Monitoring, Repair and Replanning Techniques to support Exception Handling in HTN-based Therapy Planning Systems*

Inmaculada Sánchez-Garzón[1], Juan Fdez-Olivares[1] and Luis Castillo[2]

[1] Department of Computer Science and A.I., University of Granada, Spain
{isanchez,faro}@decsai.ugr.es
[2] IActive Intelligent Solutions, Spain. L.Castillo@iactive.es

Abstract. This work presents, on the one hand, a monitoring component for detecting exceptions during the execution of a treatment plan, which is automatically generated by HTN planning techniques, in a real clinical context. On the other hand, a dynamic strategy to tackle such exceptions and to make the treatment plan valid to the evolving clinical context is proposed. The strategy is composed by different adaptive procedures (based on knowledge-based rules, alternative decomposition methods and mixed-initiative approaches) that are selected according to the nature of the detected exception, after an error diagnosing step.

1 Motivation

One of the first aims of therapy planning systems is to support the effort of healthcare professionals when they deal with the problem of designing a suitable treatment plan for a given patient. Moreover, such patient-tailored plans must be created following the decisions and procedures specified in an evidence-based, clinical protocol. Regarding this topic, Hierarchical Task Network (HTN, [1]) planning systems have proven to be successful on several real therapy planning applications. The reasons are, on the one hand, that this planning paradigm supports the representation of clinical protocols as planning domains described by hierarchies of tasks networks. On the other hand, it supports the efficient generation of patient-tailored treatment plans, which can also be seen as personalized workflows. A clear example of the effectiveness of HTN-based therapy planning approaches is *OncoTheraper* ([2], [3]), the research project in which this work is framed. This Clinical Decision Support System (CDSS) allows to generate long-term, personalized treatment plans focused on the pediatrics oncology area. In addition, the group of healthcare professionals conforming this research team has validated such automatically-generated plans and has recognized their adequacy for being directly carried out in the real world. Hence the next goal to achieve in this context is to apply these personalized workflows to *real* patients in a *real* clinical environment. Since clinical world is highly dynamic and exceptions commonly occur during the execution of a treatment plan, we consider necessary

*This work has been partially supported by the Andalusian Regional Ministry of Innovation under project P08-TIC-3572 (OncoTheraper).

to extend the currently developed HTN approaches for treatment generation with new techniques aimed at covering the life-cycle of treatment plans that are executed in the changing healthcare environment [4]. On the one hand, it might be useful to incorporate a component in charge of monitoring the progression of the patient, keeping the execution trace of treatment plans and facilitating the detection of exceptions during their execution. On the other hand, flexible adaptation capabilities for tackling such exceptions and making plans valid to the evolving clinical context are also required. A major reason is that long-term therapy plans must be frequently adapted according to the health conditions and the real progress of the patient. In addition, HTN-based approaches lack this functionality since they construct plans supposing that everything is going to happen as planned since the world is *static* and *deterministic*.

The inclusion of these components might improve the adoption, diffusion and effectiveness of current therapy planning systems by covering the topic of clinical exception management, which is a crucial part of the life-cycle of treatment plans in clinical domains. For this reason, in this paper, we propose a strategy (focused on HTN planning frameworks) that is based on the approach known as *plan execution monitoring and replanning* [5]. In general terms, the idea of this strategy is to generate the therapy plan in a deterministic form, i.e., without modeling all possible execution exceptions (as conditional branches) on the planning domain. After this generation episode (which is already achieved in OncoTheraper [3]), the treatment plan is executed in the real world and healthcare professionals interact with an event-based monitoring component in order to follow the execution trace of the plan and manage the execution of its actions. When an unanticipated event is detected, and, after an exception analyzing step for diagnosing its cause, the system will invoke a repair and replanning (R&R) component for solving it. This component is based on a strategy composed by different adaptive procedures (based on knowledge-based rules, alternative decomposition methods and mixed-initiative approaches) that try to make the plan valid to the evolving context for carrying on with its execution.

In the following section, we will describe the functionality of the monitoring component developed. Then, we will present the R&R strategy that, for the time being, is in its early stage of development. After that, we will comment some aspects related to the experimental evaluation. Finally, we will review related work and will conclude with some expected results and future work.

