in: AI Communications, 4, 1999

Plan Management in the Medical Domain

Silvia Miksch
Vienna University of Technology, Institute of Software Technology
Favoritenstraße 9-11 / 188
A-1040 Vienna, Austria
Email: silvia@ifs.tuwien.ac.at
Url: http://www.ifs.tuwien.ac.at/~silvia
Tel: +43-1-58801-18824
Fax: +43-1-58801-18899

The need to improve the quality of health care has led to a strong demand for clinical protocols and computer systems supporting both their creation and execution. Current approaches in the planning community concentrate on algorithmic improvements, but mostly fail in medical applications. Planning approaches are based on assumptions like deterministic behavior, which do not hold in medical domains. Additionally, they do not cover the problem area of acquisition and verifying complex domain knowledge. In the field of medicine there is a strong movement towards clinical protocols – resembling plans in AI – but computer support during execution of these plans (e.g., controlling, selection of alternatives) is still basic. We need to build complex plans, but also to reason about them in different ways in order to modify plans, to consider the effects of different plans over time, and to monitor execution. We call this range of reasoning tasks *plan management*. We describe the requirements for these intertwined tasks of plan management so as to respond to the practical demands, to compare approaches in planning and medical informatics to these requirements, and finally, to discuss how our Asgaard project can meet them.

1. Motivation and Introduction

Currently, several plan-representation languages and various planning techniques are available for distinct kinds of problem descriptions, domains, and purposes. However, for someone who wants to engineer a *practical* planning system for a particular *real-world* domain, it is not that straightforward to simply select a plan representation and an algorithm and directly apply it. Rather a set of open issues remains unsolved. We are interested in bridging the gap between theory and practice providing suitable support for *medical treatment planning.*

Planning Techniques. On the one hand, there are the planning problems and their solutions: Planning is a well-known technique in the Artificial Intelligence (AI) literature which was introduced in the 1960s. In the beginning, it was mainly seen as problem of search and theorem proving. In the 1970s, Fikes and Nilsson [13] introduced the STRIPS (Stanford Research Institute Problem Solver) language, which applies the *situation calculu*s [38] to describe and to solve the planning problem. A lot of approaches descended from STRIPS. The representation used logical sentences to define the "initial state", the "goal state", and "operators" [60]. The main characteristics of STRIPS are: it is a domain-indepentent approach, it has a clear structure (pre-conditions, operators, and add- and delete-lists as post-conditions), its capability is easy to capture, and it uses unbound searches in the state space to find a feasible solution. The main drawbacks of the STRIP's approach are that the problem of search space is NP-complete, it utilizes no domain knowledge to reduce the complexity (i.e., no hierarchical decomposition is used either for structuring the problem specification and the action space or for reducing the search space), and it has no notion of time and uncertainty. These main drawbacks led to new approaches. However, it remains quite unclear, if these new approaches can really solve the existing problems in medical environments such as time handling, incomplete information about the world's state and about the effects of durative actions.

Medical Real-World Domain. On the other hand, there is the medical "real-world" domain. In opposition to mathematical and natural science, medicine is a scientific field, which embodies more unsolved and unforeseeable problems, i.e., the domain knowledge does not cover all aspects with full certainty. Physicians try hard to improve the quality of health care through increased awareness of proper disease management techniques while simultaneously reducing the costs. In opposition to other applications of planning such as logistic, treatment planning from scratch typically is not necessary, as general clinical procedures exist, which should guide the medical staff. These procedures are called *clinical guidelines* and *protocols,* which describe how to treat a patient with a particular disease. However,

the large domain knowledge available is often too uncertain and vague and therefore not in a shape computer scientist would need it to build up a (semi-) formal theory [32].

The Demand. Medical plan management is needed. This plan management has to be an intertwined process consisting of various context-sensitive and task-specific sub-processes (e.g., plan generation, plan verification, plan visualization, plan execution, plan modification, and plan critiquing). A required prerequisite to proceed with such a plan management is a plan representation, which allows for knowledge roles to accomplish these particular processes. Each knowledge role is an abstract label attached to domain knowledge. The knowledge role indicates the role of this knowledge in the inference process [5]. Plan representations that are enhanced by knowledge roles are called rich plan representations. When discussing requirements for rich plan representations and plan management in this paper, we adopt Newell's perspective [51] of a "knowledge level'" analysis rather than addressing this topic at the "symbol level". Practical plan management requires a "knowledge rich" model [69] that facilitates efficient reasoning given the demands of the surrounding environment. According to our medical interest, approaches dealing with time handling, context, and incomplete information about the world's states and the effects of actions (dynamically changing environments) are most important.

In Section 2 we will describe the main challenges in the medical domain and extract the particular requirements for an appropriate plan management within the medical field. In Section 3 we will give a short historical overview of various planning approaches, illustrating which directions are promising for medical applications and which approaches we are neglecting because of their basic assumptions and feature characteristics. We have selected the three most encouraging approaches (O-Plan, Cypress, and Prodigy), which are described and compared to our list of requirements in Section 4. The same procedure is applied to the three medical protocols-based care systems Arden/MLM, EON, and GLIF in Section 5. In Section 6 we show how our Asgaard/Asbru system can meet the proposed requirements. Asgaard concentrates on a second dimension of planning, namely plan management with its large number of sub-tasks.

2. Challenges in Medical Domains

In the following section we describe the properties of the medical domain, which results in a detailed list of requirements for plan management. The proposed plan management is an interactive process consisting of several intertwined tasks.

2.1. Practical Planning Problems in Medicine

Dealing with monitoring and therapy planning in medical real-world environments raises a host of problems. We are not interested in solving general medical problem. Our main focus lies in the clinical treatment management and the medical sub-area that is dealing with mostly high-frequency data in combination with quantitative and qualitative low-frequency data, namely the domain of intensive care units (ICUs). This sub-domain is qualitatively different from pure low-frequeny domains due to highly time-stamped data, critical time-constraint situations for reaction, and rich uncertainty about the actual situation of the patient while faced with various treatment options. Additionally, we are not aiming at medical diagnosis.

Currently, medical staff are facing two major problems: (1) the information overload resulting from modern equipment, and (2) the need to improve quality of health care through increased awareness of proper disease management techniques. *Clinical guidelines* or *protocols* should help to solve these problems. A guideline can be defined as "*a method, that identifies actions, that are to be performed and that specify conditions that govern when it is appropriate to perform them*" [54]. A clinical protocol is a more detailed version of a clinical guideline and refers to a certain class of therapeutic interventions. Protocols are used for utilization review, for improving quality assurance, for reducing variation in clinical practice, for guiding data collection, for better interpretation and management of the patient's status, for activating alerts and reminders, for improving decision support [54].

Besides free text, several methods have been presented to ease authoring of such clinical protocols, including flowcharts, decision tables, and medical logic modules [54]. Using those methods a high number of clinical protocols have been acquired. Therefore, treatment planning from scratch typically is not necessary, because the medical background is somehow available. The drawback is that the acquired clinical protocols often are too uncertain and vague to build up (semi-) formal theory. The variability of clinical protocols (e.g., a medical goal can be achieved by different therapeutic actions), the mutual dependencies of parameters, the possible orders of plan execution, all the exception conditions, and the temporal dimensions are hard to represent in the proposed methods. Each of these features ought to be modelled *explicitly* [32]. Therefore, particular guidance and style-sheets to acquire the necessary medical knowledge and its roles are essential.

The characteristics of therapy management are qualitatively different from many traditional target domains in planning and scheduling. The requirements in medicine are often a superset of the requirements in typical toy-problem domains used in planning research. Therapy management is driven by the unpredictable nature of the medical domain. For example, states, events, actions, plans, goals, and effects are durative (not instantaneous) and with uncertainty in their appearances. This makes monitoring of the states and events during execution of durative actions necessary. This monitoring must be more than simple deleting and adding properties. Temporarily suspension of a plan is an important concept in medical planning, but it is missing in classical planning. For example, a plan can be started, then, caused by some external event, the same plan needs to be suspended, after a period of time this same plan can be continued, and, finally, it may be completed successfully or aborted. All these state transitions are driven by external and unpredictable events.

The most relevant characteristics of the medical domain are:

- large domain knowledge is available, but this knowledge is often uncertain and vague;
- incomplete and non-deterministic information about the world state and effects of actions;
- unobservable underlying processes determine the observable world states; not only agents affect the world states (e.g., events occur without actions);
- world states, events, actions, plans, goals, and effects are durative (not instantaneous) and with uncertainty in their occurrences; effects vanish or become uncertain after some time; actions might have delayed effects and temporally postponed goals;
- time annotations have to support multiple time lines based on different granularities by providing reference annotations (e.g., different zero-time points and time units);
- plans can be suspended.

Characteristics that are important, but hold in other domains too, are:

- a huge volume of data;
- pre- and post-conditions are needed to control the execution of durative actions or plans;
- a goal may not be achievable in time;
- sequential, parallel, and cyclical execution of plans is necessary.

2.2. Which Plan Management is Needed in Medicine?

Plan management involves more than specifying a problem, generating a possible solution path to reach a goal state from an initial state, and executing this solution path. Plan management includes everything from designing a particular plan or a hierarchy of plans to the real-word execution and evaluation of such plans (compare [57]). Plan management consists of various tasks. These tasks are not performed in a strict sequence (one by one). The various tasks are interdependent and are intertwined periodically, concurrently, and interactively. These tasks can be clustered into groups according to their purposes. Many of these groups overlap in their functionalities. Additionally, we can distinguish tasks, which need to be performed mainly at design time and those which are done mainly during execution time of plans. However, this strict separation of tasks according to the time they will be performed, is not always possible in reality. A full integration of plans' design, generation, execution, adaptation, and analysis is needed, which results in intertwining design-time tasks and execution-time tasks (compare Figure 1; adapted from [23]).

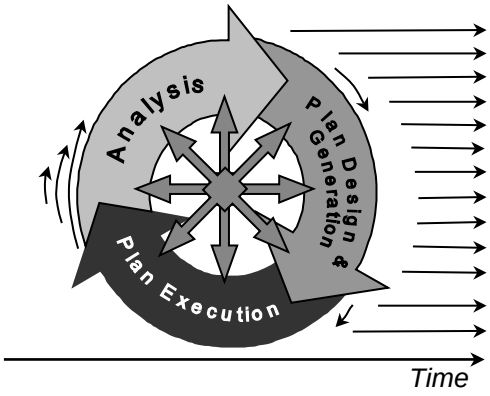

Figure 1 Plan Management seen as fully integrated and intertwined tasks.

Tasks mostly done at design time:

- ***Plan Generation***: starts from an initial state description and creates a path of activities to reach the desired goal (progressive way), or starting from the goal (regressive way);
- ***Advanced Plan Editing*****:** provides guided support to author plans and helps to browse plans or a plan hierarchy;
- ***Domain-Specific Annotations:*** provides structured support to write domain assumptions or domain activities;
- ***Plan Verification:*** examines the method semantics of plans and plan hierarchies;
- ***Plan Validation***: examines the domain semantics of plans and plan hierarchies;
- ***Plan-Senario Testing:*** enacts groups of plans by applying definite domain scenarios;
- ***Plan Visualization:*** communicates efficiently sets of plans to domain experts.

Tasks mostly done at execution time:

- ***Plan Selection:*** assigns task according to state/situation;
- ***Plan Adaptation:*** adjusts plans or plan hierarchies to distinct situations at the starting time;
- ***Plan Execution:*** performs selected activities;
- ***Plan Monitoring****:* compares assumptions with reality;
- ***Plan Modification / Alternatives****:* handles changes in the environment;
- ***Plan Evaluation / Critiquing:*** analyzes executed plans or plan hierarchies according to their goals and intentions;
- ***Plan Visualization:*** supervises and communicates plans or plan hierarchies;
- ***Plan Rationale / History:*** explains executed plans or plan hierarchies.

According to the described tasks, four main problem areas can be localized: (1) representational requirements to structure the problem specification, (2) verification and validation of the designed plans, (3) execution and reaction on changes in the environment, (4) user interface and communicating plans to domain experts. In the following we will list the representational requirements in detail and discuss the other three issues briefly.

Representational Requirements in Medicine. According to the medical demands described in Section 2.1 and the properties of the desired plan management, we summarize the most urgent characteristics needed in an appropriate plan representation language:

(1) ***hierarchical decomposition*** of plans:
(which are uniformly represented in a plan-specification library; *skeletal plans* containing variables/parameters):
- expansion by *effects or goals;*
- expansion by *plans' name;*

(2) ***knowledge rich control structure****:*
- *compulsory* and *optional* plans;
- *temporal order*, which is sequential, a concurrent, and a cyclical execution of plans;
- *conditions* that hold at the beginning and ending of plans (pre- and post-conditions);
- *conditions* that hold or can be achieved at particular plan step during execution (like setup- and suspend-conditions);
- *intentions*, considered as high-level goals;
- *preferences and resources*, constrain the selection of a plan;
- *effects*, describe the possible effects of plans;
- *basic data structure (*data-types, primitive types, units, etc.)

