Task Network based modeling, dynamic generation and adaptive execution of patient-tailored treatment plans based on Smart Process Management technologies

Juan Fdez-Olivares[1], Inmaculada Sánchez-Garzón[1], Arturo González-Ferrer[1], Juan A. Cózar[2], Ana Fdez-Teijeiro[3], Manuel R. Cabello[4], and Luis Castillo[4]

[1] Department of Computer Science and A.I, University of Granada, Spain,email: faro@decsai.ugr.es
[2] Pediatrics Oncology Service, Hospital Complex of Jaén, Spain,email: jcozarolmo@hotmail.com
[3] Pediatrics Oncology Service, Hospital Virgen Macarena, Spain, email:anateijeiro@hotmail.com
[4] IActive Intelligent Solutions, Spain, email:L.Castillo@iactive.es

Abstract. In this paper we present a knowledge-based, Clinical Decision Support System (OncoTheraper2.0) that provides support to the full life-cycle of both clinical decisions and clinical processes execution in the field of pediatric oncology treatments. The system builds on a previous proof of concept devoted to demonstrate that Hierarchical Planning and Scheduling is an enabling technology to support clinical decisions. The present work describes new issues about the engineering process carried out in the development and deployment of the system in a hospital environment (supported by a knowledge engineering suite named IActive Knowledge Studio, devoted to the development of intelligent systems based on Smart Process Management technologies). New techniques that support the execution and monitoring of patient-tailored treatment plans, as well as, the adaptive response to exceptions during execution are described.

1 MOTIVATION

The development of Knowledge Based Clinical Decision Support systems (CDSS) is a process that requires the integration of several techniques, methodologies and tools. The key issue to be addressed is the capability of representing evidence-based clinical knowledge, but it is also necessary to appropriately capture clinicians requirements (considering the decision life-cycle followed in their daily work). In addition, it is necessary to consider that clinicians are carrying out their work in an environment where Electronic Health Records (EHR) and Clinical Management Systems(CMS) already exist, the last ones mostly based on workflow technologies (also known as Business Process Management, BPM, technologies) that support administrative and organizational aspects of clinical work. Therefore, the integration of clinical data in the decision making process supported by the CDDS as well as the integration of the output provided by the CDSS into the CMS, must also be addressed. The consideration of these issues in a Knowledge Based System development process will lead to a CDSS which not only will appropriately work, but will be considered helpful for the clinicians, the main value of any Knowledge Based Clinical Decision Support project.

In previous work [1][2] authors presented a proof of concept devoted to demonstrate that Hierarchical Planning and Scheduling [3, 4] is an enabling technology to develop Knowledge Based CDSS since it is capable of providing support to clinical decisions and processes in medical treatments (concretely in the field of pediatrics oncology). Authors presented a temporal Hierarchical Task Network (HTN) representation, based on a textual language named HTN-PDDL (initially introduced in [3]), expressive enough to represent clinical processes as well as complex, periodic temporal constraints that appear in oncology protocols. Moreover, the textual representation was used to generate patient-tailored treatment plans, based on an hierarchical, knowledge-driven planning process the usefulness of which has been recognized by oncologists.

The present work describes new issues concerned with both considerations: the engineering process carried out in the development and deployment of OncoTheraper2.0, build on the previous proof of concept, in a real hospitalary ecosystem (being at the time of writing this paper an ongoing process), and the clinical decision making life-cycle supported by OncoTheraper2.0. With respect to the first consideration, and from the point of view of a knowledge engineer, in this work we introduce a suite of highly-coupled software tools, named *IActive Knowledge Studio*, that has been used to develop OncoTheraper2.0. It has been developed by *IActive Intelligent Solutions*[5] on the basis of the techniques developed by our academic research group. The suite is oriented to the development of intelligent systems based on *Smart Process Management* technologies [5]. These technologies are founded on Artificial Intelligence Planning and Scheduling techniques and they support the modeling of task-based knowledge as well as the dynamic generation and execution of timed sequences of actions based on such knowledge. On the other hand, and from the point of view of clinical knowledge workers, we will explain how OncoTheraper 2.0 provides support to the whole life-cycle of clinical decisions. Indeed, the system not only allows the formal representation of clinical protocols, but also the automated generation of patient-tailored treatment plans, by integrating patient clinical data from external EHR. In addition, the system provides support for the interactive execution of such plans (based on standard BPM technologies) and incorporates techniques to both, monitor the state of a patient during the enactment of a treatment, and dynamically and adaptively respond to exceptions arisen at execution time.