2 A Monitoring Component for Detecting Exceptions

In order to support the execution and supervision activities, treatment plans must be composed by a set of information entities that allows to know both the way in which their actions have to be executed and the properties of the clinical context that require to be monitored. Regarding this topic, treatment plans automatically generated by OncoTheraper contain a collection of partially-ordered

[3] A demonstrator of OncoTheraper has been developed and can be executed through a web application in http://tratamientos.iactive.es/

actions (with temporal annotations) representing organizational, drug administration and evaluation activities to be accomplished on a specific patient (see figure 1). Every action is defined (apart from its description and formal parameters) by a set of knowledge entities that makes our plan representation rich and complete from the execution and monitoring point of view. These entities are:

a) A list of *preconditions* specifying the properties that must be met in a clinical context before executing an action (e.g., the nurse allocated to an administration activity must be available).

b) A list of *effects* expressing the changes produced in an evolving clinical context after the correct execution of an action (e.g., after an administration action, the total quantity of a specific drug that has been applied to the patient is incremented with the new dosage administered).

c) A set of *metadata* allowing to represent additional knowledge required by the user or the monitoring component, as the mode of administration of a drug.

d) Since actions may be *manual* or *automatic*, there is another entity that specifies the *actor* (a medical staff or the system) who is in charge of controlling the execution of the action.

e) *Temporal information* specifying the duration and the estimated dates at which each action is allowed to be initiated and finished. Indeed, such dates are represented as flexible time intervals with the *earliest* and *latest* start and end dates since the treatment plan is deployed (in planning time) over a *temporal constraint network* ([3], [6]).

f) A list of *order constraints* and a list of *causal links*, generated by the reasoning process of the planner, that define dependencies with preceding actions of the plan. An analysis of these structures allows to know, among other things, the correct sequence in which actions must be executed, thus allowing to establish both sequential and parallel execution flows.

Id	Action	Start Date	End Date	Dur (hrs)	Medical Staff	Drug	Mode	Dosage (mmg)
30	PreviousEval	16/02/2011	17/02/2011	24	John	-	-	-
31	AdminDrug	01/03/2011	02/03/2011	24	Mary	VCR	IV	0.866
32	AdminDrug	08/03/2011	09/03/2011	24	Mary	VCR	IV	0.866
33	AdminDrug	01/03/2011	16/03/2011	360	Mary	PRD	O3D	23.094
34	AdminDrug	01/03/2011	16/03/2011	360	Mary	PRC	O3D	57.735
35	AdminDrug	01/03/2011	02/03/2011	24	Mary	CFM	IVP	288.675
36	AdminDrug	08/03/2011	09/03/2011	24	Mary	CFM	IVP	288.675
37	RemissionEval	16/03/2011	17/03/2011	24	John	-	-	-

- **Description**: Administer CFM to Peter
- **Parameters**: Dosage = 288.675
  Patient = Peter / Drug = CFM/ Dur = 24
- **Metadata**: type= Manual / mode = IVP
- **Actor** = Mary (nurse)
- **Start** =[08/03/2011, 08/06/2011]
- **End**= [09/03/2011, 09/06/2011]
- **Preconditions**: Mary.available = true
- **Effects**: total_dosage_CFM += 288.675
- **Order dependency** with action 35

Fig. 1: A simplified plan following Hodgkin's Protocol and the entities of action 36

Taking all these previous entities into account, a monitoring component based on user events has been developed. The reason of this operation mode is that treatment plans contain manual actions that require human intervention to be initiated and finished. Moreover, plan monitoring in clinical contexts is an interactive activity often requiring the collaboration of different members of a clinical team. Apart from events management, the monitoring component developed (see figure 2.a) is in charge of keeping the execution trace of treatment plans. To allow this activity, an **execution model** of actions has been defined. This model is based on a state transition system, so each action of the plan has an