(3) ***temporal dimension****:*
- *continuous* (durative) states, actions (plans), and effects;
- *temporal uncertainty* in the occurrence of states, actions (plans), and effects;
- *various time granularities;*
- *arbitrary relative and absolute time references*
(which includes various reference annotations: absolute, relative to the actual execution of the plan, or relative to a state of another plan);

(4) ***context.***

Two terms mentioned above are not obvious and need some clarification: (i) *skeletal plans* and (ii) *context*:

(i) ***skeletal plans***: A common strategy for the representation and the reuse of domain-specific procedural knowledge is the representation of that knowledge as a library of skeletal plans. Skeletal plans are plan schemata at various levels of detail which capture the essence of the procedure, but leave room for execution-time flexibility in the achievement of particular goals [21]. Thus, they are usually reusable in various contexts. The idea was originally proposed by Friedland [20] as a means to reduce the complexity of planning, called skeletal-plan refinement. Instead of planning in an unconstrained search space, the skeletal-plan refinement method relies on available abstract (or skeletal) plans, which are refined in the context of a particular problem.

(ii) ***context***: A context is the general condition (i.e., circumstance) in which an event, action, and so on takes place (second meaning of "context" according to the *Oxford English Dictionary* cited in [1]; the first meaning is closely related to linguistic meaning). Therefore, the context supports the better understanding of activities by annotating factors related to them rather than considered the activities on their own. The issue of context has been emerged in various areas of AI, including knowledge representation, natural language processing, and intelligent information retrieval. The main motivation for studying formal context is to resolve the problem of generality in AI [1]. Usually any contextual knowledge is generally spread throughout the knowledge base. The demand for explicit representation of context is well known [1]. However, the context is seldom explicitly used in the plan-specification languages (compare Section 4 and 5). Context-sensititvity is fundamental to effective plan management. It influences the selection of plans, the qualifications of numerical values, the verification and validation, the visualization, and the various aspects of plan execution [41; 42].

In the following we will address the issues of the other three main problem areas briefly.

Verification and Validation of the Designed Plans. The safety and transparency aspects are very critical issues in the medical domain [24]. A clinical protocol is only accepted from the medical staff, if you could prove that it is verified and valid as well as if all assumptions and consequences are lucid. A detailed discussion is given in [10].

Execution and Reaction on Changes in the Environment. In the medical domain, we need to build complex plans, but also to reason about them in different ways in order to modify plans as reaction on changes in the environment, to consider the effects of different plans over time, and to monitor execution. A detailed discussion is given in [61].

User Interface and Communicating Plans to Domain Experts. Plans are specified in powerful languages, like LISP. However, LISP-like notations are very difficult to read for domain experts. Powerful methods are useless, if they cannot be used by the people they are intended for. An appropriate user interface to the plan specification is needed which is easy to handle and which communicates the underlying concepts and issues to the domain users. A detailed discussion is given in [31].

3. Historical Overview: Planning and Scheduling

Planning and scheduling is an issue in the AI community since the 1960s. In this section we give a short overview of the history and diversification of efforts in this field and how they relate to the medical domain.

3.1. Clarifying the Terms Planning and Scheduling

Planning and scheduling are well-known techniques in the Artificial Intelligence (AI) literature. *Planning* is designing the behavior of some entity that acts as an individual, a group, or an organization. The output is some kind of blueprint of behavior [40]. *Scheduling* is selecting among alternative plans and assigns resources and times.

Planning. In general, planning is the process of selecting and sequencing activities in such a way, that the activities achieve one or more goals and satisfy a set of domain constraints. Planning research has concentrated mainly on finding a feasible set of actions that accomplishes one or more goals [18]. When planning started in the 1960s, it was mainly seen as an application of two standard AI techniques, namely search and theorem proving. There is a wide variety of planning problems, differentiated by the types of their inputs and outputs. Typically, a planning problem gets more and more difficult as more and more flexible inputs are allowed and fewer constraints on the output are required. The classical approach to planning problems is to start from specifications of the states, the actions, and effects of actions, and then try to infer a sequence of actions that accomplishes a particular state of affairs.

Scheduling. Even though Simon pointed out in 1972 the problem of allocating resources over time [66], it was not until the early 1980s that scheduling came under serious scrutiny. The blocks-world problem used by planning researchers never enforced the issues that arise in scheduling - it took a return to the "real-world" for these issues to reappear [19]. *Scheduling* selects among alternative plans and assigns resources and times for each activity so that the assignments obey the temporal restrictions of activities and the capacity limitations of a set of shared resources. The assignments also effect the quality of a schedule with respect to criteria such as cost, response time, or throughput. Thus, scheduling is an optimization process in which limited resources are allocated over time among both parallel and sequential activities.

The difference between planning and scheduling appears somehow unclear. A rule of thumb can be applied to clarify this issue: If a set of actions that must be performed is known beforehand, then it is a scheduling problem to assign resources and times to these actions. If the set of actions must be generated based on the current situation and the current goal, then the problem solution also involves planning. Other features of a problem can change it from a scheduling problem to a planning problem. For example, if some subset of the constraints in a scheduling problem will change depending on how another subset is satisfied, e.g., if the constraints on the afternoon's schedule will change depending on how the constraints on the morning's schedule were satisfied, then such a scheduling problem becomes a planning problem [83]. Scheduling can be seen as a sub-problem of planning.

3.2. A Conceptual Separation for Planning

There are different ways to cluster the various generative planners. We will illustrate three important views:

(1) according to the *domain knowledge used*;
(2) according to the *reuse* of existing plans; and
(3) according to the *strategies* of planning.

Domain Knowledge Used. First, the planning problems have been attacked in two major ways [75]: approaches which try to understand and solve the general problem without use of domain-specific knowledge (called *domain-independent* approaches) and approaches that use domain heuristics directly (called *domain-dependent* approaches). Work in domain-independent approaches has formed the bulk of research. The domain-independent approach that most planners use today describes states and operators in a restricted language known as the STRIPS language [13], or in extensions thereof. STRIPS (Stanford Research Institute Problem Solver) was used as the planner for the Shakey robot [12]. The STRIPS language is based on situation calculus [38]. Therefore, many approaches that descended from STRIPS are unable to handle durative events and actions as well as uncertainty and variability in the utility of available actions. STRIPS searches in the state space. The usual assumptions are that only the agent affects the state of the world, that all actions occur instantaneously, that the effects of the actions are instantaneous, and that all actions follow another without break in between. Finally, classical planning assume complete and deterministic information about the world's state and the effects of actions. These assumptions are inappropriate in the medical domains.

Reuse of Existing Plans. Second, we distinguish *first-principle* and *second-principle* planning. *First-principle* planning is planning from scratch – using a particular plan/action specification language – to generate a sequence of actions/steps: One gathers evidence about the world, makes a characterization of the problem, and attempts to generate a plan – a sequence of operators that will, if executed, lead to the desired goal state. *Second-principle* planning involves reusing solutions of previous problems: first, to identify an appropriate reusable candidate from a plan library, and second, to modify this plan candidate so that it solves the new problem instance. Second-principle planning is distinctly separated from case-based planning. Second-principle planning equals skeletal planning (compare Section 2.2). It formulates general concepts by abstracting common properties of previous problems' and solutions' instances.

Strategies of Planning. Third, we divide the generative planners according to their strategies: hierarchical task networks (HTN), state-space planner, and plan-state planner. A HTN decomposes abstract operators into a group of steps that forms a plan that implements the abstract operator. A state-based planner searches through the space of possible situations progressively (forward search from initial state to goal state) or regressively (backward search from goal state to the initial state). A plan-state planner searches through the space of plans instead of the space of states, starting from a simple, incomplete plan (called partial plan). SIPE [82] and O-Plan [74] are considered as HTN planners, PRODIGY4.0 [78] as state-space planner, and UCPOP [56] as plan-space planner.

The drawbacks of various frameworks led to the exploration of a wide variety of new approaches. In the following we shortly describe the approaches inside and outside our scope of interest.

3.3. Approaches Inside our Scope of Interest

For our medical domain, approaches dealing with time handling, context, and incomplete information about the world's states and the effects of actions (dynamically changing environments) are most important. The approaches should be able to build complex plan structures, to modify the plan layout partly during execution, to monitor durative actions and states, and to react to external events during execution time. Additionally, the approaches should use domain knowledge to decompose hierarchically the plan space and the state space to reduce the complexity of the task. The list of requirements for the desired plan representation and plan management has been given in Section 2.2. According to these requirements we have chosen three approaches to compare:

(1) the O-Plan project [73], which provides a generic domain-independent architecture suitable for command, planning, and execution applications — a HTN approach including time and resource handling;
(2) the Cypress project [85], which is an integrated planning environment, a domain-independent framework for defining taskable, reactive agents — a HTN approach including a powerful planning and execution technology;
(3) the Prodigy project [78], which is a general-purpose problem-solving architecture for planning and machine learning — a state-based planner including monitors.

Other candidates, which are more limited in applicability or have another emphasis, were excluded from the detailed comparison because of space limitations, such as:

- DEVISER [80] provides a method for specifying a time window on goals and activities: external events and their time of occurrence as well as delayed events caused some time after a planned action can be specified. But it has limited re-planning capacities.
- FORBIN [7] is a one of the first temporal hierarchical planning, but with too limited features of plan management.
- PROPEL [37] is a non-deterministic formalism for expressing behavior that can be used for both projection and execution. His language is mainly designed to meet real-time deadlines.
- The Time Map Manager [8] is a model for flexible reasoning about time, but has too limited features of plan management.
- TPLAN [3] uses a STRIPS-like action formalism, expect that the preconditions and effects are temporally qualified.
- At the Fourth International Conference on Artificial Intelligence Planning Systems (AIPS 98, [65]) encouraging progress in the field of plan-space searching could be observed (e.g., STAN [17], IPP [30], BLACKBOX [29]). Unfortunately, most systems use STRIPS-like domain descriptions and the Planning Domain Description Language (PDDL) [39], which does not provide for temporal constraints and uncertainty handling.

3.4. Approaches Beyond our Scope of Interest

There are other approaches in the planning community that explore problems also seen in the medical domains but come to solution not applicable in medicine.

Uncertainty is handled by *probabilistic planning*. Among others [33] present an algorithm for probabilistic planning to deal with incomplete information. A probability distribution over possible world states is used to model imperfect information about the initial world state and the actions are modeled using conditional probability over changes of the world. However, the probability distributions in the medical domain are difficult to acquire. An established study in 1972 showed that "estimates" provided by the physicians are inaccurate since they have little idea of what the "true" probabilities are [34]. Probability distributions deduced from computer-based patient records are site- and culture-specific (e.g., different populations, vocabulary used is somewhat different at each clinical site). Additionally, traditional decision theory often requires more information than available in realistic domains [49]. In this situation, a qualitative model of uncertainty and decision-making is needed, including the ability to represent the growth and the decline of uncertainty as time passes. In the ICU case, such methods might suggest certain actions based on their immediate beneficial effects (e.g., increasing respiratory rate), even if the long term effects may be less clear (e.g., leaving the respiratory rate too high for a long period of time could eventually cause a respiratory problem).

Non-deterministic changing environments are the main focus of *reactive planning*. It is mostly applied in the robotic domain: RAP (Reactive Action Package, [14; 15]) is a typical example. The major emphasis of this system is on reacting on external events. There is no planning in the sense of searching state space. Instead, all tasks to be executed are stated in a task net. Compared to medical applications, the domain knowledge incorporated in a robot control system is rather poor. Therefore, complex facilities for structuring a knowledge base like those needed in the medical domain are not provided.

Standardization efforts of considerable number have been seen in the domains of manufacturing and military mission planning. The most recent and most promising approach is the Process Specification Language (PSL) [27]. Its intended field of application spans from project and workflow management to manufacturing process planning and computer aided software engineering. It is supposed to become a standard for the data exchange of applications in these fields. While resources, time and temporal uncertainty, hierarchical decomposition and concurring actions are covered, context and reacting on changes in the environment are not.

PSL has many forerunners in the military planning community, the more important of which are the Core Plan Representation (CPR, [55]), the Knowledge Representation Specification Language (KRSL, [36]), and the Process Interchange Format (PIF, [35]). The semantics of PSL are defined in the Knowledge Interchange Format (KIF), which is an early forerunner of the later. Similar models were defined with in the planning initiative "Shared Planning and Activity Representation" (SPAR, [72], [58]).

4. Selected Approaches in the Planning Community

In this section we describe three projects, which illustrate what has been achieved in the planning community and how these approaches match the requirements for effective plan management in the medical domain. For each project, we first describe the general design, the plan representation, the application area, and then compare the features with the requirements defined in Section 2.2.

