To Plan for the User Is to Plan With the User — Integrating User Interaction Into the Planning Process

Gregor Behnke, Florian Nielsen, Marvin Schiller, Denis Ponomaryov,
Pascal Bercher, Birte Glimm, Wolfgang Minker and Susanne Biundo

Abstract Settings where systems and users work together to solve problems collaboratively are among the most challenging applications of Companion-Technology. So far we have seen how planning technology can be exploited to realize Companion-Systems that adapt flexibly to changes of the user's situation and environment and provide detailed help for users to realize their goals. However, such systems lack the capability to generate their plans in cooperation with the user. In this chapter we go one step further and describe how to involve the user directly into the planning process. This enables users to integrate their wishes and preferences into plans and helps the system to produce individual plans, which in turn let the Companion-System gain acceptance and trust from the user.

Such a Companion-System must be able to manage diverse interactions with a human user. A so-called mixed-initiative planning system integrates several Companion-Technologies which are described in this chapter. For example, a – not yet final – plan, including its flaws and solutions, must be presented to the user to provide a basis for her or his decision. We describe how a dialog manager can be constructed such that it can handle all communication with a user. Naturally, the dialog manager and the planner must use coherent models. We show how an ontology can be exploited to achieve such models. Finally we show how the causal information included in plans can be used to answer the questions a user might have about a plan.

Gregor Behnke · Marvin Schiller · Pascal Bercher · Birte Glimm · Susanne Biundo
Institute of Artificial Intelligence, James-Franck-Ring, 89081 Ulm, Germany
e-mail: firstname.lastname@uni-ulm.de

Florian Nielsen · Wolfgang Minker
Institute of Communications Engineering, Albert Einstein-Allee 43, 89081 Ulm, Germany
e-mail: florian.nothdurft@alumni.uni-ulm.de and wolfgang.minker@alumni.uni-ulm.de

Denis Ponomaryov
A.P. Ershov Institute of Informatics Systems, 6, Acad. Lavrentjev pr., Novosibirsk, 630090, Russia
e-mail: ponom@iis.nsk.su

The given capabilities of a system to integrate user decisions and to explain its own decisions to the user in an appropriate way are essential for systems that interact with human users.

1 Introduction

Planning has proven to be a successful technology for problem solving in scenarios involving humans and technical systems [5, 7, 22, 26]. Usually, a planner generates a plan to solve a given problem, e.g., to set up a home theater system (see Chap. 24), and presents the generated plan to the user in a stepwise fashion, while providing additional advanced planning capabilities, like explanations or plan repair. In this process the user is only viewed as an operator who inputs an objective and subsequently executes the actions presented to him, while the planner is treated as a black-box system. This scheme is well-suited if the task to be performed is combinatorially complex and a deeper understanding the proposed solution's structure is not relevant as long as the goal is achieved.

However, if the problem at hand is of a more personal nature, e.g., creating a fitness-training plan, or the user has certain preferences and wishes about the plan to be executed, or the user has domain knowledge not easily encodable in terms of planning actions, a black-box approach is not adequate. The user may not accept the plan in case it does not suit his individual needs or preferences. The same holds if the plan is associated with grave risks, e.g. in spaceflight [1] and military settings [25]. Here a human must be the final decider on which actions are actually executed. To circumvent these problems, the user has to be integrated into the planning process itself. Such planning systems are commonly called "mixed-initiative" as both the planner and the user propose courses of action and ask one another questions about the plan. As the result of their interplay the planner generates a final plan which solves the task and satisfies the user's wishes. We focus our discussion on the planning formalism *hybrid planning* [9], which is well suited for user-centered planning applications (see Chap. 5 and [3]). Most notably, it is similar to the way humans solve problems, i.e., in a top-down fashion [11].

Mixed-initiative planning (MIP) systems have already been studied by several researchers as they are often necessary to successfully deploy planning techniques to real-world problems. In the TRAINS/TRIPS project [13], a domain specific planner for path finding and transportation tasks was extended with the capability to interact with a user. Similarly MAPGEN [1, 6] was developed to support planning of operations of the mars-rovers Spirit and Opportunity while ensuring energy and safety constraints. Another approach, PASSAT [25], employs hierarchical planning. Here the user can "sketch" a plan, i.e., state some actions he wants to be part of the solution and the system finds a plan containing these actions. Further, PASSAT guides the user through the search space by repeatedly asking how a current plan should be altered until an accptable solution is found. In contrast, the techniques described in this chapter aim at integrating the user directly into the planning process.

In addition to a pure mixed-initiative planner a *Companion*-System [8] needs additional advanced capabilities to suitably interact with the user, including a dialog management system and explanation facilities. They need to be specifically tailored to the task of mixed-initiative planning, as they, e.g., must support dynamic changes in the current plan as well as changes of initiative from the system to the user. In the latter part of the chapter, we demonstrate how the planner and interaction components can be brought together with a mixed-initiative planner. An important part of such an integration is a shared model used by every component. We propose to use an ontology to store this model and describe how a planning model can be suitably encoded and which additional benefits can be obtained.

In this chapter we study the design of such a mixed-initiative planning system. To begin with, we outline in Sect. 3 challenges a mixed-initiative planning system has to tackle in order to successfully cooperate with a user and whether the issue should be addressed by the planner or by a dialog manager. Next, we show how a *Companion*-System can be built atop a mixed-initiative planner in Sect. 4. Section 5 explains how the common model of all components of the *Companion*-System can be created and how relevant information can be accessed. More specifically, we describe how a planning domain can be encoded in the ontology and how parts of the domain, i.e., decomposition methods, can even be inferred automatically. In Sect. 6 we study how users react to strategies applied by the dialog manager to mediate between the planner and the user. Then we discuss how explanations for plans can be enhanced to make them more comprehensible as described in Sect. 7.

We will use a fitness training domain as a running example. It is the mixed-initiative planner's objective to develop an individualized training plan achieving some fitness objective, e.g., to have well defined abdominal muscles. Exercises are grouped into workouts, which are to be performed on a single day and contribute to some fitness objective. We call a training a longer sequence of exercises that achieves a specific objective. A training is partitioned into several workouts, which are collections of exercises done on a single day of the training's schedule. The planner starts with a plan specifying a training task, representing the user's objective, and refines the plan repeatedly, first into suitable workout tasks and those in turn into concrete exercises. A similar application domain was considered by Pulido et al. [33] who described how a planner can be utilized to arrange physiotherapy exercises to rehabilitate people with upper limb injuries.

2 Preliminaries

The content of this chapter is based on the notions of hybrid planning and ontologies, which are both briefly introduced in this section. Chapter 5 explains hybrid planning in further detail and illustrates how it can be applied to user assistance in particular. In this chapter we show how the presented techniques can be complemented by integrating the user into the planning process itself.

Hybrid Planning. Planning is an AI technique for solving complex combinatorial problems. The objective of the planner is to find a so-called plan, a (partially ordered) set of actions, which, if executed in a given initial state, achieve some goal. States in planning are abstractions of the real world and are represented as sets of predicates. Actions are described in terms of their preconditions and effects, two formulae that must be true such that the action can be executed and that describe the change to the world state if the action is executed, respectively.