attribute defining its execution state (*initial, ready, started, finished, suspended, canceled* or *aborted*). Firstly, an action is in *initial* state and it changes to *ready* when all its preceding actions (known by an analysis of the causal and order dependencies) have finished their execution and the current date is consistent with the estimated start date of the action. After that, the action can initiate its execution by the monitoring component (if it is an *automatic* action) or by a request of the user (if its type is *manual*). The action changes to *started* state only if all its preconditions are satisfied in the current evolving clinical context considered for the monitoring process (see figure 2.b). This context represents the properties that require to be monitored, i.e., it is a simplified representation of the clinical data of the patient, who is undergoing the treatment plan, and the resources (both humans and materials), which are involved in its execution. In addition, clinicians may suspend, cancel or abort the execution of the action, thus changing its state to *suspended*, *canceled* or *aborted*, respectively. Finally, the action can finish its execution when the current date is consistent with its estimated end date and, in the case of manual actions, the user has sent the appropriate event. The finalization function (see figure 2.c) is based on applying the effects of the action to the evolving clinical context and activating the immediate succeeding actions of the plan that can also initiate its execution.

MONITORING ALGORITHM a)

```
Input: plan ← Treatment Plan;
       CCC ← Current Clinical Context;
while plan.hasPendingActions() do
    /* User Events Management */
    foreach e:UserEvent, a:Action, b:Actor do
        switch e.type do
            case start: startAction(a,b);
            case finish: finishAction(a,b);
            case suspend: suspendAction(a,b);
            case cancel: cancelAction(a,b);
            case abort: abortAction(a,b);
    /* Check State Transitions of the Actions */
    foreach a ∈ plan.actions such ((a.state = initial) and
    (forAll p ∈ a.precedingActions, p.state = finished) and
    (currentDate ≥ a.startDate)) do
        a.state ← ready;
    /* Start and End of Automatic Actions */
    foreach a ∈ plan.actions such ((a.type = automatic) and
    (a.state = ready)) do
        startAction(a, SYSTEM);
    foreach a ∈ plan.actions such ((a.type = automatic) and
    (a.state = started) and (currentDate ≥ a.EndDate)) do
        finishAction(a, SYSTEM);
    /* Analysis of Causal Structure */
    foreach link ∈ plan.causalLinks(currentDate) do
        if link is not satisfied in CCC then
            action ← link.getActionTo;
            action.state ← aborted;
            call ExceptionAnalysisM(action, link);
```

START ACTION b)

```
Input: a ← action; b ← actor;
Output: Success or Fail
if ((a.state=ready) and
(a.executableBy(b))) then
    foreach p ∈ a.preconditions do
        if (not p in CCC) then
            a.state ← aborted;
            call ExceptionAnalysisM(a,p);
    a.state ← started;
    return Success;
```

FINISH ACTION c)

```
Input: a ← action; b ← actor;
Output: Success or Fail
if ((a.state=started) and
(a.executableBy(b)) and
(currentDate ≥ a.End)) then
    foreach e ∈ a.effects do
        if (not applicable e in CCC)
        then
            a.state ← aborted;
            call ExceptionAnalysisM(a,e);
    a.state ← finished;
    a.activateSuccessorActions;
    return Success;
```

Fig. 2: Pseudocode of the Monitoring Algorithm

Another crucial function of the monitoring component is to facilitate the identification of exceptions during the execution of a treatment plan. An exception occurs when something is not going as planned and, after its detection, the state of the set of affected actions (i.e., *impact* of the exception) changes to *aborted* until the error is handled by the repair and replanning (R&R) module. Regarding this function, the monitoring component can identify some exceptions

autonomously, such as the failures detected either during the application of the effects of an action or during the checking step of its preconditions. In addition, this component also checks the causal structure of the plan in order to detect exceptional situations as soon as possible, thus allowing to have additional time to suggest a valid response. Finally, due to the highly dynamic nature of the clinical environment and the not completeness of the domain models [7], there are other exceptions that will be identified by clinicians at runtime instead of the monitoring component autonomously. In these cases, such experts must report on the failure to the system and offer it some feedback information to know the cause. After the detection of a failure, an episode of exception analysis is carried out in order to identify the failed activity, the cause of such error and the list of actions affected by it. Then, the R&R component is invoked.