4.1. The O-Plan Project

General Design. The O-Plan (Open PLANning Architecture) project [70; 73; 74] provides a generic domain-independent architecture exploring issues of coordinated planning and control. O-Plan combines a number of techniques:

- a hierarchical planning system, which can produce plans as partial orders on actions;
- an agenda-based control architecture in which each control cycle can post pending tasks during plan generation. These pending tasks are then picked up from the agenda and processed by appropriate handlers (Knowledge Sources);
- the notion of a "plan state'" which is the data structure containing the constraints on an emerging plan, the "agenda" of further requirements, and the information used in building the plan;
- constraint posting and least commitment on object variables;
- temporal and resource constraint handling using incremental algorithms, which are sensitively applied only when constraints alter.

O-Plan is derived from the earlier Nonlin planner [68] from which it takes and extends the ideas of goal structure, question answering (Modal Truth Criterion), and typed conditions.

The formalism to specify plans in O-Plan is called Task Formalism (TF), which extends Nonlin's style of task descriptions. TF can be seen as an implementation that rests upon the more general <I-N-OVA> constraint model [71]. <I-N-OVA> stands for Issues, Node Constraints, Ordering Constraints, Variable Constraints, and Other Constraints and is a means to represent plans as a set of constraints. In this model it is assumed that tasks, plans, processes and activities can be defined by their constraints on behavior. The different types of constraints in the <I-N-OVA> model reflect the different types of components in an O-Plan agent (issue controller, issue handlers, and plug-in constraint managers).

Plan Representation. The Task formalism (TF) [73] in O-Plan uses the term "schema" for a plan while possible solutions are called plans. The state of the world is stored in a STRIPS-like list of propositions, which are extended by optional information about their temporal validity. The problem specification is called "task". It specifies the initial state of the world in the "effects"-statement and the goals to be achieved in the "conditions"-statement. Alternatively, plans can be instantiated directly by supplying an "action"-statement.

TF uniformly represents time constraints and resource usage. The time constraints are specified by a numerical (min, max) pair, which bounds the actual values for activity duration (the activity's starting and finishing times) and which determines the delays between the activities. The resources can be consumable or reusable, the later having various sharing modes. Therefore, temporal constraints and reasoning as well as elaborate modelling of resources are provided.

In each schema (i.e., plan) sub-plans can be instantiated via an "action"- or an "achieve"-statement. In the first case, the pattern given as argument of the "action"-statement is matched with the pattern of the "expands"-statement of each schema, which results in a mechanism resembling subroutine calls. In the second case, the condition given as

argument of the "achieve"-statement is matched with the effects given in the "only_use_for_effects"-statement of each plan.

Temporal ordering of the actions, which is part of a schema can be specified via keywords like "parallel" or "serial" or by defining a graph of temporal succession. TF allows the start, end and duration of a plan to be defined as an interval. All timing is relative to the starting time of a specified plan.

Application Area. O-Plan has been applied in mission sequencing (e.g., air campaign planning workflow, crisis response) planning and control of supply and distribution logistics (e.g., construction and house building) and project management.

Comparison. O-Plan is a very powerful system. O-Plan seen as a classical planning system allows for search in the state space via the "achieve"-statement. Suitable plans to achieve the condition given in the statement are searched for in a depth-first fashion. Matching is done by looking at the world state before the execution of the plan as defined in "only_use_for_query" and the world state after the execution of the sub-plan as defined in "only_use_for_effects".

An "action"-statement in the plan body provides the hierarchical decomposition. The key given as an argument must match that in the "expands"-clause of a plan to be selected as a suitable sub-plan.

O-Plan provides a rich control structure. The authors of a plan can define the successors of sub-plans via arbitrary graphs. All kinds of the temporal order of executing a plan, of temporal uncertainties, of different time granularities, and of effects are supplied. Any condition can be defined to be crucial for a plan (e.g., supervised, unsupervised) and must be guaranteed to be true for the whole period of execution of a plan. Otherwise the plan is not considered to be executable.

O-Plan has no distinct facility to specify the intention of a plan. It is only implicitly given by the conditions to be achieved or the sub-plans to be executed. Therefore, no plan critiquing is provided. The functionality of the intentions in the plan selection process can be implemented via conditions. There is no provision for automatic re-planning in case some sub-plan fails due to unpredictable influences or for suspending a plan. O-Plan has a powerful resource handling.

In O-Plan there is no concept of context. As effects of the missing context, no additional conditions can be specified to select a plan. O-Plan expands its plans by plans' name, which results in a quite complex process of plan selection in case of a large plan library.

By the power of TF, O-Plan supports the basic plan management functionalities, like generation, execution, and basic controlling of plans. Plan visualization, plan alternatives, and monitoring is realized to a certain degree or implicitly possible. However, more advanced features, like plan verification, plan adaptation, and plan critiquing are missing (compare Table 2).

4.2. The Cypress Project

General Design. The Cypress project is an integrated planning environment, a domain-independent framework for defining taskable, reactive agents [85]. The system can react on partial plan failure by modifying plans in part while continuing the execution of those parts of the plan, which are not affected by the failure. Both temporal and probabilistic uncertainties are covered. Cypress consists of various technologies:

- SIPE-2: A generative planner, which allows the generation of action sequences that include parallel actions with multiple agents, conditional actions and resource assignments as well as the modification of its plans during execution. It supports partial-order planning at multiple levels of abstractions (a hierarchical task network (HTN) planner) [82];
- PRS-CL: A reactive execution system, which is an extension of the Procedural Reasoning System (PRS) developed by [22], for constructing persistent, reactive controllers that can perform complex tasks in dynamic environments [84];
- Gister-CL: An evidential reasoner, which can be used during plan generation and plan execution to analyze uncertain information about the world and possible actions. It provides both probabilistic models (Bayesian and DempsterShafer) and possibilistic models (fuzzy logic);
- Act-Editor: A procedural knowledge editor, which is a tool for graphically browsing and editing procedural knowledge expressed in the Act formalism;
- GKB-Editor: A declarative knowledge editor, which is a tool for graphically browsing and editing knowledge bases across multiple frame representation systems (e.g., Loom, Ontolingua) in a uniform manner;
- Grasper-CL: A common graphical user interface, which is a system for viewing and manipulating graph-structured information [28].

The core of Cypress is its powerful planning and execution technology, namely SIPE-2 and PRS-CL. As an interlingua between SIPE-2 and PRS-CL, the ACT representation for actions and plans is used [84].

Plan Representation. The ACT language [50] is a domain-independent language for expressing the kind of procedural information required by both plan-generation and plan-execution systems. At the heart of the formalism is the “act” structure, which corresponds to both an operator or plan in the terminology of generative planners, and a plan fragment or operating procedure in the terminology of plan execution systems [84]. Each Act describes a set of actions that can be taken to fulfill some designated purpose under certain conditions. The purpose could be either to satisfy a goal or to respond to some event in the world. An Act can represent, among other things, a procedure, a planning operator, or a plan at a particular level of detail. A “plot” represents a group of closely coupled actions. Preconditions are gating conditions and are used to constrain the application of an act. The “cue” indicates the purpose for which the Act can be used.

By default, the order of plans is only determined by their interdependencies (partial-ordered plans). Plans can explicitly be defined to execute in parallel. Cyclical execution of plans can be achieved using the “rebind”-statement to bind new values to variable and then re-achieving a target again. Since the connections of plot nodes are arbitrary, it may also possible to arrange them in a cyclical graph.

Application Area. The different components of Cypress have been tested in various domains (mobile robots, house and office construction, and beer production) and with toy problems (blocks world, tower of hanoi). The Cypress project as a whole has been applied to joint military operations planning.

Comparison. Cypress is the most encouraging approach, which supplies a lot of functionalities. The hierarchical decomposition is specified via the “sub-plans”-clause. Each operator specifies its purpose or goal in the “purpose”-clause and the sub-goals to be achieved in the “goals”-clauses. The “goal”-clauses are used as a search condition in plan space and they are matched against the "purpose"-clauses of other plans.

The ACT language allows the author of a plan to specify a complex control structure. Conditional plans can be specified via the “type conditional”-clause. Partial ordered plans permit sequential and concurrent execution of plans. Cyclical plans are realized applying the trick with the “re-bind”-statement. Intentions can be realized by “meta-predicates” which provide for high-level plan selection. The meta-predicate “require-until” specifies a protection interval, namely a condition to be maintained until another indicated condition for terminating this requirement occurs. During plan execution the PRS-CL executor handles failures of this meta-predicate. Setup-conditions are evaluated only once, but achievable conditions can be realized by a trick with the “re-bind”-statement. However, there is no explicit notion, when to suspend a plan. Additionally, there is no effort to make the precondition true: a false precondition simply means that the operator is inappropriate. The effects are listed in the “achieve”-clause in the cue. There is no explicit preferences slot of a plan but meta-predicates provide various ways to implement the functionality of preferences. Resources necessary can be specified in the "use-resource"-clause. The "achieve-by"-clause can be used to focus on a limited number of means for achieving a goal.

Cypress provides access to powerful temporal reasoning capabilities. The temporal relation of any pair of plot nodes (i.e., instantiated plans) can be specified by one of the Allen-relations [2]. For each instant in plan execution (i.e., node), a time window can be specified, which defines earliest and latest start times, earliest and latest finishing times, and minimum and maximum durations. Time is given as absolute number relative to the start of the plot or as (positive) infinity or epsilon. However, no different time granularities and arbitrary relative time references are allowed. The unit of time is defined by the implementation respectively implicitly given. The uncertainty of the states and effects are deduced implicitly. All states and effects last until they are changed by other actions. There is no notion of context.

From the planning management point of view, Cypress is very favorable too. It includes generative planning, plan execution, a procedural and declarative knowledge editor, and a graphical user interface (compare Table 2). However, plan verification and validation and critiquing are not supported, which are very important in the medical domain because guaranteed predictability of a system is crucial (e.g., safety issues).

4.3. The Prodigy Project

General Design. Prodigy is a general-purpose problem-solving architecture that serves as basis not only for planning, but also for machine learning [78]. It was originally begun by Jaime Carbonell [6]. The planner within Prodigy (Prodigy 4.0) is a nonlinear problem solver searching in the state space. It selects sub-plans via pre-conditions and effects (post-conditions). The preconditions can be sub-goaled upon and therefore they roughly correspond to SIPE’s goals (compare Section 4.2). The control rules (or inference rules) globally enforce the selection, rejection or preference of a candidate plan over its alternatives under specified conditions. There is neither a representation of time nor uncertainty in Prodigy, because it is close to the STRIPS language [13]. Actions change the environment in STRIPS-style by adding and deleting propositions in the list describing the world. The formalism to specify plans in Prodigy is called Prodigy's Description Language (PDL).

In 1998, Veloso et al. [79] introduced monitors to handle changes in the environment during the generation of a plan. Monitors are defined independently from plans. Each monitor is related either to a selected plan or to the alternatives of a selected plan. In each monitor sub-goals, usability conditions, or quantified conditions are defined. In reaction to the defined conditions, a monitor can add and delete plans from the to-do-list. A sub-goal monitor might e.g. delete a plan from the to-do-list, if its goals unexpectedly get fulfilled by other activities or external events. A usability-condition monitor can delete a plan, if some precondition of a plan becomes false or add a plan, if the condition becomes true again. A quantified-condition monitor looks at the number of some items and e.g. cancels a transportation task, if the number of items to transport becomes zero or orders another truck, if the number exceeds the capacity of one.

Plan Representation. A problem in Prodigy's PDL 4.0 [6] is given by an initial state and a goal state specification. The description of an initial state (or any state, for that matter) is composed of a list of objects and their corresponding types together with a set of instantiated predicates (i.e., literals that describes the configuration of those objects).

The plan library consists of operators and inference rules. Both share the same syntax (and functionality), but their semantics is different. Operators represent actions that can produce changes in the state of the world. Inference rules represent deductions to be derived from the existing state. Each of them has preconditions and effects. Preconditions contain type specifications for the objects used and arbitrary expressions to specify the conditions, under which the effects occur. Effects contain a list of "add"- and "delete"-statements to change the list of propositions representing the world's state. No time and uncertainty is defined.

Application Area. Prodigy has been applied to different domains, like robotic path planning, process planning, and logistics planning.

Comparison. Prodigy is the less favorable approach because of its state-based roots. In Prodigy, standard plan selection is done via pre-conditions. There is no way to explicitly nominate a sub-plan to be executed as part of another plan. All plan selection is done via conditions. Therefore, no hierarchical decomposition is provided, except if sub-plans could be associated with conditions.

The control structure is quite poor because of the STRIPS-like notation. Since plans are only selected to achieve a given goal, any plan is optional depending on whether its goal is already achieved by other circumstances or not. Monitors support to remove plans from the list of plans to be generated even after initial planning is done if the goal happens to be achieved by other means (as a side effect of other plans or because of non-determinate behavior of the environment). Preferences could be modelled within the control-rules. Control-rules, which can be seen as meta-knowledge of the whole plan, can influence the order, in which alternative plans are tried out. There is no explicit notion of intentions or goals. Therefore, critiquing is not provided. The effect of using goals on plan selection can be achieved via additional condition and via control rules. Particular conditions are modelled by the extension of the monitor's concept. The monitors help to react on changes in the environment in a flexible way. At plan's creation time, all conditions are assumed to hold until explicitly changed. Effects are treated implicitly. After execution of a plan, the conditions, which define its goal, are assumed to be true. Since there is no notion of time, the interval of time during which the goal condition should be true, cannot be specified. I.e., if that condition is falsified by another plan, it does not matter, if same condition is not in the goal of a superior plan.