Hybrid planning is the fusion of two other planning approaches, namely Hierarchical Task Network (HTN [12]) and Partial Order Causal Link (POCL [24, 31]) planning. From the former it inherits the subdivision of actions into primitive and abstracts ones. Primitive actions are assumed to be directly executable by an operator, e.g. a human user, while abstract actions represent more complex courses of action. The aim in HTN planning is, given an initial set of abstract actions, to refine them into a plan solely containing primitive actions. To do so, an HTN planning domain contains so-called *decomposition methods* mapping abstract actions to plans—not necessarily containing only primitive tasks—by which they may be replaced. We use the expression $\mathtt{A} \mapsto_{\prec} \mathtt{B}_1, \ldots, \mathtt{B}_n$ to denote such a method for the abstract task $\mathtt{A}$ decomposing it into a plan containing the subtasks $\mathtt{B}_1$ to $\mathtt{B}_n$ which is ordered w.r.t. the partial order $\prec$. If the partial order $\prec$ is omitted, we assume that no order is present, i.e., $\prec = \emptyset$. POCL planning introduces the notion of causal links to hybrid planning. A causal link describes the relation between two actions, i.e., that one is executed to achieve an effect needed by the other. Furthermore, in standard HTN planning abstract actions have neither preconditions nor effects, whereas in hybrid planning they do. They enable causal reasoning about the plan at every level of abstraction and especially early during the planning process.

A planner for hybrid planning domains, PANDA, is presented in Chap. 5. Its purpose is to refine a given initial plan into a solution to the hybrid planning problem, i.e., to be an executable plan without abstract action that can be obtained via decomposition from the initial plan. It uses a heuristically guided plan-space search to find solutions. At each step during the search a so-called *flaw*, a property that keeps it from being a solution, is selected to be resolved. For instance, each abstract action in a plan constitutes a flaw, as well as preconditions of actions without supporting causal links. Thereafter, all possible modifications solving this flaw are applied to generate the plan's successors in the search space.

Ontologies. Ontologies based on Description Logics (DLs) serve to model knowledge in an application domain, with a focus on the *concepts* of a given domain and the relations (*roles*) that hold between them. The formal representation of a knowledge base enables the use of reasoners to infer additional knowledge that is logically implied and provides a model-theoretic semantics for the contents of such a knowledge base. One application of ontologies is the Semantic Web, in whose context the web ontology language OWL was established as a W3C standard. In this chapter, we consider the fragment of OWL corresponding to the DL $\mathcal{ALC}$ [35]. Concepts are either primitive (represented by a set of concept names) or complex. Complex concept expressions are formed using the connectives $\sqcap$ (conjunction of concepts) and $\sqcup$ (disjunction), and quantifier-like $\exists/\forall$ constructs, which specify re-

lationships between concepts with respect to a particular role. For instance, suppose that *includes* is a role name, and *HardExercise* is a concept name, then the expression $\exists$*includes.HardExercise* represents the concept of all things that include (at least) something that is a *HardExercise*. By contrast, the expression $\forall$*includes.HardExercise* represents the concept of all things that include nothing but *HardExercise*s. Two distinguished concepts, $\top$ and $\bot$, represent the universal concept (which encompasses every other concept) and the empty concept, respectively.

DLs are fragments of first-order logic and possess a model-theoretic semantics. Concepts are interpreted as subsets of a domain. A domain element that is in the interpretation of a concept is referred to as an instance of the concept. Roles are interpreted as binary relations in the domain. For example, to qualify as an instance of the concept $\exists$*includes.HardExercise*, a domain element needs to be related by the role *includes* to (at least) one instance of the concept *HardExercise*.

An *ontology* or *knowledge base* $\mathcal{O}$ specifies a finite set of axioms. Axioms of the form $C \sqsubseteq D$ are referred to as concept inclusion axioms (alternatively: subsumption axioms), and specify that every instance of the concept C is also an instance of the concept D. Equivalence axioms are of the form $C \equiv D$, and state that D subsumes C and vice versa. An interpretation that satisfies all axioms of a knowledge base $\mathcal{O}$ is called a *model* of $\mathcal{O}$. Using an ontology reasoner, a given knowledge base $\mathcal{O}$ can be queried about whether the subsumption relationship holds between two concepts C and D, namely whether the axiom $C \sqsubseteq D$ holds in any model of $\mathcal{O}$ (in which case the axiom is *entailed* by the knowledge base). Another type of query concerns whether a concept is *satisfiable* in $\mathcal{O}$, that is, whether a concept can have any instances (without leading to a contradiction).

The generation of natural-language output from ontologies (*ontology verbalization*) has traditionally focused on making the formalized content accessible to non-expert users (e.g. in the *NaturalOWL* system [2]). Recent work also aims at verbalizing ontology reasoning (in a stepwise manner), including [28] and [34].

3 Technology Concept and Design

In this section, we take a closer look at how automated planners and humans solve problems. We elucidate the major differences between them and describe how these can be handled either by altering the planner or by equipping a dialog manager with appropriate behavior.

User-friendly search strategies. As mentioned earlier, most automated planners employ efficient search strategies, like A^* or *greedy search*, guided by heuristics. These strategies visit search nodes, in our case yet unfinished plans, in an order determined by the heuristic starting with the most promising plan, e.g., the plan for which the heuristic estimate of the distance to a solution is minimal. Given a perfect heuristic, the planner would basically maintain always the same plan and alter it until a solution has been found. Since all heuristics computable in a reasonable time are necessarily imperfect, the planner does not necessarily visit plans after each other

which are neighbors in the search space, but may jump between completely separate parts of the search space. The considered plans may have nothing in common at all.

This alternating between plans is not in line with the human planning processes, as we tend to refine only one plan at a time. For example, in an experimental study on planning behavior, Byrne [11] found that subjects dealt with goals "one by one". Further research helped to put this finding in perspective, and postulates that the degree to which people adhere to a hierarchical top-down approach of sequential plan refinement depends on various factors; whether the problem domain can easily be recognized by people as hierarchically structured, whether human problem-solvers already dispose of expertise with hierarchical schemata to address problems in the domain, or—on the other hand—in how far they feel enticed to explore the domain by using bottom-up processes rather than a more straightforward top-down approach [17]. Thus, these empirical studies suggest that a structured planning process is a feature of skilled and goal-directed decision-making. In contrast, an automated planner's $A*$ strategies may seem erratic to the user. Such may result in the perception that user-decisions exert only an arbitrary influence on the planning process and may promote a lack of subjective control and transparency. In order to prevent this perception—which would most probably lead to the user not using the system—the gap between the way human and automated planners solve a planning problem needs to be bridged in a mixed-initiative planning system. Instead of A* or greedy search, we propose to use the search strategy *depth-first search* (DFS), which repeatedly refines a current plan until a solution has been found. If a plan is reached that cannot be further refined into a solution, e.g. a plan without a possible refinement, the decisions leading to this plan are reverted until another possible refinement is found. This continuous refinement of a single plan reflects the human process of problem solving much closer.