3 A Dynamic Repair and Replanning Strategy

Taking the characteristics of healthcare environments into account [4], the proposed strategy should fulfill the following features. In the first place, it should support some level of interaction with healthcare professionals to take advantage from their expertise and intuition. On the other hand, since most of the unexpected events are related to changes in *temporal data* (defining some variable parameters of the patient, as his weight or his height) and these exceptions (not being predicted in advance) are commonly solved with local modifications in the plan, the strategy should be flexible enough for handling both simple and more complex failures. Moreover, it should minimize the changes induced in the treatment plan, thus promoting the *plan stability*. The reason of this conservative attitude is that, when the execution of a treatment plan fails, many of its parts remain unchanged, others have already been executed and another parts may require resources (humans or materials) that have already been committed, thus redirecting a new plan may be costly. In addition, this stability proposition might reduce the cognitive load of healthcare professionals since the suggested plan should be close to the original one that they previously validated. Finally, the strategy should exploit the reasoning and decision-making process performed by the planner in the construction of the original plan (avoiding as far as possible the replanning from scratch). The reason is that planning domains (representing clinical protocols) and treatment plans (representing personalized workflows) are relatively complex (a lot of knowledge entities and restrictions), so a replanning from scratch is not recommended to repair a simple and single error.

All these requirements have motivated both the design and the structure of the repair and replanning (R&R) strategy that, as a result, is organized into different levels (see figure 3). The first level is aimed at applying knowledge-based repair rules to adapt the treatment plan with local changes. The objective of the second level (which produces more global changes in the plan) is to initiate a *replanning* episode for generating a new sub-plan according to the evolving clinical context. Finally, a mixed-initiative approach is also considered (as last alternative) in the third level. Other aspect to comment is that the degree of autonomy of the proposal and the degree of user involvement depend on the

criticality of the exception detected, i.e., the resolution of errors identified by clinicians at runtime (which commonly are more difficult to solve) requires more expert participation since they are not explicitly modeled in the planning domain and the monitoring component is unaware of how to solve them autonomously. In the next sections we explain the functionality of each level of the R&R strategy.

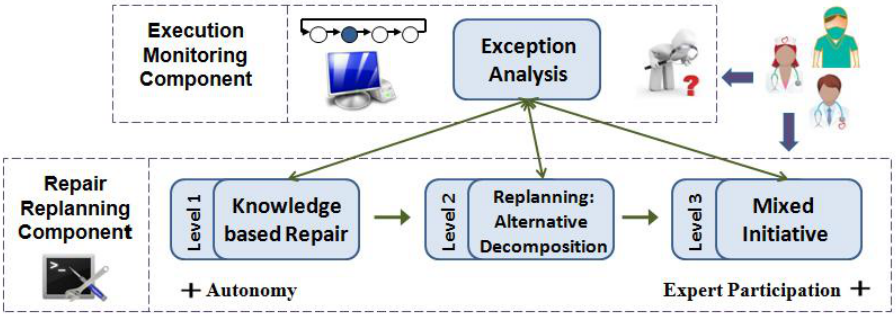

Fig. 3: Overview of the Repair-Replanning Strategy

3.1 Level 1. Knowledge-based Repair

The objective of this first level is to provide a collection of pre-defined repair rules that tries to automatically solve *expected* exceptions with local changes in the plan. The reason is that clinical protocols (concretely the oncology ones) often specify the steps to perform in some well known exceptional situations and this information can be used to create such collection of knowledge-based rules, which are represented as ECA rules [8]. An ECA rule is defined by three components: an *event* (the signal that triggers the invocation of the rule), a *condition* (the logical test that, if it is evaluated to true, causes the repair action to be carried out) and a *repair action* (the set of steps to adapt the treatment plan). For example, in an oncology protocol we might define a rule in charge of adjusting the dosage of the next drug administration actions of a plan when, after an updating process of the clinical data of the patient, the monitoring component detects that his body mass index has changed (see figure 4.a). Although an effective R&R mechanism should be a domain-independent strategy, we are aware that in the clinical context the type of repair to apply depends on the considered application domain, on the detected exception and even on the current health conditions of the patient. For this reason, the idea here is to provide a list of predefined repair actions that will be customized by the knowledge workers, during the protocol representation process, according to the application domain considered.