There is no notion of time or uncertainty in Prodigy. The order of execution of plans depends solely on their mutual interdependency and on the strategy of the search algorithm (depth-first by default). Therefore, no different temporal dimensions, etc. exist.

There is no explicit notion of a context, but conditions can be used to achieve the effect or it can be handled by control rules.

The plan management possibilities are very limited. Prodigy mainly supports generation, execution, and monitoring of plans (compare Table 2). The machine learning capability of Prodigy was not subject of this comparison.

Table 1 Representational Requirements. Summarizing the comparison of the approaches in the planning and medical informatics communities. Legend on the rating: "*yes*" means "is supported", "*no*" mean "is not supported", "partly" means "is somehow supported". The evaluation criteria are described in Section 2.2, explanation for the planning approaches are given in Section 4, and for the medical informatics approaches are given in Section 5.

	Approaches in the Planning Community			**Approaches in the Medical Informatics Community**		
Representational Requirements	**O-Plan**	**Cypress**	**Prodigy**	**Arden MLM**	**EON**	**GLIF**
hierarchical decomposition						
▪ expansion by effects or goals	yes	yes	partly	no	no	no
▪ expansion by plans' name	yes	yes	no	no	yes	yes
knowledge rich control structure:						
▪ compulsory and optional plans	yes	yes	partly	no	partly	partly
▪ temporal order	yes	yes	no	partly	yes	yes
▪ conditions (beginning and ending of plans)	yes	yes	partly	yes	yes	yes
▪ conditions (during plan execution)	no	partly	partly	no	yes	yes
▪ intentions (high-level goals)	no	partly	no	no	partly	partly
▪ preferences and resources	yes	yes	partly	no	partly	partly
▪ effects	yes	yes	partly	no	partly	partly
▪ basic data structure	yes	yes	yes	yes	yes	yes
temporal dimension						
▪ continuous (durative) states, actions (plans), and effects;	yes	yes	no	no	partly	no
▪ temporal uncertainty in the occurrence of states, actions (plans), and effects;	yes	yes	no	no	partly	no
▪ various time granularities;	yes	no	no	no	partly	no
▪ arbitrary relative and absolute time references	yes	no	no	no	partly	no
context						
▪ available	no	no	no	no	no	no

Table 2 Plan Management. Summarizing the comparison of the approaches in the planning and medical informatics communities. Legend on the rating: "*yes*" means "is supported", "*no*" mean "is not supported", "partly" means "is somehow supported". The evaluation criteria are described in Section 2.2, explanation for the planning approaches are given in Section 4, and for the medical informatics approaches are given in Section 5.

	Approaches in the Planning Community			**Approaches in the Medical Informatics Community**		
Plan Management	**O-Plan**	**Cypress**	**Prodigy**	**Arden MLM**	**EON**	**GLIF**
Design Time:						
▪ Plan Generation	yes	yes	yes	no	no	no
▪ Advanced Plan Editing	yes	yes	partly	partly	yes	partly
▪ Domain-Specific Annotations	yes	yes	partly	partly	partly	partly
▪ Plan Verification	no	no	no	no	no	no
▪ Plan Validation	no	no	no	no	no	no
▪ Plan-Senario Testing	yes	yes	yes	no	yes	yes
▪ Plan Visualization	partly	yes	no	no	partly	no
Execution Time:						
▪ Plan Selection	yes	yes	yes	yes	yes	yes
▪ Plan Adaptation	no	no	no	no	no	no
▪ Plan Execution	yes	yes	yes	yes	yes	yes
▪ Plan Monitoring	partly	yes	yes	no	partly	partly
▪ Plan Modification/ Alternatives	partly	yes	no	no	no	no
▪ Plan Evaluation / Critique	no	no	no	no	no	no
▪ Plan Visualization	partly	yes	no	no	partly	no
▪ Plan Rationale / History	partly	partly	no	no	partly	no

5. Selected Approaches in the Medical Informatics Community (Protocol-Based Care)

Researchers in medicine and medical informatics have recognized the importance of protocol-based care to ensure a high quality of care since the 1970s. *Clinical protocols* and *guidelines* should help to overcome these difficulties. The term "clinical guideline" refers to a general principle by which a course of actions is determined and "clinical protocol" refers to a general class of therapeutic interventions. Protocols and guidelines can be seen as plans or sets of actions. Both terms (protocol and guideline) are used in various approaches without really distinguishing their particular meaning. Therefore, we will use clinical guideline and protocol interchangeably in the following.

A number of methods have been applied to represent clinical protocols and guidelines, including free text, flowcharts, decision tables, and medical logic modules [54]. However, the large domain knowledge available is often too uncertain and vague. To represent the implicit medical knowledge with these techniques explicitly is a nasty and intractable problem [32]. Therefore, most protocol-based care systems basically consider protocols as composition of *actions* (which are to be performed) and *conditions* (which define when it is appropriate to perform the actions). Most of the systems support elementarily command-based or graphical flowchart editing of protocols, simple descriptions of domain-specific annotations (mostly unstructured text only), and straightforward plan execution without interruption or later adaptation according to changes in the environment.

According to our requirements we have chosen three most promising approaches to compare:

- The Arden Syntax and the Medical Logic Modules (MLMs) [26];
- the EON approach [48; 76];
- the GLIF (GuideLine Interchange Format) Approach [52].

Other candidates were excluded from the detailed comparison because of space limitations, such as:

- the MBTA (Modeling Better Treatment Advice) system [4], which aims to support the automation of practice protocols by using a client-server architecture and an object-oriented data representation. The plan representation relies on the application of regular programming languages like C++, which lead to a big gap between the perception of the domain experts and the representation required by the system.
- the GEODE-CM (Guided Entry of Data Elements for Clinical Management) system [67], which is mainly a flowcharting tool for protocol and concentrates on presenting the divers protocols of care.
- the DILEMMA project [25], which provides a protocol authoring tool and a protocol task manager as a prototype.
- the PRO*forma* project [16], which is similar to the EON approach. It was intensively tested by [81].
- the SPIN (Skeletal Plan Instatiation) approach [77]. It instantiate and execute clinical protocols, which are modeled as hierarchical, skeletal plans and is expressed in a system-specific programming language leading to problems similar to MBTA.

A more detailed study can be found in [10] for the technical issues and in [54] for the medical issues.

In the following we describe the three projects, which illustrate what has been achieved in the medical informatics community and how these approaches match the requirements for effective plan management. For each project, we first mention the general design, the plan representation, the application area, and then compare the features with the requirements defined in Section 2.2.

5.1. Arden Syntax and Medical Logic Modules (MLMs)

General Design. A group of investigators, working for the American Society for Testing and Materials (ASTM), have defined a standard procedural language, known as the Arden syntax [26]. The Arden syntax encodes situation-action rules. Developers of the Arden syntax have promoted this Pascal-like language because of the pressing needs to facilitate exchange of protocols among health-care institutions using existing software technology.

The Arden syntax is mostly used for alert- or reminder-systems.

Plan Representation. In the Arden syntax a plan/protocol is modeled by a set of Medical Logic Modules (MLMs). MLMs were originally designed to have a single set of data for input, a single application of criteria logic, and a single set of resulting actions. A single MLM is thus not sufficient for the representation of complex guidelines, because protocols consist of sequences of actions. Each sequence requires its own set of data including dependencies between different actions and decisions. However, efforts have been made to implement guidelines by chaining together multiple MLMs [64]. In this approach a guideline is modeled by a set of MLMs that are sequentially executed, and each MLM implements one single step within the guideline. Typically MLMs interact with the physician through messages.

MLMs use various kinds of elements to structure the control flow: a “data slot” to specify the links to patients’ records; an “evoke slot” to define what triggers a MLM; and a “logic slot” to determine whether the one action defined within the “action slot” should be done. Additionally, there are slots concerning the maintenance of the module and references to the literature.

Application Area. Examples of application areas are hypokalemia with digoxin therapy, tuberculosis cultures with positive or invalid results, calculation of creatinine clearance, and identification of patient records that include admissions but no discharge summaries.

Comparison. The Arden syntax and MLMs are a well-known standard in the medical informatics community. However, this standard has significant limitations: The language currently supports only atomic data types, misses any form of data structure and lacks defined semantics for creating temporal dimensions and comparing them. It provides no principled way to represent clinical protocols that are more complex than individual situation-action rules. There is no way to represent concurrent and cyclical execution of plans or compulsory and optional plans. A protocol is implicitly modeled by a very loose coupling of independent MLMs, connected exclusively by the fact that some of them refer to the output of others. Only a filter-condition is used, but there is no monitoring of the protocol’s

execution. Additionally, the approach does not allow the representation of knowledge roles. There is no concept of goals or intentions, policies or resource constraints, nor effects. Finally, no context annotation is possible.

According to their limited plan representation, this approach supports simple command-based editing of protocols, simple descriptions of domain-specific annotations in free text as comments or Pascal-like functions, and plan execution. Therefore, the Arden syntax and the MLMs are not applicable for more complex issues described in Section 2.2.

5.2. The EON Approach

General Design. The EON approach defines an architecture that is intended to support the development of applications for protocol-based care [48; 76]. The EON system is composed of a set of cooperating components: Components serve to select the set of protocols that may be applied to a certain patient and give information about a patient's state within a certain executing protocol.

Similar to MLMs (compare Section 5.1), EON provides recommendations about the treatment of patients, and allows for commenting on the interventions performed by the practitioner if they differ from those recommended by the protocol.

Plan Representation. The elementary knowledge structures used within EON for the representation of a protocol are:

- *procedures*, are used to implement the clinical algorithm underlying a protocol;
- *protocol steps*, may be intervention steps that specify an intervention or they may be assessment steps that specify data to be collected to assess or diagnose a patient. Intervention and assessment steps may also be combined within one protocol step. The term "intervention" in medicine corresponds to action in planning;
- *intervention states*, are types of interventions, which may be in one of the states active, completed, aborted, or suspended to indicate the progress of the intervention;
- *eligibility criteria*, allow the identification of patients to whom a protocol may be applied and are represented explicitly within an EON protocol;
- *selection conditions*, define when to proceed from one protocol step to the next and which protocol step to choose from a set of alternatives
- *intervention rules*, define the domain-specific attributes of protocols; and
- *revision rules*, are instructions that define how to change the state of an intervention or modify attribute values such as the dose of a prescribed drug.

EON protocols have a hierarchical structure, where each protocol may be decomposed into a set of finer-granularity protocols. A protocol is composed of a declarative and a procedural component: The declarative component defines parts of the protocol, their properties, and the relationships among them, whereas the procedural component specifies the temporal sequencing, branching, and looping of prescribed or suggested treatment interventions. The declarative part is modeled by a so-called ontology, which provides specific classes of protocols for different clinical domains. The procedural part is defined and visualized by means of a directed graph and an associated state model.

Application Area. Examples of protocols implemented in EON are clinical-trial protocols for breast-cancer treatment and protocols for the management of AIDS patients within the T-HELPER system [47].

Comparison. EON has a very powerful and knowledge-rich representation of protocols: Hierarchical decomposition in the sense of skeletal plans and a well-situated control structure are provided. The execution model provides a set of operators to guide the control flow of a protocol, such as sequencing, looping, and parallel actions. The representation of the procedural aspect of a protocol by means of a graph allows instant visualization and thereby alleviates an intuitive understanding of the guideline logic.

The drawbacks lie in the detail. A lot of knowledge-roles are simple free-text. For example, EON does not allow a structured specification of the intention or goal concept. An intention is a simple annotation in free text. A suitable (semi-) automated plan management cannot utilize this free-text annotations, which results in poor possibilities of plan management (compare Table 2). Additionally, the flowchart representation of the protocol's logic sets a limit to the capabilities of the EON approach: When it comes to the implementation of complex guidelines, incorporating features like e.g., temporal uncertainty, a representation of the protocols's logic by means of a graph will not be sufficient [32]. It simply does not provide the necessary flexibility in such cases. Finally, no context annotation is possible.

5.3. The GuideLine Interchange Format (GLIF) Approach

General Design. The GuideLine Interchange Format (GLIF) is a model for representing protocols in a machine-readable format [52]. The main goal in the development of GLIF was to create a standard for representing protocols that facilitates protocol sharing across the software tools at different medical institutions that manipulate, analyze, or otherwise compute with an electronic representation of a health-care protocol. GLIF is the result of a joined effort of researchers from three different US universities (Columbia, Harvard and Stanford), called the InterMed Collaboratory. The basis of GLIF was provided by the analysis of four existing systems. The examined systems were MLM / Arden [26], EON [48; 76], MBTA [4], and GEODE-CM [67].