A drawback of DFS is that it is a blind search, i.e., it does not consider an estimate of how far a plan is away from being a solution when choosing a refinement. To remedy this problem, a mixed-initiative planner can consider a heuristic if the user is indifferent between options, or even weight the information of a heuristic against the decisions of the user. This scheme enables the user to inform the planner that it should use its best judgment to determine a solution for a subproblem, e.g. if it is combinatorially too complex to be solved by a human. DFS is also incomplete meaning that it may not find a solution even if it exists, as it can "get stuck" in infinite parts of the search space not containing a solution.

Handling unsolvable plans. Another difference is the way humans and planners deal with failed plans. During the search, the planner will explore plans that cannot possibly be refined into a solution anymore, either because the plan has a flaw without a possible modification to solve it or because a heuristic has determined this property. In this case, no successors are generated for the plan—it may still have resolvable flaws—and the search is continued. If the search procedure DFS is applied, this will lead to backtracking. As every practically usable computable heuristic is necessarily imperfect[1], there are usually whole parts of the search space

[1] Computing a perfect heuristic is as difficult as planning itself, e.g. in the case of HTN planning, undecidable [14].

only containing unsolvable plans which are not recognized as such. If, either by chance or due to the user's decisions, the search enters such a region, a failed plan will eventually be obtained and the backtracking procedure will be started, leading to the complete exploration of this unsolvable part of the search space.

On the other hand, if humans recognize that a plan is unsolvable, they can most of the time determine a reason for failure and alter only relevant parts of the plan. Current planning systems often fail in determining which applied modification was the reason for the failure. Instead they use backtracking to resolve the problem. *Backtracking*, especially through a large search space, is very tedious and frustrating for a human user performing or witnessing these steps. A large search space requires extensive backtracking through unsolvable alternatives. This may result in the user interpreting the system's strategy as naive and impairs the trust and perceived competence of the planner. Additionally, this strategy does not prevent the repetition of similar (unsuccessful) decisions, leading furthermore to frustration.

To remedy this problem, we can utilize the computational power of the planner. If options for a refinement are presented to the user he usually takes several seconds (if not longer) to decide for one of the options. During this time, the planner can start to explore the search spaces induced by the modifications presented to the user. The planner may determine, using a well-informed heuristic, that the search space induced by one of the options only leads to failed plans. We call such a modification a *dead-end* and if it occurs, the respective option can be removed from consideration and thus backtracking through this part of the search space can be averted. Here the important question is how this information should be communicated to the user, as simply disabling the respective option in the user interface without any apparent reason would be rather irritating and seems not to be appropriate. If the planner, on the other hand, has found a solution, the mixed-initiative planning system knows that backtracking is not necessary if the user agrees with the solution. We describe our approach in the next section and evaluate it in Sect. 6.

4 Integration of Planning and Dialog

The integration of automated planning and user-centered dialog begins with the statement of the user's goals. This first dialog between user and machine has the objective of defining the goals in a way understandable for the assisting automated planning system. This requires on the one hand a user-friendly and efficient task-selection dialog, and on the other hand the creation of a valid *planning problem*. Thus, the semantics of the dialog have to be coherent with the *planning domain*, resulting in a valid mapping between dialog result and *planning problem*.

Once the problem is passed on to the planner the interactive planning itself may start. Using the described DFS the initial plan will be refined by selecting appropriate modifications for available flaws. In order to decide whether to integrate the user or not during this process, an elaborate decision model, integrating various information sources, is required. Relevant information sources are, for example, the *dialog*

history, e.g., was the user's decision the same for all past similar episodes, the kind of *plan flaw*, e.g., is this flaw relevant for the user, the *user profile*, e.g., does the user have the competencies for this decision, or the current *situation*, e.g. is the current cognitive load of the user low enough for interaction. These sources illustrate that a decision model uses information from the *dialog management* and the *planner*, and is therefore located in a superordinate component.

In case of user integration the information on the current *plan decision* has to be communicated to the user. This means that the *plan flaw* and the corresponding decision between the available *modifications* have to be represented in the dialog suitably. Hence, the corresponding plan information needs to be mapped to human-understandable dialog information. As this mapping potentially needs to exist for every plan information and for every dialog information, the requirement of coherent models between planner and dialog system becomes an existential factor for MIP systems. The thorough matching of both models would be an intricate and strenuous process, requiring constant maintenance, especially when a model needs to be updated. Thus, a more appropriate approach is the automatic generation of the respective models using one mutual model as source. This way, once the transformation functions work correctly, coherence is not an issue any more, even when updating the domain. How these essential constituents of a conceptual MIP system architecture (depicted in Fig. 1) were implemented in our system is explained below.

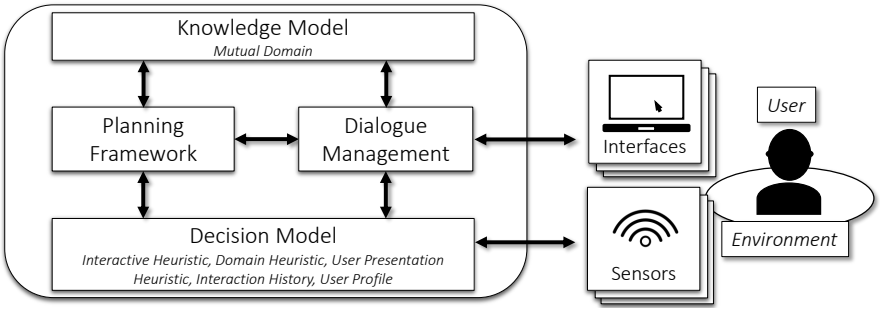

Fig. 1 Essential components of a mixed-initiative planning system integrating the user [29]

The decision model. This model is in charge of deciding when and how to involve the user into the planning process. It is composed of several subcomponents, acts as an interface to the planner and decides, upon planner requests, whether a user involvement is useful, i.e., if this kind of flaw is understandable to a human user.

For this it also includes a list of essential domain decisions that are interesting and relevant for the user (e.g. for a training domain: day, workout, and exercises)—the rest is left for the fallback-heuristic and thus decided by the planner. If it is in favor of user involvement, the flaw and its corresponding modifications have to be passed on to the user. Then, the decision on the form of user integration is made. The dialog may either provide the complete set of modifications, a pruned list, a sorted list, implicit confirmations, explicit confirmations, for presentation or only to

inform the user. This decision depends not only on the interaction history, but also on additional information (e.g. affective user states like overextension, interest, or engagement) stored in the user state.

The *Decision Model* also records the dialog- and planning history. There are several reasons for that: The dialog history may enable a prediction of future user behavior (e.g. in selections), and additionally this knowledge is mandatory for *backtracking* processes, when the current plan does not lead to a solution. The history stores which decisions were made by the user. In case of *backtracking* the decisions are undone step-by-step, with the goal of finding a solution by applying alternative modifications. Whenever a user-made decision is undone, the user is notified, because this system behavior would otherwise appear irritating.