Other example of repair rules is a *temporal delaying rule*, that postpones the pending actions of a treatment plan until a desirable state is satisfied (e.g., to delay the plan until the fever that suffers the patient has remitted). This delay is made by a *constraint propagation* process over the *temporal constraint network* also stored in the plan [6]. Another rules are a *canceling rule*, to remove a specific action of the plan (e.g., a high toxicity level in the patient forces the oncologist to delete an administration action) or an *adding rule*, to incorporate a new action in

the treatment plan (e.g., in order to protect the patient against infections after a bad reaction of a drug, the oncologist adds an action to administer antibiotics). Repair rules related to *reorder* the actions of a treatment plan or *replace* some parameters (e.g., change this drug, for this one) or resources (e.g., replace this not available nurse with this suitable candidate) of them are also considered.

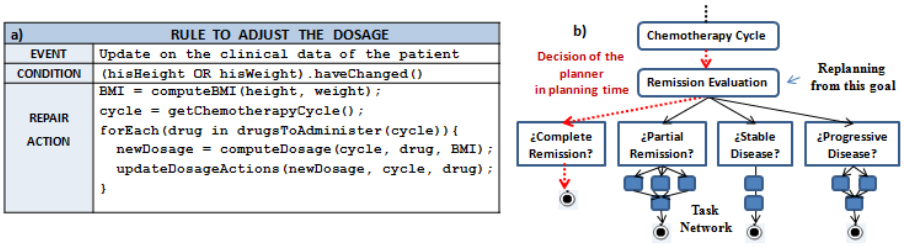

Fig. 4: Example of a Repair Rule and an extract of a Decision Graph

3.2 Level 2. Replanning: Alternative Decomposition

The previous level tries to make local changes for solving the failed treatment plan. However, if this proposal is not effective enough for complex exceptions, the solution here is to initiate a replanning episode for replacing the damaged actions. The main motivation is that, during the plan generation process, the planner has to assume what the health status of the patient will be after applying a set of chemotherapy cycles. Since, in planning time, there is not enough information about the future response of the patient to the treatment, the planner generates the complete, long-term plan following an optimistic attitude, i.e., assuming that the tumour will have a complete remission. Unfortunately, this philosophy is not entirely realistic and, for this reason, the objective of this module is to invoke the planner in order to generate a valid subplan when the patient has not progressed as planned and when more information about his health conditions and progression is provided. Regarding the HTN paradigm [1], the idea is to apply an alternative method to decompose the high level goal from which the plan is not valid (see figure 4.b). A decomposition method specifies the set of actions (task network) to perform in order to achieve a goal (e.g., Remission Evaluation) under certain conditions (e.g., the tumour has a partial instead of a complete remission). In addition, in order to avoid the replanning from scratch, the previous decision-making process carried out by the planner for the construction of the original plan is recorded in an additional structure called **decision graph** [9]. Among such decisions, we mention the list of alternatives (other decomposition methods) for reducing a goal and some information related to the internal state of the planner, which are useful items for initiating the replanning episode.

Finally, if the exception analyzing step considers that the unexpected event can not be solved by none of the levels seen so far or, conversely, none of them has suggested a valid repair, the solution here is to invoke the mixed-initiative approach of the third level of the R&R strategy.

3.3 Level 3. Mixed-Initiative Approach

The objective of this third level is to create a framework in which healthcare professionals can conveniently interact to the R&R component in order to guide the plan adaptation process. The reason is that not all medical exceptions can be predicted and known in advance and, in this case, the solution is to take advantage from the intuition and experience of clinicians in previous and similar episodes. For this reason, this interactive framework should allow them to adjust, organize, delete or add activities in the treatment plan in order to make it valid to the current evolving context. On the one hand, clinicians might add some actions to the failed plan in order to achieve a state, where a set of conditions are met, and from which the planner can be invoked as in the second level. On the other hand, the new added actions might constitute either a new repair rule or a new conditional branch of the clinical protocol, thus incrementally improving the representation quality of the protocol. Moreover, clinicians might add both actions that are pre-defined in the planning domain or completely new actions (specifying, in this case, their parameters, preconditions and effects).