GLIF provides a standard to share protocols among various protocol-based care systems.

Plan Representation. The GLIF approach defines a specific data model, which comprises a set of classes for guideline entities, attributes of those classes, and data types for the attribute values. The topmost object of the model, which is the GLIF guideline object, contains a name, a list of authors, a characterization of the guideline's intention, a specification of the patient-eligibility criteria, an unordered list of all steps in the guideline, an indication of the starting step in the guideline, and a list of supporting didactic material. This syntax is based on a generic data interchange format called ODIF standing for Object Data Interchange Format [53], which allows the representation of object-instances in text.

Sequences of decisions and actions can be explicitly defined within GLIF by specifying a collection of steps for a guideline, and by defining the starting step thereof. *Guideline steps* can be either "action steps" or "decision steps", the latter being divided into "conditional steps", "branch steps", and "synchronization steps". Beginning with the starting step, each guideline step comprises its successor step(s) within the specified sequence.

Application Area. Examples of guidelines modeled with GLIF are guidelines for influenza vaccination, for cholesterol screening and management, for breast-mass workup, and for breast-cancer treatment protocol.

Comparison. GLIF benefits from combined expertise of various members of institutions, who are medical and technical specialists in protocol-based care and who analyze four existing systems and extract the essential concepts. However, similar drawbacks as mentioned for the EON approach are obvious (compare Section 5.2): Several basic concepts within the GLIF model are currently represented only as strings of free text (e.g., intention, the description of an action, any kind of temporal information, or GLIF's criterion logic). Without a structured encoding of these concepts, most tasks relevant to plan management will not be feasible (compare Table 2). Several types of temporal uncertainty are not addressed in the current version of GILF (GLIF-2, version 3). However, the next version of GLIF should solve some of above problems.

We have discussed the benefits and limitations of planning approaches within the planning and the medical informatics communities. The next section describes, how our approach, called Asgaard/Asbru Project, meets the demands for plan management as listed in Section 2.2.

6. Building the Bridge: The Asgaard/Asbru Project

General Design. The **Asgaard/Asbru**[*] project [46; 62; 63] outlines some useful task-specific problem-solving methods to support both designers and executors of skeletal plans. The Asgaard/Asbru project tries to build the bridge between the planning approaches and the medical approaches, addressing the demands of the medical staff on the one side and applying rich plan management on the other side. The project concentrates on supporting therapeutic issues. Within our plan-representation language **Asbru**, we define the knowledge roles needed for problem-solving methods to support the various tasks of the plan management (see Table 3 and 4). Asgaard is a hierarchical planner based on skeletal plans, searching the plan space. The problem-solving methods correspond to the tasks of the plan management. These tasks are to be accomplished in a highly intertwined way. However, some of them are mostly done during *design time* and others during *execution time* of skeletal plans:

[*] In Norse mythology, Asgaard was the home and citadel of the gods, corresponding to Mount Olympus in Greek mythology. It was located in the heavens and was accessible only over the rainbow bridge, called Asbru (or Bifrost).

Methods mostly done at design time

- ***Plan Generation:*** fits suitable skeletal plans together;
- ***Advanced Plan Editing:*** is an editor guided by the knowledge-roles to give full access to expert users, who tune features of the knowledge roles;
- ***Domain-Specific Annotations:*** supplies structured support to write domain assumptions or domain functions;
- ***Plan Verification:*** examines the correctness of interrelated skeletal plans by a three-level detection of anomalies (*method semantic*s);
- ***Plan Validation:*** compares the intended states against the prescribed actions and intended plans (*domain semantics*);
- ***Plan-Senario Testing:*** applies scenarios of protocols to test their functionalities and their course of activities; it is partly included in the plan visualization;
- ***Plan Visualization:*** is a graphical editor to design and to browse the topological and the temporal views of the connected plans.

Methods mostly done at execution time

- ***Plan Selection:*** chooses applicable skeletal plans from the plan library according to the patient's state, the plan's overall intentions, and plan effects;
- ***Plan Adaptation:*** adjusts the parameters of a skeletal plan according to the patient's state and the medical environment;
- ***Plans Execution:*** performs according to the execution-plan's prescribed actions
- ***Plans Monitoring:*** oversees and administers whether the executed plans are still applicable at the particular time of execution at the sampling frequencies given in the temporal patterns; executed plans can be suspended and reactivated;
- ***Plan Modification / Alternatives:*** chooses alternative actions or plans, which are relevant at this time for achieving a given intention
- ***Plan Critiquing:*** (a) *recognition of intentions*: Why is the executing agent executing a particular set of actions, especially if those actions deviate from the skeletal plan's prescribed actions; (b) *critique of the executing agent's actions*: Is the executing agent deviating from the prescribed actions or intended plan? Are the deviating actions compatible with the author's plan and state intentions?
- ***Plan Evaluation:*** examines retrospectively, if the executed skeletal plans achieved the desired effects according to the patient's state, intentions, executed actions, and plan effects;
- ***Executed Plan Visualization:*** visualizes, which plans have been executed (when and how);
- ***Plan Rationale / History:*** bookkeeping of events, states, intentions and performed actions; generates on-line help information and explanations about plan selection, adaptation, execution states, success and failure of plans integrating the domain-specific annotations.

Plan Representation. We have developed a time-oriented, intention-based, skeletal plan-specification language, called **Asbru**, specific to the set of plan-management tasks [46]. Asbru enables the designer to represent both the prescribed actions of a skeletal plan and the knowledge roles required by the various problem-solving methods performing the intertwined supporting sub-tasks.

The major features of Asbru are that

- prescribed actions and states can be continuous;
- intentions, conditions, and world states are temporal patterns
- uncertainty in both temporal scopes and parameters can be flexibly expressed by bounding intervals;
- plans might be executed in sequence, all plans or some plans in parallel, all plans or some plans in a particular order, or periodically;
- particular conditions are defined to monitor the plans' execution; and
- explicit intentions and preferences can be stated for each plan separately.

A temporal pattern is either a *parameter proposition* — a parameter (or its abstraction), its value, a context, and a time annotation (e.g., the *state* abstraction of the blood-gas parameter is *normal*, as defined in the context of weaning therapy, during a certain time period) — , a *combination* of multiple parameter propositions, or a *plan state* associated to an instantiated plan (plan pointer) and a time annotation.

The time annotations used allow to represent uncertainty in starting time, ending time, and duration ([9], [59]). The time annotation supports multiple time lines (e.g., different zero-time points and time units) by providing arbitrary *reference annotations*. Temporal shifts from the reference annotation are used to define the uncertainty in starting time, ending time, and duration. We allow incomplete time annotation, denoted by an underscore "_". Different short-

cuts are used to simplify complex time annotations (e.g., the symbol "*" is used to allow that a condition holds during the span of time over which the plan is executed). To allow temporal repetitions, sets of cyclical time points and cyclical time annotations are defined.

The skeletal plans are uniformly represented and organized in the *plan-specification library*. A *plan* in the plan-specification library is composed hierarchically of a set of plans with arguments and time annotations.

A plan consists of a name, a set of arguments, including a time annotation (representing the temporal scope of a plan), and five components: *preferences*, *intentions*, *conditions*, *effects*, and a *plan body* (*layout*), which describes the actions to be executed. The plan name is compulsory and all other components are optional. Table 3 explains the important knowledge roles of Asbru.

Table 3 Components of Asbru. Describing the several knowledge roles needed for plan management.

Knowledge Roles	**Possible Elements are**	**Explanation**
Preferences		*constrain the selection of a plan*
	`strategy`	a general strategy for dealing with the problem
	`utility`	a set of utility measures
	`select-method`	a matching heuristic for the applicability of the whole plan
	`resources`	a set of prohibited, recommended, discouraged, and obligatory resources
	`responsible-actor`	a set of actors, who are entitled to adapt the protocols (e.g., physician, nurse)
Intentions		*are high-level goals at various levels of the plan, an annotation specified by the designer; intentions are temporal patterns that should be* maintained, achieved, *or* avoided
	`intermediate-state`	the state(s) that should hold *during* the applicability of the plan
	`intermediate-action`	the action(s) that should take place *during* the execution of the plan
	`overall-state-pattern`	the overall pattern of state(s) that should hold *after* finishing the plan
	`overall-action-pattern`	the overall pattern of action(s) that should hold *after* finishing the plan
Conditions		*are temporal patterns, sampled at a specified frequency, that need to hold at particular plan steps to induce a particular state transition of the plan instance; we specify different conditions that enable transition from one plan state into another (see Figure 2)*
	`filter-preconditions`	the preconditions which need to hold initially if the plan is applicable, but can not be achieved by any action (e.g., sex of a patient) and are necessary for a plan to become `possible`
	`setup-preconditions`	the preconditions which need to be achieved to enable a plan to start and which allow a transition from a `possible` plan to a `ready` plan
	`activate-condition`	a token which determines if the plan should be started manually or automatically
	`suspend-conditions`	the conditions which determine when an `activated` plan has to be suspended – certain conditions (protection intervals) need to hold
	`abort-conditions`	the conditions which determine when an `activated`, `suspended`, or `reactivated` plan has to be aborted
	`complete-conditions`	the conditions which determine when an `activated` or reactivated (suspended and re-activated) plan can be completed successfully
	`reactivate-conditions`	the conditions which determine when a `suspended` plan has to be `reactivated`, i.e., continued
Effects		*describe the possible effects of plans*
	`functional relationship`	relationship between the plan arguments and measurable parameters
	`overall effect`	overall effect of a plan on parameters independent of plan's arguments

Table 3 (continued)

Knowledge Roles	Possible Elements are	Explanation
Plan-body (layout)		*is a set of plans to be executed in sequence, in parallel, in any order, or in some frequency*
	`Sequential (do-all-sequentially)`	a set of plans that are executed in sequence (executed in total order) – all plans must be completed; no `continuation-condition`
	`Sequential (do-some-sequentially)`	a set of plans that are executed in sequence (executed in total order) – some plans must be completed; the `continuation-conditions` specify a subset of plans, which must be completed
	`concurrent : parallel (do-all-together)`	a set of plans that are executed in parallel – all plans must start together; no `continuation-condition`
	`concurrent : parallel (do-some-together)`	a set of plans that are executed in parallel – some plans must start together; the `continuation-conditions` specify a subset of plans, which must be completed
	`concurrent : any order (do-all-any-order)`	a set of plans that are executed in any order – all plans must be completed; no `continuation-condition`
	`concurrent : any order (do-some-any-order)`	a set of plans that are executed in any order – some plans must be completed; the `continuation-conditions` specify a subset of plans, which must be completed
	`cyclical (every)`	a repeated plan with optional temporal and continuation arguments that can specify its behavior

Plan-State Model. All plans and actions are durative, therefore, a set of mutually exclusive *plan states* describes the actual status of the plan during the plan selection and the plan execution. Particular *state-transition criteria* specify transition between neighboring plan states. Figure 2 illustrates the different plan states and their corresponding transition criteria shown on the arrows. The meaning of the state-transition criteria is explained in Table 3. We distinguish plan states during the plan-selection phase (left-hand side of Figure 2) and plan states during the execution phase (right-hand side of Figure 2). For example, if a plan has been `activated`, it can only be `completed`, `suspended`, or `aborted` depending on the corresponding criteria. The gray triangle on the right-hand side of Figure 2 includes the three basic states; they should always be defined.

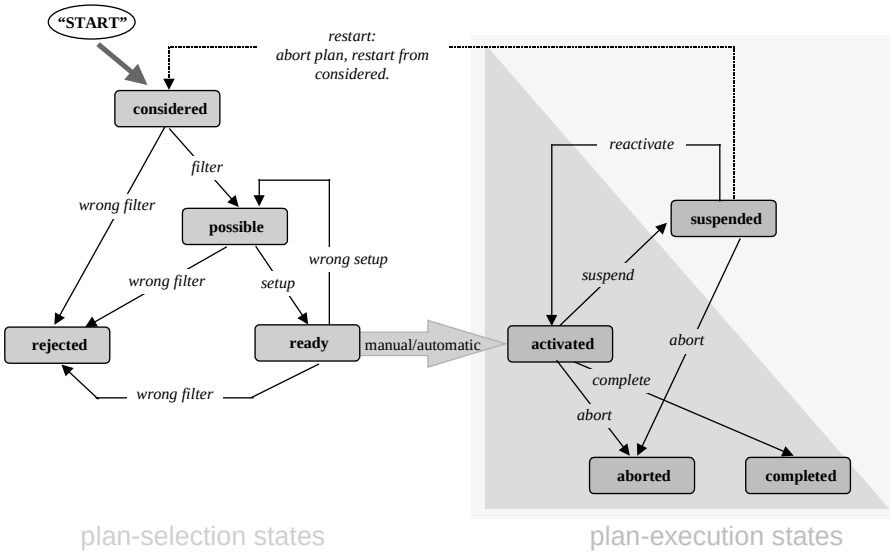

Figure 2 Plan-state model. Plan states and state-transition criteria.