Since *backtracking* as well as *dead-ends* are peculiar phenomena in a MIP system, the communication of these might have a critical influence on the user experience. Together with the *Dialog Management (DM)*, the *Decision Model* orchestrates the corresponding system behavior. The main difference between *backtracking* and *dead-ends* is the temporal ordering of the awareness of the unsolvable plan and made decision. For *backtracking* the awareness is achieved after the decision, and for *dead-ends* during the decision. As we assumed that *backtracking* will impair the user experience significantly, a parallel search for *dead-ends*, as described in Sect. 3, was implemented. The process itself is, of course, inherently different from *backtracking*, but may prevent it. Removing dead-ends from the search space when the relevant modification is not part of the current selection is a rather easy task. Otherwise, the current selection has to be modified to prevent the user from selecting a *dead-end*. However, removing it without any notification from the list seems like a confusing behavior.

5 Coherent Models Across the System

The system described in the previous section relies on a shared vocabulary and a coherent description of the planning domain in both the dialog system (DS) and the planner. In this section we describe how this coherence can be achieved using an ontology as the central knowledge-base component. To do so, relevant[2] parts of the planning domain are encoded in description logic. To obtain a unified view of the system's knowledge we also demonstrate how the remaining planning specific information can be encoded in a way not interfering with the reasoning process, and thus how all information can be stored in the ontology. Further we describe how planning and dialog domain can be automatically extracted from the ontology.

As an additional advantage, new decomposition methods for the planning domain can be inferred using ontology reasoning without the help of a human expert modeler, easing creating domains significantly. This is especially useful in our applica-

[2] those which must be accessible by other systems

tion scenario—individualized fitness training. Imagine a user found a new workout[3], e.g. while browsing the Internet, and wishes to add it to his training plan[4].

Using ontological reasoning, the system can infer which training objective the workout has and how it can be integrated into the existing planning domain, without additional input from the user. Furthermore, using plan and ontology explanations the *Companion*-System can explain how and why it has integrated the new workout into the model in a certain way. If the workout does not comply with the user's objective it could even explain why the user should not use the workout.

A few previous approaches have attempted to integrate ontological reasoning into planning. Most approaches targeting classical, i.e., non-hierarchical, planning attempt to either increase the performance of the planner or increase the expressivity of the planning formalism, e.g. by changing the notion of states. We refer to an article by Gil [15] for an extensive survey. Another approach by Sirin et al. [37, 36], called HTN-DL, uses ontology reasoning to solve web service composition problems encoded in an HTN. Their main objective is to determine which abstract tasks can be decomposed by a predefined plan, based on annotations to that plan. In that, they infer new decomposition methods, but only those for which the plan's task network was provided by the domain modeler. Their matching cannot take the actual content of the plan into account, but only an abstract description of the plan in terms of preconditions and effects. Furthermore, there is no guarantee on the relation of the plan's content and these descriptions. One could, e.g., use a legality criterion for decomposition methods [9] to determine whether the description is correct. Our approach, on the other hand, can infer completely new decomposition methods and is able to infer them based on the actual plan's steps to be contained in them.

5.1 Integrating Planning Knowledge into Ontologies

We start by describing how a planning domain can be encoded in an ontology. Tasks in the planning domain are represented as concepts in the ontology. For each planning task `T` there is a corresponding concept T in the ontology. Preconditions and effects of actions are encoded in the ontology using four distinct data properties: *needs* for positive preconditions, *hindered-by* for negative preconditions, and *adds* and *deletes* for positive and negative preconditions, respectively. Here, only the predicates of preconditions and effects are contained in the ontology, while their parameters are omitted. Expressing them correctly, i.e. in a way amenable to logical reasoning, would require common references, e.g., for an action requiring $has(x)$ and resulting $\neg has(x)$ where both instances of x must be equal. Unfortunately, description logics are not suited for such kinds of expressions, due to the tree model property [39]. One example of such an action is the `Day` action, describing the transition of a day to the next one, i.e. modeling time explicitly in the domain.

[3] a partially-ordered set of exercises

[4] A system that supports this search-and-extraction, resulting in an extension to the ontology, has been developed but is not yet published.

$$Day \equiv \exists deletes.''fatigue''$$

This concept describes an action without any preconditions, leading to a state where the predicate `fatigue` does not hold after the action has been executed, for some parameters. In this case it models that a person is not fatigued after he has slept, i.e., a day has passed. Additionally our approach allows for domain-depended extensions. They define that certain axioms $C \sqsubseteq E$, where C is a concept and E an expression, are interpreted as a set of preconditions and effects. In our application scenario, we used this mechanism to map declarative descriptions of exercises to actions. These descriptions are based on parts of the NCICB corpus [27], describing the musculoskeletal system of the human body. As an example, we provide the following definition of the exercise *BarbellHackSquat*, which describes that the action `BarbellHackSquat` has the precondition *warmedup(Hamstring)* and the effects *trained(Hamstring)*, *warmedup(Soleus)*, and *warmedup(GluteusMaximus)*.

$$\begin{aligned} BarbellHackSquat \sqsubseteq\ & \exists trains.Hamstring \sqcap \exists engages.Soleus \\ & \sqcap \exists engages.GluteusMaximus \end{aligned}$$

The most important part of a hierarchical planning domain are its decomposition methods. Each method $\mathtt{A} \mapsto_{\prec} \mathtt{B}_1, \ldots, \mathtt{B}_n$ describes that a certain abstract task `A` can be achieved by executing the tasks $\mathtt{B}_1, \ldots, \mathtt{B}_n$ under a restriction $\prec$ on their order. To transform decompositions into ontological structures consistently, we state an intuition on the meaning of the concept-individual relation in the created ontology, by which our subsequent modeling decisions are guided. Individuals in the ontology should be interpreted as plans and a concept T, corresponding to some task `T`, as the set of all plans (i.e. individuals) that can be obtained by repeatedly decomposing `T`. A concept inclusion $B \sqsubseteq A$ thus states that every plan obtainable from `B` can also be obtained by decomposing `A`. To ease modeling, a special role—*includes*—is designated to describe that a plan contains some other plan. That is, if an individual a is an instance of a plan and $includes(a,b)$ holds, the plan described by the individual a also contains all actions of the plan b. First, so-called unit-methods $\mathtt{A} \mapsto \mathtt{B}$, which allow for replacing the task `A` with the task `B`, are translated into a simple concept inclusion $B \sqsubseteq A$, for which the intuition clearly holds. Methods creating more than a single task must be translated into more elaborated constructs in the ontology. Such a method $\mathtt{A} \mapsto_{\prec} \mathtt{B}_1, \ldots, \mathtt{B}_n$ defines a set of tasks $\{\mathtt{B}_1, \ldots, \mathtt{B}_n\}$ which must be contained in the plan obtained from decomposing `A`, while also stating that they are sufficient. The relation "is contained in a plan" is expressed by the role *includes*. Following its description, an expression $\exists includes.T$ describes the set of all plans containing the task `T`. Using this property, the decomposition method could be translated into the following axiom

$$\prod_{i=1}^{n} \exists includes.B_i \sqsubseteq A$$

It is, however, not sufficient due to the open world assumption of description logics. The expression solely describes the required tasks, but not that only these tasks are

contained in the plan. To express the latter, we use the syntactic *onlysome* quantifier, originally introduced for OWL's Manchester syntax [20].