This module requires a more exhaustive study in order to also include verification mechanisms for guaranteeing, somehow, the correctness and consistency of the repaired plan suggested by experts. Once explained the R&R strategy, we are going to comment the experimental evaluation. Then, we will review related work and we will conclude with some expected results and future work.

4 Experimental Evaluation

OncoTheraper [2] is constituted by a group of oncologists that periodically evaluates the achievements of the development team. Such evaluations are based on proofs of concept focused on specific parts of its technology, as the process for generating personalized plans[4], the process related to the interactive execution of such plans by clinicians[5] and the process for detecting exceptions by the monitoring component. For this last purpose, we have designed a simulator of exogenous events and we have checked that the monitoring module is able to detect errors autonomously. This previous experimentation demonstrates that the future proofs of concept that will be designed to evaluate the R&R strategy will be supported by a robust and proven technology. These proofs are focused on evaluating several aspects of the R&R strategy. On the one hand, in order to evaluate the proposal in terms of efficiency, we will analyze the consumed time for adapting a plan with a repair rule instead of a replanning (from scratch) episode. Moreover, we will measure the overhead added to the planning process by constructing the decision graph and we will compare any gains in problem-solving performance of adapting plans by a replanning from scratch and a replanning

[4] http://tratamientos.iactive.es/ is a web application to generate personalized plans to treat de Hodgkin's disease according to the clinical initial data of a patient.

[5] In order to have a first prototype of tool for the interactive execution of plans, we have integrated the output of our planner with the input of a standard BPM (Business Process Management) runtime engine through a model-to-model transformation [9].

using the decision graph. On the other hand, in order to know both the practicability and the usefulness of the proposal, it would be interesting to know how many percent of the treatment plans need repair on each level of the strategy and how do oncologists rate the solutions found by the system. For this last purpose, we will consider a real clinical context and several proofs are being carried out about the integration of *OncoTheraper* with external information systems[6].

5 Related Work

The idea of adapting a failed therapy plan has been discussed from different perspectives, as in [10], which presents a generic approach for handling unexpected events in runtime. A limitation of this work is that clinicians can not dispose of the complete treatment plan that will be applied to a patient because the list of goals to achieve are *refined* during plan execution. The reason is that, the sections of the treatment that require more information about the health condition of the patient are kept as goals and are refined later. This fact might make difficult to estimate, in advance, both the set of actions and resources that will be required in the future and it might even complicate their later commitment.

On the one hand, although the development of Computer-Interpretable Guidelines (CIG, [11]) languages has provided formalisms for the representation of clinical protocols, a corresponding effort in the knowledge reasoning side is needed in order to generate personalized treatment plans taking into account both the clinical data of the patient and the temporal and resource constraints of the care process. However, HTN planning paradigm is capable of managing and efficiently reasoning about these requirements. On the other hand, regarding the properly HTN-based paradigm, we mention several works. In [12] and [13], the repair strategies rely on additional structures as our *decision graph*. However, we consider that these proposals do not offer a flexible response for tackling exceptions in the healthcare environment since they try to initiate a replanning episode in any case (both for simple and complex errors). In addition, if there is not any appropriate decomposition method to begin the replanning episode, the strategy will not offer any solution. Another related area is that of workflow systems, where issues concerned to exception handling have been investigated. Furthermore, most of the existing works in this area are focused on rule-based approaches, such as [14] and [15]. However, we consider that this alternative might not be effective enough for exceptions requiring more global changes in the plan since the adaptation process is only based on applying pre-defined rules.