Propagation of Plan States. The plan states, which are stopping the execution of a plan (successfully or unsuccessfully: `rejected`, `completed`, `aborted`, and `suspended` states), may be propagated in both directions: the parent plan propagates its plan states to its children; a child plan propagates its plan state only if it is *relevant* for the parent's completion. The propagation is done over multiple hierarchies. A plan is called relevant, if it is included in the `continuation-conditions` (see Table 3). In case of sequential plans all sub-plans are relevant. Internally, the plan monitor keeps track whether a plan state results from the appropriate conditions or from the propagation (e.g., let plan A_a be a child plan of plan A and plan A_a be the parent plan of plan A_{aa}. If plan A_a's `abort-conditions` become true, plan A_a propagates its state to its child plan A_{aa}, internally called "parent-stopped". If plan A_a is relevant for its parent plan A, it further propagates its state to its parent plan A, internally called "children-stopped".).

Example I-RDS written in Asbru. After infants' respiratory distress syndrome (I-RDS) is diagnosed, a plan dealing with limited monitoring possibilities is activated, called *initial-phase*. Depending on the severity of the disease, three different kinds of plans are followed: *controlled-ventilation*, *permissive-hypercapnia*, or *crisis-management*. Only one plan at a time can be activated, however the order of execution and the activation frequency of the three different plans depend on the severity of the disease. Additionally, it is important to continue with the plan *weaning* only after a successful completion of the plan *controlled-ventilation*. After a successful execution of the plan *weaning*, the extubation should be initiated. The extubation can be either a single plan *extubation* or a sequential execution of the sub-plans *cpap* and *extubation*.

The most important part is the sub-plan *controlled-ventilation.* The intentions of this sub-plan are to maintain a normal level of the blood-gas values and the lowest level of mechanical ventilation (as defined in the context of controlled ventilation therapy) during the span of time over which the sub-plan is executed. This sub-plan is activated immediately, if peak inspiratory pressure PIP ≤ 30 and the transcutaneously assessed blood-gas values are available for at least one minute after activating the last plan instance *initial-phase* (as reference point). The sub-plan must be aborted, if PIP > 30 or the increase of the blood-gas level is too steep (as defined in the context of controlled ventilation-therapy) for at least 30 seconds. The sampling frequency of the abort condition is 10 seconds. The sub-plan is completed successfully, if $F_iO_2 \leq 50\%$, PIP ≤ 23, f ≤ 60, the patient is not dyspnoeic, and the level of blood gas is normal or above the normal range (as defined in the context of controlled ventilation-therapy) for at least three hours. The sampling frequency of the complete condition is 10 minutes. The body of the sub-plan *controlled-ventilation* consists of a sequential execution of the two sub-plans *one-of-increase-decrease-ventilation* and *observing*.

This part of the I-RDS protocol would be written in Asbru:

```
(PLAN I-RDS-therapy  ...
(DO-ALL-SEQUENTIALLY
   (initial-phase)
   (one-of-controlled-ventilation)
   (weaning)
   (one-of-cpap-extubation)))

(PLAN one-of-controlled-ventilation ...
(DO-SOME-ANY-ORDER
   (controlled-ventilation)
   (permissive-hypercapnia)
   (crisis-management)
    CONTINUATION-CONDITION controlled-ventilation))

(PLAN controlled-ventilation
(PREFERENCES (SELECT-METHOD BEST-FIT))
(INTENTION:INTERMEDIATE-STATE
   (MAINTAIN STATE(BG) NORMAL controlled-ventilation *))
(INTENTION:INTERMEDIATE-ACTION
   (MAINTAIN STATE(RESPIRATOR-SETTING) LOW controlled-ventilation *))

(SETUP-PRECONDITIONS
   (PIP (<= 30) I-RDS *now*) (BG available I-RDS
      [[_, _], [_, _],[1 MIN,_] (ACTIVATED initial-phase-1#)]))
(ACTIVATED-CONDITIONS AUTOMATIC)
```

```
(ABORT-CONDITIONS ACTIVATED
   (OR (PIP (> 30) controlled-ventilation  [[_, _], [_, _], [30 SEC, _], *self*])
       (RATE(BG) TOO-STEEP controlled-ventilation
                [[_, _], [_, _], [30 SEC,_], *self*])))
   (SAMPLING-FREQUENCY 10 SEC))
(COMPLETE-CONDITIONS
   (FiO2 (<= 50) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
   (PIP (<= 23) controlled-ventilation  [[_, _], [_, _], [180 MIN, _], *self*])
   (f (<= 60) controlled-ventilation [[_, _], [_, _], [180 MIN, _], *self*])
   (STATE(patient) (NOT DYSPNOEIC) controlled-ventilation
        [[_, _], [_, _], [180 MIN, ], *self*]))
   (STATE(BG) (OR NORMAL ABOVE-NORMAL) controlled-ventilation
        [[_, _], [_, _], [180 MIN,_], *self*])
   (SAMPLING-FREQUENCY 10 MIN))
(DO-ALL-SEQUENTIALLY
   (one-of-increase-decrease-ventilation)
   (observing)))
```

Application Area. The Asgaard system as a whole is designed to support a wide range of medical applications from intensive care units to competitive sports. Parts of the Asgaard project have been applied to clinical treatment protocol for artificial ventilation of newborn infants and for gestational diabetes mellitus.

Table 4 The knowledge roles used in the several tasks of plan management.

Knowledge Roles	**Possible Elements are**	**Tasks of the Plan Management**															
		Design Time							Execution Time								
		Plan Generation	Advanced Plan Editor	Domain-Specif. Annotations	Plan Verification	Plan Validation	Plan-Scenario Testing	Plan Visualization	Plan Selection	Plan Adaptation	Plan Execution	Plan Monitoring	Plan Modification	Plan Critiquing	Plan Evaluation	Executed Plan Visualization	Plan Rationale / History
Preferences																	
	strategy	×	×	×	×	×	×	×	×			×	×		×	×	×
	utility	×	×	×	×	×	×	×	×	×		×	×		×	×	×
	select-method	×	×	×	×	×	×	×	×		×	×	×		×	×	×
	resources	×	×	×	×	×	×	×	×			×	×		×	×	×
	responsible-actor	×	×	×	×	×	×	×	×			×	×		×	×	×
Intentions																	
	intermediate-state	×	×	×	×	×	×	×				×	×	×	×	×	×
	intermediate-action	×	×	×	×	×	×	×				×	×	×	×	×	×
	overall-st.-pattern	×	×	×	×	×	×	×	×			×	×	×	×	×	×
	overall-act.-pattern	×	×	×	×	×	×	×	×			×	×	×	×	×	×
Conditions																	
	filter-preconditions	×	×	×	×	×	×	×	×			×	×		×	×	×
	setup-preconditions	×	×	×	×	×	×	×		×		×			×	×	×
	activate-condition	×	×	×	×	×	×	×			×	×			×	×	×
	suspend-conditions	×	×	×	×	×	×	×				×			×	×	×
	abort-conditions	×	×	×	×	×	×	×				×			×	×	×
	complete-conditions	×	×	×	×	×	×	×				×	×		×	×	×
	reactivate-cond.	×	×	×	×	×	×	×				×			×	×	×
Effects																	
	funct. relationship	×	×	×	×	×	×	×	×				×	×	×	×	×
	overall effect	×	×	×	×	×	×	×	×				×	×	×	×	×
Plan-body																	
	sequential	×	×	×	×	×	×	×		×	×	×		×	×	×	×
	concurrent	×	×	×	×	×	×	×		×	×	×		×	×	×	×
	cyclical	×	×	×	×	×	×	×		×	×	×		×	×	×	×

Comparison. While other planning systems restrict planning as an exhausting search in state space or in plan space, Asgaard is not intended to solve classical problems involving search. Asgaard concentrates on a second dimension of planning, namely plan management with its huge number of sub-tasks. It is designed for medical applications of plan management. In this field direct control over the planning process is crucial, since the user community, namely the medical staff, would refuse a system which does not offer maximum transparency and safety issues in the decision process. Instead of unrestricted or heuristic search in state space, suitable plans are found through search in plan space applying meta-plans called "skeletal plans". These plans prescribe a set of sub-targets to be reached by means of conditions which must be fulfilled. While the skeleton of the final solution is defined by the skeletal plan, it does not define exactly, which sub-plans to select to reach the target. Instead, such plans are found by matching their intentions, their condition as well as the context, for which they are defined, with those demanded by the skeletal plan.

Asbru places particular emphasis on an expressive representation of time-oriented actions and world states in combination with the underlying intentions as temporal patterns to be maintained, achieved or avoided. It supports the use of different granularities and reference points to represent multiple time lines. Asbru's representation includes annotations for the duration of actions, their success or failure, and allows time annotation of events, actions/plans, and world states with uncertainty in their appearances. Asbru allows the definition of optional plans as well as mutual exclusive alternatives. It provides a set of temporal relations between sub-plans which exceeds parallel, sequential and cyclical execution, namely do-some-any-order, do-some-together. Preferences, intentions, conditions, effects, and actions are specified on various detail levels depending on their appearances and evidences. Such a complex plan-specification language is needed to capture all requirements of a dynamically changing environment, such as medicine. The framework heavily stresses the notion of context. This concept is driven by medical practice too [42].

The Asgaard project tries to support the various tasks of plan management described in Section 2.2. In this section we showed how Asgaard met these tasks applying particular knowledge roles, which are defined in the plan-specification language Asbru. We have implemented and scenario-based evaluated a database layer, called Skid, to administer the skeletal plan library efficiently [23], the advanced plan editor and AsbruView, our graphical editor to design and to browse the topological and the temporal views of the plans written in Asbru [31; 43], and time-oriented data abstraction methods to arrive at intuitive qualitative descriptions of temporal patterns [44; 45]. We have developed knowledge-based methods based on a three-level detection of anomalies to verify and validate Asbru plans [10; 11]. Futher research has been done by refining the system architecture to accomplish the complex intertwined plan-management process and by focusing on the plan executor and monitor.

First evaluation results are very encouraging, that our proposed solution within Asgaard/Asbru project truly meets the demands of the medical staff and its concept is comprehensible and lucid to computer scientists too. Colleagues from Amsterdam, who were not involved in the designing process of our Asgaard/Asbru project, have written protocols in Asbru. They were looking at protocols of general practitioners in detail. The team involved four people with different backgrounds, namely AI, medical informatics, and pediatrics. They wrote the same general practitioners' protocol (the sinusitus protocol) in Asbru independently, which resulted in three versions of the Asbru's plan hierarchies. We compared these three versions. All three experts came up with roughly the same structure of protocols written in Asbru.

7. Conclusion

Planning in the medical domain is different to classical planning due to the virtue of the field of medicine. While there is a conceptual solution to most planning problems, the details are often unpredictable since the system to be modeled – the human body – incorporates many processes which can be qualitatively described, but the exact value of parameters is often unknown.

The amount of available though vague domain knowledge distinguishes medical planning from reactive planning with which it shares the need to react on non-deterministic changes in the environment.

The field of medical informatics supplied a series of representations for medical guidelines – the pendant to plans in AI. While recent approaches provide sufficient means to express medical knowledge in its vagueness and complexity, they lack computer support for adaption to changes in the environment during execution of a plan, which is needed in effective plan management.

In this paper we described requirements for a system, which could provide both powerful knowledge modeling facilities and execution time assistance as needed in the medical domain and how the Asgaard/Asbru project matches these requirements of plan management. We have very favorable evaluation results, that our approach truly meets the desired requirements. Independent people with different backgrounds acquired roughly the same structure of protocols written in Asbru.

Acknowledgments

The author thanks Georg Duftschmid, Johannes Gärtner, Klaus Hammermüller, Werner Horn, Peter Johnson, Robert Kosara, Franz Paky, Christian Popow, Andreas Seyfang, Yuval Shahar, who contribute to the development of the project and Frank van Harmelen, Annette Ten Teije and their medical collaborators who have examined Asgaard/Asbru very closely. This project is supported by "Fonds zur Förderung der wissenschaftlichen Forschung" (Austrian Science Fund), P12797-INF.

References

[1] Akman, V. and Surav, M.: Steps Toward Formalizing Context. *AI Magazine,* **17**(3) (1996), 55-72.

[2] Allen, J. F.: Towards a General Theory of Action and Time. *Artificial Intelligence,* **23**(2) (1984), 123-154.

[3] Allen, J. F. and Koomen, J. A.: Planning Using a Temporal World Model, in: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-88)*, 1988.

[4] Barnes, M. and Barnett, G. O.: An Architecture for a Distributed Guideline Server, in: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, Gardner, R. M., ed New Orleans, Louisiana, Hanley & Belfus, 1995, pp. 233-237.