Definition 1. Let r be a role and $C_1, \ldots, C_n$ concept expressions. Then we define the onlysome quantification of r over $C_1, \ldots, C_n$ by

$$\mathrm{O}r.[C_1, \ldots, C_n] := \sqcap_{i=1}^{n} \exists r.C_i \sqcap \forall r\,(\sqcup_{i=1}^{n} C_i)$$

As an example in the fitness domain, consider a method that specifies that a particular workout decomposes into the set of tasks represented by its exercises, for example `Workout1` $\mapsto$ `FrontSquat,BarbellDeadlift`. This can be specified in the ontology as $\mathrm{O}\mathit{includes}.[\mathit{FrontSquat}, \mathit{BarbellDeadlift}] \sqsubseteq \mathit{Workout1}$. In general, with this definition we can describe a decomposition method $\mathtt{A} \mapsto_{\prec} \mathtt{B}_1, \ldots, \mathtt{B}_n$ with the following axiom, fulfilling the stated requirement.

$$\mathrm{O}\mathit{includes}.[B_1, \ldots, B_n] \sqsubseteq A$$

The intuition on the interpretation of concepts and individuals implies a criterion when two concepts A and B, described by such axioms, should subsume each other. That is, A should subsume B based on the axioms in the ontology, if and only if every plan described by B is also a plan for A. We have stated this criterion previously and proven that it holds with only minor restrictions to the ontology for expressions defined in terms of *onlysome* restrictions [4]. So far, we have not mentioned possible ordering constraints imposed on the tasks in a method. Representing them s.t. that they can be accessed by a DL reasoner poses a problem similar to representing variables, as a single task may be referred to several times in describing a partial order. To circumvent this problem ordering is encoded only syntactically, i.e., in a way ignored by any reasoner while it still can be retrieved from the ontology by analyzing its axioms. If a plan contains some task `A` before some task `B`, then any occurrence of B in an onlysome restriction is substituted with $B \sqcup (\bot \sqcap \exists \mathit{after}.A)$.

5.2 Generating New Decomposition Methods Using DL Reasoning

Having integrated the planning domain into an ontology, we can utilize DL reasoning to infer new decomposition methods. Based on the interpretation of concept subsumption, each inferred subsumption $E \sqsubseteq A$ can be interpreted as the fact that every plan obtainable from the expression E can also be obtained from the abstract task A. If it is possible to associate E with a distinct plan P we can add a method $\mathtt{A} \mapsto P$ to the planning model. The task of ontology classification is to find the subsumption hierarchy of the given ontology, i.e., all subsumptions between named concepts occurring in the ontology. They have the form $B \sqsubseteq A$ and can be easily transformed into unit-methods $\mathtt{A} \mapsto \mathtt{B}$. Taking *Workout1* as an example, if it can be inferred that *Workout1* $\sqsubseteq$ *StrengthTraining* (*Workout1* is a strength training), a method `StrengthTraining` $\mapsto$ `Workout1` is created. A more challenging task

is to find more complex decomposition methods, described by concept inclusions between named concepts (i.e. the task to be decomposed) and expressions describing plans. For example, is the combination of *Workout1* with another workout classified as a *StrengthTraining*?

Since generating all possible subsumptions between concepts and arbitrary expressions is impossible in practice, only a certain set of candidate expressions should be considered. The easiest way to do so is to add new named concepts $C \equiv E_C$ for every candidate expression E_C to the ontology. This enables a uniform scheme to generate new decomposition methods. First, a set of candidate expressions is generated and added as named concepts to the ontology. Second, ontology classification is used to determine all subsumptions between named concepts in the ontology. Third, these subsumptions are translated into decomposition methods. This is done for subsumptions that connect two named concepts from the original ontology, and subsumptions between a named concept and the concepts in a newly generated candidate expression.

Behnke et al. [4] described which expressions should be considered as potential candidate concepts. Most notably, they argued that a syntactic combination of concepts defined by onlysome expressions should be defined. Our fitness training domain initially contains 310 tasks and only a few methods, while the corresponding ontology contains 1230 concepts (of which 613 are imported from the NCICB corpus) and 2903 axioms (of which 664 are from NCICB). Using DL reasoning—provided by the OWL reasoner FaCT++ [38]—206 new decomposition methods are created. On an up-to-date laptop computer (Intel® Core™ i5-4300U) it takes 3.6 seconds to compute the whole extended planning domain.

5.3 Dialog Domain

In order to integrate the user into the planning process and to communicate the generated solution, a dialog management component is needed to control the flow and the structure of the interaction. In order to communicate a solution, all planned tasks have to be represented in the dialog domain, while integrating the user requires the ongoing presentation of planning decisions. This includes, most notably, the choice of a decomposition method if an abstract task is to be refined. The use of shared knowledge considerably facilitates coherency of the interaction. Although the planning knowledge stored in the ontology alone is not sufficient for the generation of the dialog domain, it contributes to its structure and enables an unisono view on the domain, eliminating inconsistency and translation problems.

The integrated planning knowledge, used to infer new decompositions for existing planning domains, can be used to create a basic dialog structure as well. Analogous to Sect. 5.1, a dialog $\mathtt{A}$ can be decomposed into a sequence of subdialogs containing the dialogs $\mathtt{B}_1, \ldots, \mathtt{B}_n$ by an axiom $\mathrm{O}\mathit{includes}[B_1, \ldots, B_n] \sqsubseteq A$. For example, in our application scenario a strength training can be conducted using a set of workouts $\mathtt{A}_1, \ldots, \mathtt{A}_m$, each of which consists of a set of exercises $\mathtt{B}_1, \ldots, \mathtt{B}_n$.

This way a dialog hierarchy can be created, using the topmost elements as entry points for the dialog between user and machine. Nevertheless, this results only in a *valid* dialog structure, but not in a *most suitable* one for the individual user. For this, concepts of the ontology can be excluded from the domain generation or conjugated to other elements in a XML configuration file. This way elements can be hidden or rearranged for the user. The dialogs are also relevant during the MIP process. When selecting between several *Plan Modifications*, these have to be translated to a format understandable by the user. Hence, in addition to the knowledge used to generate plan steps, resources are required for communicating these steps to the user. Therefore, texts, pictures, or videos are needed, which can be easily referenced from an ontology. Using this information, dialogs suitable for a well-understandable human-computer interaction can be created and presented to the user.