6 Conclusions and Future Work

This work proposes new techniques (focused on HTN-based therapy planning systems) for tackling exceptions during the execution of a treatment plan in

[6] A demonstrator in the Health Living Lab of Andalusia has been developed: http://www.livinglabsalud.es/projects/oncotheraper/.

a real clinical environment. On the one hand, a monitoring component is presented in order to keep the execution trace of the treatment plan, facilitate the supervision of the state of the patient and allow both to detect exceptions and analyze their nature. On the other hand, a dynamic repair and replanning (R&R) strategy, which is organized in three different levels, is presented to make the treatment plan valid to the evolving clinical context. Although the proposal is in an early stage of development, we consider that it has several advantages. On the one hand, flexibility is a *sine-qua-non* condition of every exception handling approach and we consider that the proposal here presented encourages this principle by covering both local and more global changes, with different degrees of user participation and an enough coverage for solving a wide variety of errors. For this reason, the R&R strategy might be seen as a contribution from the medical standpoint since its incorporation into the current therapy planning systems might improve their effectiveness, adoption and diffusion. On the other hand, this proposal is also innovative in the area of HTN-based planning as it is a flexible and *quasi*-generic approach that can be applied in fields that are different from the clinical domains. Moreover, aspects related to *efficiency* (reuse the previous reasoning process of the planner), *stability* (minimize the changes induced in the plan, thus reducing the cognitive load of clinicians) and *interactivity* with experts, are also considered in the development of the strategy.

To conclude this work say that our final aim is to create a framework that integrates the required components for covering the complete life-cycle of treatment plan executed in a real clinical environment.

References

[1] E.D. Sacerdoti. The nonlinear nature of plans. In *Proc. of IJCAI*, pages 206–214, 1975.

[2] J. Fdez-Olivares, J. A. Cózar, and L. Castillo. OncoTheraper: Clinical decision support for oncology therapy planning based on temporal hierarchical tasks networks. In *Knowledge Management for Health Care Procedures. LNCS*, volume 5626, pages 25–41, 2009.

[3] J. Fdez-Olivares, L. Castillo, J. A. Cózar, and O. García-Pérez. Supporting clinical processes and decisions by hierarchical planning and scheduling. *Computational Intelligence*, 27:103–122, 2011.

[4] S. Miksch. Plan management in the medical domain. *AI Communications*, 12:209–235, 1999.

[5] S. Russell and P. Norvig. *Artificial Intelligence: A Modern-Approach*. Prentice Hall, 2009.

[6] L. Castillo, J. Fdez-Olivares, O. García-Pérez, and F. Palao. Efficiently handling temporal knowledge in an HTN planner. In *Proc. of ICAPS06*, pages 63–72, 2006.

[7] Kambhampati. S. Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models. In *National Conf. on Artificial Intelligence*, 2007.

[8] N.W. Paton. *Active Rules in Database Systems*. Springer, 1999.

[9] J. Fdez-Olivares, I. Sánchez-Garzón, A. González-Ferrer, J. A. Cózar, A. Fdez-Teijeiro, M. R. Cabello, and L. Castillo. Task network based modeling, dynamic generation and adaptative execution of patient-tailored treatment plans based on Smart Process Management technologies. In *Proc. of Workshop on Knowledge Representation for HealthCare, AIME*, 2011.

[10] A. Grando, M. Peleg, and D. Glasspool. A goal-oriented framework for specifying clinical guidelines and handling medical errors. *Biomedical Informatics*, 43:287–299, 2010.

[11] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. A. Greenes, R. Hall, and et al. Comparing computer-interpretableguideline models: A case-study approach. *American Medical Informatics Association*, 10(1):52–68, 2003.

[12] I. Warfield, C. Hogg, S. Lee-Urban, and H. Muñoz Avila. Adaptation of hierarchical task network plans. In *Proc. of FLAIRS*, pages 429–434, 2007.

[13] N.F. Ayan, U. Kuter, F. Yaman, and R.P. Goldman. HOTRiDE: Hierarchical ordered task replanning in dynamic enviroments. In *Proc. of ICAPS*, 2007.

[14] U. Greiner, R. Müller, E. Rahm, J. Ramsch, B. Heller, and M. Loeffler. AdaptFlow: Protocol-based medical treatment using adaptative workflows. *Methods of Information in Medicine*, 44:80–88, 2005.

[15] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa. Flexible guideline-based patient careflow systems. *Artificial Intelligence in Medicine*, 22:65–80, 2001.