[5] Breuker, J. and Velde, W. v. d. (Eds.): *CommonKADS Library for Expertise Modelling*. Amsterdam: IOS Press, 1994.

[6] Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M. and Wang, X.: *PRODIGY 4.0: The Manual and Tutorial*, Department of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-92-150, 1992.

[7] Dean, T., Firby, R. J. and Miller, D.: Hierarchical Planning Involving Deadlines, Travel Time, and Resources. *Computational Intelligence*, **4**(4) (1988), 381-398.

[8] Dean, T. L. and McDermott, D. V.: Temporal Data Base Management. *Artificial Intelligence*, **32**(1) (1987), 1-55.

[9] Dechter, R., Meiri, L. and Pearl, J.: Temporal Constraint Networks. *Artificial Intelligence, Special Volume on Knowledge Representation*, **49**(1-3) (1991), 61-95.

[10] Duftschmid, G.: *Knowledge-based Verification of Clinical Guidelines by Detection of Anomalies.* Ph.D. Thesis, Vienna University of Technology, 1999.

[11] Duftschmid, G., Miksch, S., Shahar, Y. and Johnson, P.: Multi-Level Verification of Clinical Protocols, in: *Proceedings of the Workshop on Validation & Verification of Knowledge-Based Systems (V&V'98), in conjunction with the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, Trento, Italy, 1998, pp. 4/1-4/10.

[12] Fikes, R. E., Hart, P. E. and Nilsson, N. J.: Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, **3**(4) (1972), 349-371.

[13] Fikes, R. E. and Nilsson, N. J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, **2**(3/4) (1971), 189-208.

[14] Firby, R. J.: *Adaptive Execution in Complex Dynamic Worlds.* Ph.D. Thesis, Technical Report, YALEU/CSD/RR #672 Thesis, Yale University, 1989.

[15] Firby, R. J.: *The RAP Lanauge Manual Animate Agent Project*,University of Chicago, Working Note AAP-6, 1995.

[16] Fox, J., Johns, N. and Rahmanzadeh, A.: Protocols for Medical Procedures and Therapies: A Provisional Description of the PROforma Language and Tools, in: *Proceedings of the Artificial Intelligence in Medicine, 6th Conference on Artificial Intelligence in Medicine Europe (AIME-97)*, Keravnou, E., Garbay, C., Baud, R. and Wyatt, J., eds, Grenoble, France, March 23-26, Springer, Berlin, 1997, pp. 21-38.

[17] Fox, M. and Long, D.: STAN Homepage. *http://www.dur.ac.uk/~dcs0www/personnel/dcs0mf.html*, (1998).

[18] Fox, M. S.: ISIS: A Retrospective. in: *Intelligent Scheduling*, Zweben, M. and Fox, M. S., eds, San Francisco, CA: Morgan Kaufmann, 1994, pp. 3-28.

[19] Fox, M. S., Allen, B. and Strohm, G.: Job-Shop Scheduling: An Investigation in Constraint-Directed Reasoning, in: *Proceedings of the Second Annual National Conference on Artificial Intelligence (AAAI-82)*, Pittsburg, Pennsylvania, American Association for Artificial Intellignce, 1982, pp. 155-8.

[20] Friedland, P. E.: *Knowledge-Based Experiment Design in Molecular Genetics.* PhD Thesis, Computer Science Department, Stanford University, Stanford, CA, 1979.

[21] Friedland, P. E. and Iwasaki, Y.: The Concept and Implementaion of Skeletal Plans. *Journal of Automated Reasoning*, **1**(2) (1985), 161-208.

[22] Georgeff, M. P., Ingrand, F. F. and Lanskey, A. L.: *Procedural Reasoning System User Guide*,SRI International Artificial Intelligence Center, 1989.

[23] Hammermüller, K.: *Design einer Datenbank für kontextspezifische Daten, Pläne und deren zeitlichen Verlauf.* Master Thesis, Vienna University of Technology, 1998.

[24] Hammond, P., Sergot, M. J. and Wyatt, J. C.: Formalisation of Safety Reasoning in Protocols and Hazard Regulations, in: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, Gardner, R. M., ed, New Orleans, Louisiana, Hanley & Belfus, 1995.

[25] Herbert, S. I., Gordon, C. J., Jackson-Smale, A. and Renaud Salis, J.-L.: Protocols for Clinical Care. *Computer Methods and Programs in Biomedicine*, **48** (1995), 21-26.

[26] Hripcsak, G., Ludemann, P., Pryor, T. A., Wigertz, O. B. and Clayton, P. D.: Rationale for the Arden Syntax. *Computers and Biomedical Research,* **27** (1994), 291-324.

[27] Jha, K. N.: *Specification of the Process Specification Language (PSL),*ACT, Boeing Corp., National Institute of Standards and Technology (NIST), August 17, 98, 1998.

[28] Karp, P. D., Lowrance, J. D., Strat, T. M. and Wilkins, D. E.: The Grasper-CL Graph Management System. *LISP and Symbolic Computation: An International Journal,* **7** (1994), 251-290.

[29] Kautz, H. and Selman, B.: BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving, in: *Proceedings of the Working Notes of the Workshop on Planning as Combinatorial Search, held in conjunction with AIPS-98, Pittsburgh, PA*, 1998.

[30] Koehler, J., Nebel, B., Hoffmann, J. and Dimopoulos, Y.: Extending Planning Graphs to an ADL Subset, in: *Proceedings of the Fourth European Conference on Planning ECP '97 (EWSP '97),*, Toulouse, France, September 24-26,1997, Springer, Berlin, 1997.

[31] Kosara, R. and Miksch, S.: Visualization Techniques for Time-Oriented, Skeletal Plans in Medical Therapy Planning, in: *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, Horn, W., Shahar, Y., Lindberg, G., Andreassen, S. and Wyatt, J., eds, Aalborg, Denmark, Springer, Berlin, 1999, pp. 291-300.

[32] Kosara, R., Miksch, S., Shahar, Y. and Johnson, P.: AsbruView: Capturing Complex, Time-oriented Plans - Beyond Flow-Charts, in: *Proceedings of the Second Workshop on Thinking with Diagrams 1998 (TwD'98)*, The University of Wales, Aberystwyth, United Kingdom, 1998, pp. 119-126.

[33] Kushmerick, N., Knoblock, C. A. and Yang, Q.: An Algorithm for Probabilistic Planning. *Artificial Intelligence, Special Volume on Planning and Scheduling,* **76**(1-2) (1995), 167-238.

[34] Leaper, D. J., Horrocks, J. C., Staniland, J. R. and DeDombal, F. T.: Computer-Assisted Diagnosis of Abdominal Pain Using "Estimates" Provided by Clinicians. *British Medical Journal,* **11**(November) (1972), 350-354.

[35] Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A. and Yost, G.: *The PIF Process Interchange Format and Framework Version 1.1*. MIT Center for Coordination Science., Technical Report Working Paper No 194, 1996.

[36] Lehrer, N. B.: *Knowledge Representation Specification Language (KRSL) - Version 2.0.2.* DARPA/Rome Laboratory Planning and Scheduling Initiative, 1993.

[37] Levinson, R.: A General Programming Language for Unified Planning and Control. *Artificial Intelligence, Special Volume on Planning and Scheduling,* **76**(1-2) (1995), 319-75.

[38] McCarthy, J. and Hayes, P. J.: Some Philosophical Problems from the Standpoint of Artificial Intelligence. in: *Machine Intelligence 4*, Meltzer, B., Michie, D. and Swann, M., eds, Edinburgh, Scotland: University Press, 1969, pp. 463-502.

[39] McDermott, D., Ghallab, M., Howe, A., Ram, A., Veloso, M., Weld, D. S. and Wilkins, D. E.: *PDDL - The Planning Domain Definition Language*, Yale Center for Computational Vision and Control, Tech Report CVC TR-98-003/DCS TR-1165, 1998.

[40] McDermott, D. and Hendler, J.: Planning: What it is, What it could be, An Introduction to the Special Issue on Planning and Scheduling. *Artificial Intelligence, Special Volume on Planning and Scheduling,* **76**(1-2) (1995), 1-16.

[41] Miksch, S., Horn, W., Popow, C. and Paky, F.: Context-Sensitive Data Validation and Data Abstraction for Knowledge-Based Monitoring, in: *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI 94)*, Cohn, A. G., ed, Amsterdam, Wiley, Chichester, UK, 1994, pp. 48-52.

[42] Miksch, S., Horn, W., Popow, C. and Paky, F.: Context-Sensitive and Expectation-Guided Temporal Abstraction of High-Frequency Data, in: *Proceedings of the Tenth International Workshop for Qualitative Reasoning (QR-96)*, Iwasaki, Y. and Farquhar, A., eds, May 21-24, Stanford Sierra Camp, Fallen Leaf Lake, CA, AAAI Press, 1996, pp. 154-163.

[43] Miksch, S., Kosara, R., Shahar, Y. and Johnson, P.: AsbruView: Visualization of Time-Oriented, Skeletal Plans, in: *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems 1998 (AIPS-98), June 7-10, 1998*, Carnegie Mellon University, Pittsburgh Pennsylvania, USA., AAAI Press, 1998, pp. 11-18.

[44] Miksch, S. and Seyfang, A.: Finding Intuitive Abstractions of High-Frequency Data, in: *Proceedings of the Workshop "Computers in Anaesthesia and Intensive Care: Knowledge-Based Information Management", in conjunction with the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, Aalborg, Denmark, 1999.

[45] Miksch, S., Seyfang, A., Horn, W. and Popow, C.: Abstracting Steady Qualitative Descriptions over Time from Noisy, High-Frequency Data, in: *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, Horn, W., Shahar, Y., Lindberg, G., Andreassen, S. and Wyatt, J., eds, Springer, Berlin, 1999, pp. 281-290.

[46] Miksch, S., Shahar, Y. and Johnson, P.: Asbru: A Task-Specific, Intention-Based, and Time-Oriented Language for Representing Skeletal Plans, in: *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, Motta, E., van Harmelen, F., Pierret-Golbreich, C., Filby, I. and Wijngaards, N., eds, Milton Keynes, UK, January 22-24, The Open University, Milton Keynes, UK, 1997, pp. 9/1-9/20.

[47] Musen, M. A., Carlson, C. W., Fagan, L. M., Deresinski, S. C. and Shortliffe, E. H.: T-HELPER: Automated Support for Community-Based Clinical Research, in: *Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care (SCAMC-92)*, Frisse, M. E., ed McGraw Hill, New York, NY, 1992, pp. 719-23.

[48] Musen, M. A., Tu, S. W., Das, A. K. and Shahar, Y.: EON: A Component-Based Approach to Automation of Protocol-Directed Therapy. *Journal of the American Medical Information Association*, **3**(6) (1996), 367-88.

[49] Musliner, D. J., Hendler, J. A., Agrawala, A. K., Durfee, E. H., Strosnider, J. K. and Paul, C. J.: The Challenges of Real-Time AI. *IEEE Computer*, **28**(1) (1995), 58-66.

[50] Myers, K. L. and Wilkins, D. E.: *The Act Formalism, Version 2.2*. SRI International Artificial Intelligence Center, 1997.

[51] Newell, A.: The Knowledge Level. *Artificial Intelligence*, **18** (1982), 87-127.

[52] Ohno-Machado, L., Gennari, J. H., Murphy, S., Jain, N. L., Tu, S. W., Oliver, D. E., Pattison-Gordon, E., Greenes, R. A., Shortliffe, E. H. and Barnett, G. O.: The GuideLine Interchange Format: A Model for Representing Guildelines. *Journal of American Medical Informatics*, **5**(4) (1998), 357-372.

[53] Pattison-Gordon, E.: *ODIF: Object Data Interchange Format*,Decision Systems Group, Brigham & Women's Hospital, DSG-96-04, 1996.

[54] Pattison-Gordon, E., Cimino, J. J., Hripcsak, G., Tu, S. W., Gennari, J. H., Jain, N. L. and Greenes, R. A.: *Requirements of a Sharable Guideline Representation for Computer Applications*,Stanford University, Report No. SMI-96-0628, 1996.

[55] Pease, R. A. and Carrico, T. M.: *Object Model Working Group (OMWG) Core Plan Representation - Request for Comment", Version 2, 24 January 1997*,Defense Advanced Research Projects Agency. (DARPA), 1997.

[56] Penberthy, J. S. and Weld, D. S.: UCPOP: A Sound, Complete, Partial-Order Planner for ADL, in: *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, Nebel, B., Rich, C. and Swartout, W., eds, Cambridge, MA, Morgan Kaufmann, San Mateo, 1992, pp. 103-114.

[57] Pollack, M. E. and Horty, J. F.: There's More to Life than Making Plans: Plan Management in Dynamic, Multi-agent Environements. *AI Magazine*, to appear (1999).

[58] Polyak, S.: *Requirements for a Rich, Shared Plan Representation*,Artificial Intelligence Applications Institute, University of Edinburgh, Scotland, 1997.