One key aspect of state-of-the-art DS is the ability to individualize the ongoing dialog according to the user's needs, requirements, preferences, or history of interaction. Coupling the generation of the dialog domain to the ontology enables us to accomplish these requirements using ontological reasoning and explanation in various ways as follows: Dialogs can be pruned using ontological reasoning according to the user's needs (e. g. "show only exercises which do not require gym access"), to the user's requirements (e. g. "show only beginner exercises") or adapted to the user's dialog history (e. g. "preselect exercises which were used the last time") and preferences (e. g. "present only exercises with dumbbells").

6 Explanations for Plans and Planning Behavior

Integrating proactive as well as requested explanations into the interaction is an important part of imparting used domain knowledge and clarifying system behavior. Using a coherent knowledge source to create dialog and planning domains enables us to use predefined declarative explanations [30] together with the dynamically generated plan explanations described in Chap. 5 and explanations for ontological inferences without dealing with inconsistency issues. This way *Plan Steps* (e.g. exercises) can be explained in detail, dependencies between plan steps can be explained to exemplify the necessity of tasks (i.e. plan explanation), and ontology explanations can justify inferences from which the planning model and the dialog domain where generated. All of which increase the user's perceived system transparency. In the scope of mixed-initiative planning, events like *backtracking* may confuse the user. Especially in this context we deem the integration of explanations a very valuable system capability. In order to investigate the effects of the use of explanations in MIP we designed an experiment comparing different kinds of explanations. As context a typical *backtracking* situation was chosen, initiated by external information.

Methodology

Participants were presented a scenario where they were tasked to create individual strength training workouts. They were guided through the process by the system, which provided a selection of exercises for training each specific muscle group necessary for the workout. Figure 2 shows an exemplary course of interaction for the introductory round. Here, the user had to plan a *Full Body Workout* to train the listed body parts. For each body part a dialog was presented, providing the user with a selection of exercises to train the specific body part. For example, when training the *legs* the user could choose from exercises such as the *barbell squat* or the *dumbbell squat* (see Fig. 3).

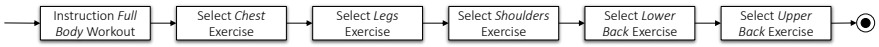

Fig. 2 Sequence of (sub-)dialogs for planning the introductory workout

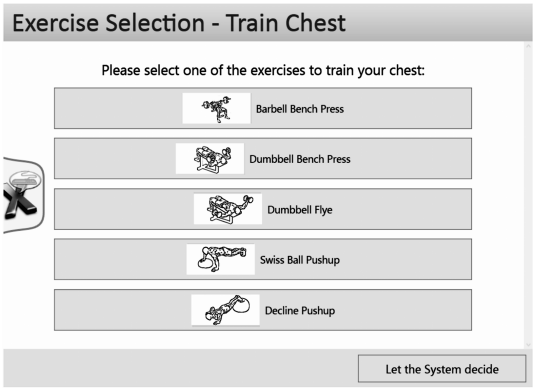

Fig. 3 A screenshot of a typical selection dialog (we employ a system developed by Honold et al. [18]). The user is prompted to select an exercise to train the legs or to "let the system decide", in which case the exercise is selected at random.

The experimental design compared two conditions, distinguished by the use of explanations. During the session, *backtracking* was initiated due to an artificially induced event. In the first condition (backtracking with notification, BT-N), a context-independent high-level explanation was provided:

> "The system has detected that the previously presented options do not lead to a solution. Therefore, you have to decide again."

In the second condition (backtracking with explanation, BT-E), a more concrete description of the reason for backtracking (the external event) was provided:

> "The system has detected that the gym is closed today due to a severe water damage. Therefore, you have to decide again and select exercises suitable for training at home."

Participants were assigned randomly to conditions. Due to incomplete data some had to be removed resulting in 43 participants (25 to BT-N, 18 to BT-E). Measures were obtained after the interaction for the following variables:

Human-Computer Trust (HCT) describes the trust relationship between human and computer and was assessed using the questionnaire by Madsen and Gregor [23] measuring five dimensions (*Perceived Understandability, Perceived Reliability, Perceived Technical Competence, Personal Attachment, Faith*).

AttrakDiff assesses the perceived pragmatic quality, the hedonic qualities of stimulation and identity, and the general attractiveness of dialog systems (or software in general). We used the questionnaire developed by Hassenzahl et al. [16].

Cognitive Load is assessed using an experimental questionnaire developed by Pichler et al. [32] which measures all three types of cognitive load (intrinsic, extraneous and germane cognitive load) separately, along with the overall experienced cognitive load, fun and difficulty of the tasks.

Results

A pairwise t-test on the uniformly distributed data revealed significantly higher scores for the HCT items *perceived reliability* ($t(3.0) = 57, p = .004$), *perceived understandability* ($t(3.99) = 57, p = .000$) and *perceived technical competence* ($t(2.06) = 41, p = .045$) in the BT-E condition as compared to BT-N. For the cognitive load questionnaires, we only found that the germane load was higher for the BT-E condition, but only to a marginally significant degree ($t(1.99) = 41, p = .053$). Note that germane load is positive load—it occurs in the processes inherent in the construction and automation of schemas (e.g. building mental models). In the AttrakDiff we observed a significantly higher score for BT-E in the dimension of experienced *pragmatic qualities* ($t(2.37) = 41, p = .022$), which can be attributed to significant differences in the subscales *unpredictable - predictable, confusing - clearly structured, unruly - manageable* and *unpleasant - pleasant*.

These results strengthen our hypothesis that providing explanations of system behavior, in this case of backtracking, does indeed help to perceive the system as more reliable, more understandable, and more technically competent. It seems that the explanations kept the user motivated, compared to a more frustrating experience of receiving no explanation for the impairing system behavior for BT-N. The findings concerning AttrakDiff provide evidence for the conjecture that systems with explanation capabilities seem to be perceived as not so complicated, more predictable, manageable, more clearly structured and in general as more pleasant. However, providing explanations of internal system processes, which increase the transparency of the system, requires corresponding reasoning capabilities. This includes the ability to explain causal dependencies between tasks and their related hierarchical structure (i.e. decomposition methods), using *plan explanations* (cf. Chap. 5) and extensions thereof discussed in the following.

7 Extending Plan Explanations with Ontology Explanations

Plan explanations focus on elaborating the structure of a generated plan; namely the relationships between preconditions and effects and how primitive tasks are obtained by decomposing abstract tasks. However, as described in Sect. 5.1, task decompositions are inferred from domain knowledge in the ontology, which lends itself to more detailed explanations. For example, rather than presenting a task decomposition only in the form of a statement such as "Task A was necessary, since it must be executed to achieve B", the assumptions and the reasoning behind this decomposition can be used to further elucidate *why* such a task A serves to achieve B. One advantage of using a description logics formalism for this knowledge is the opportunity to make use of *ontology verbalization* techniques that have been devised to generate texts and explanations from ontologies that are understandable to lay people. A considerable part of related work in this field has so far concentrated on how selected facts from an ontology can be presented to users in a fluent manner, and how well they are understood, for example [2, 21]. Approaches to the generation of explanations for reasoning steps have been developed [10, 28, 34]. The work presented here is in line with the principles common to these three approaches. Whereas in the following, we illustrate our approach using the explanation of a task decomposition as an example, also other logical relationships between facts modeled in the ontology can be explained with the help of the presented techniques. The presented mechanism has been implemented as a prototype—developing the prototype into a mature system remains a topic for future work.

