*, 1–12.

Bernardini, S., and Porayska-Pomsta, K. 2013. Planning-based social partners for children with autism. In *Proc. of the 23rd Intl. Conf. on Automated Planning and Scheduling (ICAPS 2013)*, 362–370.

Bidot, J.; Biundo, S.; and Schattenberg, B. 2008. Plan repair in hybrid planning. In *Proc. of the 31st German Conf. on AI (KI 2008)*, 169–176.

Biundo, S., and Schattenberg, B. 2001. 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)*, 157–168.

Biundo, S.; Bercher, P.; Geier, T.; Müller, F.; and Schattenberg, B. 2011. Advanced user assistance based on AI planning. *Cognitive Systems Research* 12(3-4):219–236. Special Issue on Complex Cognition.

Boger, J.; Poupart, P.; Hoey, J.; Boutilier, C.; Fernie, G.; and Mihailidis, A. 2005. A decision-theoretic approach to task assistance for persons with dementia. In *Proc. of the 19th Intl. Joint Conf. on AI (IJCAI 2005)*, 1293–1299.

Elkawkagy, M.; Bercher, P.; Schattenberg, B.; and Biundo, S. 2012. Improving hierarchical planning performance by the use of landmarks. In *Proc. of the 26th Natl. Conf. on AI (AAAI 2012)*, 1763–1769.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the 2nd Intl. Conf. on AI Planning Systems (AIPS 1994)*, 249–254.

Geier, T., and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *Proc. of the 22nd Intl. Joint Conf. on AI (IJCAI 2011)*, 1955–1961.

Geier, T.; Reuter, S.; Dietmayer, K.; and Biundo, S. 2012. Track-person association using a first-order probabilistic model. In *Proc. of the 24th IEEE Intl. Conf. on Tools with AI (ICTAI 2012)*, 844–851.

Honold, F.; Schüssel, F.; Weber, M.; Nothdurft, F.; Bertrand, G.; and Minker, W. 2013. Context models for adaptive dialogs and multimodal interaction. In *Proc. of the 2013 9th Intl. Conf. on Intell. Env.'s (IE 2013)*, 57–64.

Honold, F.; Schüssel, F.; and Weber, M. 2012. Adaptive probabilistic fission for multimodal systems. In *Proc. of the 24th Australian Computer-Human Interaction Conference (OzCHI 2012)*, 222–231.

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

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *Proc. of the 19th Natl. Conf. on AI (AAAI 2004)*, 617–622.

Lim, B. Y.; Dey, A. K.; and Avrahami, D. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proc. of the SIGCHI Conf. on Human Factors in Comp. Systems (CHI 2009)*, 2119–2128.

Marthi, B.; Russell, S. J.; and Wolfe, J. 2008. Angelic hierarchical planning: Optimal and online algorithms. In *Proc. of the 18th Intl. Conf. on Automated Planning and Scheduling (ICAPS 2008)*, 222–231.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proc. of the 9th Natl. Conf. on AI (AAAI 1991)*, 634–639.

Muir, B. M. 1992. Trust in automation: Part i. theoretical issues in the study of trust and human intervention in automated systems. In *Ergonomics*, 1905–1922.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *AI* 76(1):427–454.

Nguyen, X., and Kambhampati, S. 2001. Reviving partial order planning. In *Proc. of the 17th Intl. Joint Conf. on AI (IJCAI 2001)*, volume 17, 459–466.

Nothdurft, F., and Minker, W. 2012. Using multimodal resources for explanation approaches in technical systems. In *Proc. of the 8th Conf. on Intl. Language Resources and Evaluation (LREC 2012)*, 411–415. European Language Resources Association (ELRA).

Nothdurft, F.; Bertrand, G.; Heinroth, T.; and Minker, W. 2010. GEEDI - Guards for Emotional and Explanatory DIalogues. In *6th Intl. Conf. on Intell. Env.'s (IE 2010)*, 90–95.

Parasuraman, R., and Riley, V. 1997. Humans and automation: Use, misuse, disuse, abuse. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 39(2):230–253.

Petrick, R., and Foster, M. E. 2013. Planning for social interaction in a robot bartender domain. In *Proc. of the 23rd Intl. Conf. on Automated Planning and Scheduling (ICAPS 2013)*, 389–397.

Porteous, J.; Cavazza, M.; and Charles, F. 2010. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Trans. Intell. Syst. Tech.* 10:1–10:21.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62(1-2):107–136.

Schröder, M. 2010. The SEMAINE API: towards a standards-based framework for building emotion-oriented systems. *Advances in Humam-Computer Interaction* 2010.

Schüssel, F.; Honold, F.; and Weber, M. 2013. Using the transferable belief model for multimodal input fusion in companion systems. In *Multimodal Pattern Recognition of Social Signals in Human-Computer-Interaction (MPRSS 2012)*, LNCS/LNAI 7742. 100–115.

Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users – a formal approach for generating sound explanations. In *Proc. of the 22nd Intl. Conf. on Automated Planning and Scheduling (ICAPS 2012)*, 225–233.

Tate, A.; Drabble, B.; and Kirby, R. 1994. O-plan2: an open architecture for command, planning and control. In *Intell. Scheduling*, 213–239.

Younes, H. L. S., and Simmons, R. G. 2003. VHPOP: Versatile heuristic partial order planner. *Journal of AI Research (JAIR)* 20:405–430.