[59] Rit, J.-F.: Propagating Temporal Constraints for Scheduling, in: *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, Morgan Kaufmann, Los Altos, CA, 1986, pp. 383-388.

[60] Russell, S. and Norwig, P.: *Artificial Intellience: a Modern Approach* PTR Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[61] Shahar, Y., Miksch, S. and Johnson, P.: A Task-Specific Ontology for Design and Execution of Time-Oriented Skeletal Plans, in: *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop (Track KA for Temporal Reasoning and Planning)*, Gaines, B., ed Banff, Alberta, Canada, November 9-14, 1996.

[62] Shahar, Y., Miksch, S. and Johnson, P.: The Asgaard Project: A Task-Specific Framework for the Application and Critiquing of Time-Oriented Clinical Guidelines. *Artificial Intelligence in Medicine,* **14** (1998), 29-51.

[63] Shahar, Y. and Musen, M. A.: Plan Recognition and Revision in Protocol-Based Care, in: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, Gardner, R. M., ed New Orleans, Lousiana, 1995.

[64] Sherman, E. H., Hripcsak, G., Starren, J., Jender, R. A. and Clayton, P.: Using Intermediate States to Improve the Ability of the Arden Syntax to Implement Care Plans and Reuse Knowledge, in: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, Gardner, R. M., ed New Orleans, Louisiana, Hanley & Belfus, 1995, pp. 238-242.

[65] Simmons, R., Veloso, M. M. and Smith , S. (Eds.): *The Fourth International Conference on Artificial Intelligence Planning Systsems (AIPS-98).* AAAI Press, Menlo Park, CA, 1998.

[66] Simon, H. A.: On Reasoning about Actions. in: *Representation and Meaning: Experiments with Information Processing Systems*, Simon, H. A. and Siklossy, L., eds, New York: Prentice-Hall, 1972, pp. 414-430.

[67] Stoufflet, P., Ohno-Machado, L., Deibel, S., Lee, D. and Greenes, R.: GEODE-CM: A State-Transition Framework for Clinical Management, in: *Proceedings of the 20th Annual Symposium on Computer Applications in Medical Care*, Hanley and Belfus, Philadelphia, 1996.

[68] Tate, A.: *Project Planning Using a Hierarchical Nonlinear Planner*,Department of Artificial Intelligence, Report 25, 1976.

[69] Tate, A.: Putting Knowledge-Rich Plan Representations to Use., in: *Proceedings of the Papers of the 14th Machine Intelligence Workshop*, 1993.

[70] Tate, A.: *O-Plan: Task Formalism Manual*,Artificial Intelligence Application Institute, University of Edinburgh, UK, Manual, 1995.

[71] Tate, A.: Representing Plans as a Set of Constraints - the <I-N-OVA> Model, in: *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, Drabble, B., ed Edinburgh, Scotland, Morgan Kaufmann, 1996, pp. 221-228.

[72] Tate, A.: Roots of SPAR - Shared Planning and Activity Representation. *The Knowledge Engineering Review,* **13**(1) (1998).

[73] Tate, A., Drabble, B. and Dalton, J.: O-Plan: A Knowledge-Based Planner and its Application to Logistic. in: *Advanced Planning Technology*, Morgan Kaufmann, 1996, pp. 213-39.

[74] Tate, A., Drabble, B. and Kibry, R.: O-Plan2: An Open Architecture for Command, Planning, and Controll. in: *Intelligent Scheduling*, Zweben, M. and Fox, M. S., eds, Vol. 1, San Francisco, CA: Morgan Kaufmann, 1994, pp. 213-39.

[75] Tate, A., Hendler, J. and Drummond, M.: A Review of AI Planning Techniques. in: *Readings in Planning*, Allen, J. F., Hendler, J. and Tate, A., eds, Los Altos, CA: Morgan Kaufmann, 1990, pp. 26-49.

[76] Tu, S. W. and Musen, M. A.: The EON Model of Intervention Protocols and Guidelines, in: *Proceedings of the 1996 AMIA Annual Fall Symposium (formerly SCAMC), October 26-30, 1996*, Cimino, J. J., ed Washington, DC., Hanley & Belfus, Inc., Medical Publishers, Philadelphia, 1996, pp. 587-591.

[77] Uckun, S.: Instantiating and Monitoring Skeletal Treatment Plans. *Methods of Information in Medicine,* **35** (1996), 324-333.

[78] Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E. and Blythe, J.: Integrating Planning and Learning: The PRODIGY Architecture. *Journal of Theoretical and Experimental Artificial Intelligence,* **7**(1) (1995), 81-120.

[79] Veloso, M. M., Pollack, M. E. and Cox, M. T.: Rational-Based Monitoring for Planning in Dynamic Environments, in: *Proceedings of the The Fourth International COnference on Artificial Intelligence Planning Systsems (AIPS-98),* Simmons, R., Veloso, M. M. and Smith , S., eds, CMU, Pittsburgh, PA, AAAI Press, Menlo Park, CA, 1998, pp. 171-179.

[80] Vere, S.: Planning in Time: Windows and Durations for Activities and Goals. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* **3** (1983), 246-267.

[81] Vollebregt, A., ten Teije, A., van Harmelen, A., van der Lei, J. and Mosseveld, M.: A Study of PROforma, a Development Methodology for Clinical Procedures. *Artificial Intelligence in Medicine,* to appear (1999).

[82] Wilkins, D. E.: *Practical Planning* Los Altos, CA: Morgan Kaufmann, 1988.

[83] Wilkins, D. E.: Can AI Planners Solve Practical Problems? *Computational Intelligence,* **6**(4) (1990), 232-246.

[84] Wilkins, D. E. and Myers, K. L.: A Common Knowledge Representation for Plan Generation and Reactive Execution. *Journal of Logic and Computation,* **5**(6) (1995), 731-761.

[85] Wilkins, D. E., Myers, K. L., Lowrance, J. D. and Wesley, L. P.: Planning and Reacting in Uncertain and Dynamic Environments. *Journal of Experimental and Theoretical AI,* **7**(1) (1995), 197-227.

Appendix: Definitions

In the last years, various planning frameworks were developed, which use manifold terminologies to describe the different elements of the planning process. In the following we will try to clarify various terms, which are most frequently used in generative planners, plan executors, etc. Similar categories have been defined by [58] within the planning initiative "Shared Planning and Activity Representation" (SPAR, [72]). They have clustered these elements into conceptual groups (in alphabetical order). We have adapted their groups and definitions and added synonyms used in various planning frameworks.

- **Activities:** The representation of activities (actions, events, operations, etc.) is the heart of plan representations. Activity specifications define the list of steps performed within a plan. Typical activity requirements usually involve some mechanism for abstraction and level decomposition as well. In the terminology of generative planners an operator is often called an activity in contrast to the terminology of plan execution systems, an action is called a plan fragment or operating system. An action or operator is an activity, which is performed by an agent. An event is an activity, which happens without any influence of an agent.
  - ⇒ *Representational Elements:* activity arguments and variables, simple and complex sequences (orderings, etc.), hierarchical decomposition of activities, parallel activities, alternative activities, activities with context-dependent effects, activities with indirect effects, pre-conditions, post-conditions (add and delete lists), effects (post-conditions), etc.
  - ⇒ *Synonyms:* action, act, event, episode, plan (set of actions), plan fragment, process, operating procedure, operator

- **Agents:** The term "agent" in a plan representation generally refers to the people and systems assuming various roles throughout a plan life-cycle. Agent models are necessary to clarify how the various agents interrelate.
  - ⇒ *Representational Elements:* agent model, agent roles, agent purposes, etc.
  - ⇒ *Synonyms:* users

- **Control Structures (used for Execution or Simulation)**: A detailed control structure is required for plan execution or simulation. This control structure may include constructs such as loops, conditional activities, and world state monitors or queries.
  - ⇒ *Representational Elements:* loops, cycles, conditionals, etc.
  - ⇒ *Synonyms:* loops, monitors

- **Domain Knowledge:** Plans are developed with respect to knowledge of a particular domain.
  - ⇒ *Representational Elements:* domain assumptions, domain model, etc.
  - ⇒ *Synonyms:* knowledge sources, knowledge roles

- **Evaluations:** A number of plans may satisfy the requirements for a specific set of objectives. It is often the case that these plans need to be evaluated in certain ways to determine if they are acceptable for use (e.g. plan selection, plan execution). Various domains require particular sets of criteria that can be applied for these evaluations.
  - ⇒ *Representational Elements:* evaluation criteria, etc.
  - ⇒ *Synonyms:* heuristics, critiquing

- **General Structures:** General representational structures such as lists, sets, numbers, symbols, sentences, etc. are necessary to express plan knowledge.
  - ⇒ *Representational Elements:* terms and expressions, symbols, numbers, measurements/amounts (e.g., size, weight, time, cost), etc.
  - ⇒ *Synonyms:* data-types, primitive-types, constraints, units

- **Goals:** A goal is an intention of the plan designer, which should be fulfilled during or after executing a plan. It is needed to express tasks and to connect plans to their intended purposes. The state of a goal (e.g., satisfied, unsatisfied), the type of goal (e.g., achievement, maintenance, etc.), as well as expressed task constraints are used throughout the planning and execution process to provide purposeful behavior.
  - ⇒ *Representational Elements:* goals of achievement, goal of maintenance, goal of avoidance, goals ordering, conditional goals, etc.
  - ⇒ *Synonyms:* requirements, objectives, mission, tasks, cues, purposes, intentions, post-conditions, effects

- **Organizational:** Plans constructed within an organizational environment require representation of specific organizational (site-specific) concepts. This may include models of authority (permission to work with various parts of a plan), deadlines, milestones, policies, products, etc.
  - ⇒ *Representational Elements:* Who, What, Where, When, Why, How , etc.
  - ⇒ *Synonyms:* preferences, policies, authorities

- **Plans:** A plan may comprise cascades of sub-plans (a plan hierarchy), which are finally decomposed into series of atomic actions. A number of general requirements may also be expressed for the overall structure of plans or schedules. This may include the ability to represent various abstraction levels, sub-plans, decompositions, etc. Shared plan structures may also require a mechanism (or set of mechanisms) to extend a representation for a particular domain or use.
  - ⇒ *Representational Elements:* schedule/ordering/sequencing for planned activities, etc.
  - ⇒ *Synonyms:* set of actions/activities, plots, schema (scheme, schemata), operators, plan fragments, operating procedures

- **Rationales:** A rich plan representation may be required, which not only contains the solution object (i.e., the actions, and orderings) but also a trace of how it had arrived at this result. This is analogous to the way that the proof steps of a theorem are required products of a proof. This may list items like the decisions taken, the alternatives considered, and the criteria used to evaluate the alternatives.
  - ⇒ *Representational Elements:* explanation information, historical trace of plan generation, etc.
  - ⇒ *Synonyms:* explanations, history

- **Resources/Objects:** Plans can create, utilize, manipulate, and destroy objects. These objects may play the role of resources that are used to complete a task. Resources typically need to be categorized whether they can and cannot be utilized (i.e., consumable, reusable, prohibited, etc.). Detailed models of resources/objects and how they interact with activities may be required as well.
  - ⇒ *Representational Elements:* resource capability/characteristics, resource/object roles, eligible resources, implicit / explicit resource association, etc.
  - ⇒ *Synonyms:* preferences, site-specific characteristics

- **States:** State representation is a very important part of plan representation. Plan generators must be able to create numerous hypothetical states that correspond to projected results of plan steps. Plan executors must be able to maintain a state representation that closely matches their current environment. States may represent the total set of concepts known at a particular point in time or may represent a difference between two such states.
  - ⇒ *Representational Elements:* state representation, changes in world state, etc.
  - ⇒ *Synonyms:* situations, predicates, clauses, relations (pairs of variables and values)

- **Time:** Plans are usually constrained by time. Temporal constraints require plan steps to be ordered and to constrain the search space. This ordering may involve relative qualitative or quantitative temporal relationships (e.g. A must occur before B, A must occur 10 hours before B), as well as various constraints on a single activity, such as activity durations and uncertainty. Plan steps may also be required to be connected to temporal constraints, which may be related to a reference point or an activity as well as to an absolute time annotations. (e.g. do A 10 minutes after breakfast, do A 10 minutes after finishing B, A on March 28th at 11:00am).
  - ⇒ *Representational Elements:* time interval, time points, temporal constraints, activity duration, begin/end of an activity, qualitative/quantitative relations (e.g., [2]), metric, calendar, etc.
  - ⇒ *Synonyms:* durations, time windows

- **Uncertainty/Ambiguity:** In real-world scenarios, one is unable to deterministically specify the effects of an action or to accurately hypothesize a future world state. In cases such as this, provisions are often made for ways to represent parts of the plan that are uncertain or ambiguous. This may include ranges on values (e.g. time windows), abstract plan steps, or probabilistic estimations.
  ⇒ *Representational Elements:* minimum/maximum information for time windows and resource levels , etc.
  ⇒ Synonyms: certainty factor, probabilistic factor