Explanations are generated for facts that can be inferred from the axioms in the ontology. For a short example, consider a workout that includes two tasks, front squat and barbell deadlift (each to be performed for a medium number of repetitions), called *Workout1*. Further assume that with the background knowledge of the ontology, *Workout1* is classified as a strength training (i.e. $\mathit{Workout1} \sqsubseteq \mathit{StrengthTraining}$ holds), which—as discussed in Sect. 5.1—introduces a decomposition $\texttt{StrengthTraining} \mapsto \texttt{Workout1}$. The user may now ask for a justification for the decomposition, which is provided based on the axioms and the logical relationships that served to classify *Workout1* as a strength training. These explanations are generated in a stepwise fashion from a formal proof, such that the individual inference steps (in particular, including intermediate facts) are translated to simple natural language texts. In our running example, the resulting explanation is the following:

> A strength training is defined as something that has strength as an intended health outcome. Something that includes an isotonic exercise and a low or medium number of repetitions has strength as an intended health outcome, therefore being a strength training. Workout1 is defined as something that strictly includes a medium number of repetitions of front squat and a medium number of repetitions of barbell deadlift. In particular Workout1 includes a medium number of repetitions of front squat. Given that front squat is an isotonic exercise, a medium number of repetitions of front squat is an isotonic exercise. Thus, we have established that Workout1 includes an isotonic exercise and a low or medium number of repetitions. Given that something that includes an isotonic exercise and a low or medium number of repetitions is a strength training, Workout1 is a strength training.

In the following, we discuss the two submechanisms involved in generating such an explanation—reasoning and explanation generation—in more detail.

Reasoning. The first step consists of identifying those axioms that are logically sufficient to show (and thus explain) the fact in question. This task is known as *axiom pinpointing*, for which an efficient approach has been developed by Horridge [19]. The set of necessary axioms (which needs not be unique) is called a *justification* for the inferred fact. We simply use Horridge's mechanism as a preprocessing step to obtain a set of relevant axioms to infer the fact in question, and then use inference rules implemented in the prototype to build a stepwise (also called consequence-based) proof. As of current, the prototype uses inference rules from various sources (e.g. [28]) and has not been optimized for efficiency (working on the justifications instead of the full ontology makes using such a simple mechanism feasible).

Explanation Generation. The generated proofs have a tree structure, which is used to structure the argument. Each inference rule provides a template, according to which textual output is produced. As a general rule, first the derivation of the premises of an inference rule needs to be explained before the conclusion may be presented (this corresponds to a post-order traversal, with the conclusion at the root of the tree). Inference rules with more than one premise specify the order in which the premises are to be discussed, which is determined by the logical form of the premises. For example, for the following inference rule, the derivation for the left premise is presented before the derivation of the right premise, though logically, the order of the premises is irrelevant for the validity of the proof.

$$\frac{\begin{array}{cc}\vdots & \vdots \\ X \sqsubseteq \exists r.B & B \sqsubseteq C\end{array}}{X \sqsubseteq \exists r.C}\, R^{+}_{\exists}$$

Lexicalisation of facts is done with the help of the ontology. For each concept name or role name (or other named elements in the ontology) the value of the *label* attribute is used as a lexical entry. Thus, the domain modeler is required to specify adequate names for the elements of the domain when modeling them in the ontology, for example, "isotonic exercise" for the concept name *IsotonicExercise*. A special case is represented by concepts that can be used as attributes in complex concept expressions. For instance, consider *MediumNumberOfRepetitions* (things that are repeated a moderate number of times) and *FrontSquat*, which can be combined to *MediumNumberOfRepetitions*$\sqcap$*FrontSquat*. Since *MediumNumberOfRepetitions* can be used on its own, the label specifies a name that can stand on its own, such as "medium number of repetitions". However, the concept needs to be combined adequately when in combination, for which a second type of label is used, in this case "medium number of repetitions of", such that the combination reads as "medium number of repetitions of front squat". In formulae, concept names are generally represented in indeterminate form, e.g., "a medium number of repetitions of front squat". Connectives in formulae are translated as $\sqsubseteq$: "is", $\sqcap$: "and" (unless the concepts can be combined as described above), $\sqcup$: "or", $\exists$: "something that", etc. This way, texts are generated similar to those studied in [21] and [28]. While this template-based approach is in general similar to previous related work (e.g.

[28]), some mechanisms and templates are specific to our approach. For example, our modeling relies on the *onlysome* macro, which represents a rather lengthy formula when expanded. When generating output, class expressions that correspond to the form of *onlysome* are identified and treated using a succinct text template to avoid being repetitive; in the example above, this is done for the third sentence, in which the onlysome statement O*includes*.$[$*MediumNumberOfRepetitions* $\sqcap$ *FrontSquat*, *MediumNumberOfRepetitions* $\sqcap$ *BarbellDeadlift*$]$ is verbalized as: "something that strictly includes a medium number of repetitions of front squat and a medium number of repetitions of barbell deadlift". Without this macro, the verbalization of the logically equivalent statement would state both the existence of and the restriction to these two exercises separately, and can thus be considered repetitive.

8 Conclusion

This chapter discussed a nexus of considerations and techniques that can form the basis of a planning system in which the user actively takes part in the planning process. For this purpose, we put priority on technologies that serve to make the planning process *comprehensible* and that emphasize its structure, by identifying adequate search strategies and techniques for handling flaws and dead-ends. Furthermore, we addressed the requirements of such a system to provide a coherent view on its domain knowledge. To this end, we developed an integration of the planning domain with an ontology to provide a central knowledge component for such an interactive system, such that planning is suitably linked with reasoning and dialog management. A further aspect considered important for the empowerment of the user concerns the provision of explanations. In addition to enabling the user to effectively participate in the problem-solving process, explanations increase the perceived reliability, understandability and competence of such a system, as was shown in the presented experiment. We discussed how different kinds of explanations (in particular, of reasoning steps) help to realize a dedicated user-oriented approach to planning, and outlined the scope for individualizing the planning process and the system's communication to address users' preferences.

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

References

1. Ai-Chang, M., Bresina, J., Charest, L., Chase, A., Hsu, J.J., Jonsson, A., Kanefsky, B., Morris, P., Rajan, K., Yglesias, J., Chafin, B., Dias, W., Maldague, P.: MAPGEN: Mixed-initiative planning and scheduling for the Mars exploration rover mission. Intelligent Systems, IEEE **19**(1), 8–12 (2004)

2. Androutsopoulos, I., Lampouras, G., Galanis, D.: Generating natural language descriptions from OWL ontologies: The NaturalOWL system. Journal of Artificial Intelligence Research (JAIR) **48**, 671–715 (2013)
3. Behnke, G., Höller, D., Bercher, P., Biundo, S.: Change the plan – how hard can that be? In: Proc. of the 26th Int. Conf. on Automated Planning and Scheduling (ICAPS). AAAI Press (2016)
4. Behnke, G., Ponomaryov, D., Schiller, M., Bercher, P., Nothdurft, F., Glimm, B., Biundo, S.: Coherence across components in cognitive systems – One ontology to rule them all. In: Proc. of the 24th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 1442–1449. AAAI Press (2015)
5. Bercher, P., Biundo, S., Geier, T., Hoernle, T., Nothdurft, F., Richter, F., Schattenberg, B.: 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), pp. 386–394. AAAI Press (2014)
6. Bercher, P., Höller, D.: Interview with David E. Smith. Künstliche Intelligenz (2016). DOI 10.1007/s13218-015-0403-y. Special Issue on Companion Technologies
7. Bercher, P., Richter, F., Hörnle, T., Geier, T., Höller, D., Behnke, G., Nothdurft, F., Honold, F., Minker, W., Weber, M., Biundo, S.: A planning-based assistance system for setting up a home theater. In: Proc. of the 29th Nat. Conf. on Artificial Intelligence (AAAI). AAAI Press (2015)
8. Biundo, S., Höller, D., Schattenberg, B., Bercher, P.: Companion-technology: An overview. Künstliche Intelligenz (2016). DOI 10.1007/s13218-015-0419-3. Special Issue on Companion Technologies
9. Biundo, S., Schattenberg, B.: 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), pp. 157–168. AAAI Press (2001)
10. Borgida, A., Franconi, E., Horrocks, I.: Explaining ALC subsumption. In: Proc. of the 14th European Conf. on Artificial Intelligence (ECAI), pp. 209–213. IOS Press (2000)
11. Byrne, R.: Planning meals: Problem solving on a real data-base. Cognition **5**, 287–332 (1977)
12. Erol, K., Hendler, J.A., Nau, D.S.: UMCP: A sound and complete procedure for hierarchical task-network planning. In: Proc. of the 2nd Int. Conf. on Artificial Intelligence Planning Systems (AIPS), pp. 249–254. AAAI Press (1994)
13. Ferguson, G., Allen, J.F.: TRIPS: An integrated intelligent problem-solving assistant. In: Proc. of the 15h Nat. Conf. on Artificial Intelligence (AAAI), pp. 567–572. AAAI Press (1998)
14. Geier, T., Bercher, P.: On the decidability of HTN planning with task insertion. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 1955–1961. AAAI Press (2011)
15. Gil, Y.: Description logics and planning. AI Magazine **26**(2), 73–84 (2005)
16. Hassenzahl, M., Burmester, M., Koller, F.: AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In: Mensch & Computer 2003: Interaktion in Bewegung, pp. 187–196. Teubner (2003)
17. Hayes-Roth, B., Hayes-Roth, F.: A cognitive model of planning. Cognitive Science **3**, 275–310 (1979)
18. Honold, F., Schüssel, F., Weber, M.: Adaptive probabilistic fission for multimodal systems. In: Proc. of the 24th Australian Computer-Human Interaction Conf. (OzCHI), pp. 222–231. ACM (2012)
19. Horridge, M.: Justification Based Explanations in Ontologies. Ph.D. thesis, University of Manchester, Manchester, UK (2011)
20. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.H.: The Manchester OWL syntax. In: Proc. of the OWLED'06 Workshop on OWL: Experiences and Directions, vol. 216 (2006). CEUR Workshop Proceedings
21. Kuhn, T.: The understandability of OWL statements in controlled English. Semantic Web **4**(1), 101–115 (2013)
22. Lin, N., Kuter, U., Sirin, E.: Web service composition with user preferences. In: The Semantic Web: Research and Applications, *LNCS*, vol. 5021, pp. 629–643. Springer (2008)
23. Madsen, M., Gregor, S.: Measuring human-computer trust. In: Proc. of the 11th Australasian Conf. on Information Systems (ACIS), pp. 6–8 (2000)

24. McAllester, D., Rosenblitt, D.: Systematic nonlinear planning. In: Proc. of the 9th Nat. Conf. on Artificial Intelligence (AAAI), pp. 634–639. AAAI Press (1991)
25. Myers, K.L., Jarvis, P., Tyson, M., Wolverton, M.: A mixed-initiative framework for robust plan sketching. In: 13th Int. Conf. on Automated Planning and Scheduling (ICAPS), pp. 256–266. AAAI Press (2003)
26. Nau, D.S., Au, T.C., Ilghami, O., Kuter, U., Muñoz-Avila, H., Murdock, J.W., Wu, D., Yaman, F.: Applications of SHOP and SHOP2. IEEE Intelligent Systems **20**(2), 34–41 (2005)
27. NCICB (NCI Center for Bioinformatics): (2015). http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl (accessed February 9, 2015)
28. Nguyen, T.A.T., Power, R., Piwek, P., Williams, S.: Predicting the understandability of OWL inferences. In: The Semantic Web: Semantics and Big Data, *LNCS*, vol. 7882, pp. 109–123. Springer (2013)
29. Nothdurft, F., Behnke, G., Bercher, P., Biundo, S., Minker, W.: 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), pp. 344–353. ACL (2015)
30. Nothdurft, F., Richter, F., Minker, W.: Probabilistic human-computer trust handling. In: Proc. of the Annual Meeting of the Special Interest Group on Discourse and Dialogue, pp. 51–59. ACL (2014). URL http://www.aclweb.org/anthology/W14-4307
31. Penberthy, J.S., Weld, D.S.: UCPOP: A sound, complete, partial order planner for ADL. In: Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR), pp. 103–114. Morgan Kaufmann (1992)
32. Pichler, M., Seufert, T.: Two strategies to measure cognitive load. In: EARLI Conf. 2011. Education for a Global Networked Society, pp. 928–929. European Association for Research on Learning and Instruction (2011)
33. Pulido, J.C., González, J.C., González-Ferrer, A., García, J., Fernández, F., Bandera, A., Bustos, P., Suárez, C.: Goal-directed generation of exercise sets for upper-limb rehabilitation. In: Proc. of the 5th Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), pp. 38–45 (2014)
34. Schiller, M., Glimm, B.: Towards explicative inference for OWL. In: Proc. of the 26th Int. Description Logic Workshop, vol. 1014, pp. 930–941. CEUR (2013)
35. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence **48**, 1–26 (1991)
36. Sirin, E.: Combining description logic reasoning with AI planning for composition of web services. Ph.D. thesis, University of Maryland at College Park (2006)
37. Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D.: HTN planning for web service composition using SHOP2. Web Semantics **1**(4), 377–396 (2004)
38. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Proc. of the 3rd Int. Joint Conf. on Automated Reasoning (IJCAR), pp. 292–297. Springer (2006)
39. Vardi, M.Y.: Why is modal logic so robustly decidable? Descriptive Complexity and Finite Models **31**, 149–184 (1997)