The remainder of this paper is structured as follows. The next section describes the general decision making life-cycle followed by oncologists in the management of treatment plans, and summarizes the main functionalities of OncoTheraper2.0, explaining them with an example. Then, the main issues concerned with the development and integration of the CDSS using IActive Knowledge Studio are briefly described. Finally, the paper is focused on the techniques that support the execution of treatment plans based on standard BPM tools, as well as the monitoring of plans and the dynamic response to exceptions arisen during the interactive execution of such plans.

[5] A spin-off company based on the research results of our group http://www.iactive.es

2 ONCOTHERAPER 2.0 IN A NUTSHELL

In the case of oncology treatments, the clinical decision life-cycle carried out by clinical staff is as follows: once the diagnostic step is carried out, oncologists design a patient-tailored, long-term treatment plan with a time-horizon of months. This planning step is carried out by considering the patient profile and following the decisions and procedures specified in an evidence-based, oncology clinical protocol. The treatment plan is composed by pre-programmed actions which, in turn, are used by clinical staff as a way to organize their work for the next weeks or months. The clinical actions include temporal information specifying the time at which actions have to be performed including: evaluation sessions, chemotherapy cycles (organized in several drug-administration actions), radiotherapy cycles, or even administrative actions (like analysis requests, filling a form, sending a prescription, etc.). Secondly, the plan is enacted, and through the pre-programmed evaluation and follow-up sessions the state of the patient is monitored by doctors. Thirdly, the plan may be modified either by changes in the progress of the state of the patient or directly by clinicians preferences, what may raise a plan repairing process or even a replanning step that may trigger new clinical decision-making process.

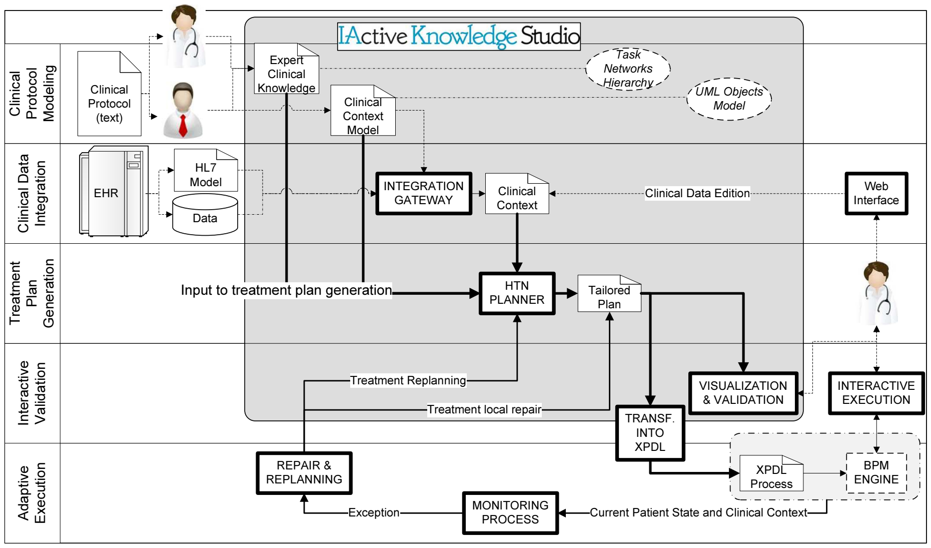


Fig. 1. A diagram showing the main aspects for which the tool provides support: acquisition and representation of clinical protocols, based on the Context Model and the Expert Knowledge Model, automated generation of patient-tailored treatment plans, validation of such plans and integration with external EHR, and adaptive execution of the plans automatically generated.

OncoTheraper 2.0 is intended to fully support the whole life-cycle of clinical decision support above described: from a formal representation of clinical protocols, based on a temporal Hierarchical Task Networks (HTN) representation, the system automatically generates patient tailored, long-term, treatment plans which integrate external patient clinical data. These plans can be seen as patient-centered clinical pathways, that is, time annotated workflows instantiated for a given patient. Therefore, the system also supports the interactive and dynamic execution of these workflows, by clinical staff, based on standard BPM (runtime) engines. This capability is provided by a transformation from the treatment plans obtained into XPDL (XML Process Definition Language), a standard format to interchange workflows (or business processes) between different tools, allowing to integrate the output of the intelligent system (treatment plans generated) with the input of standard workflow tools. Moreover, this functionality opens the way to integrate the output of a CDSS with legacy Clinical Management Systems (most of them including workflow engines). This transformation is intended to support a rapid-prototyping life-cycle for clinical knowledge applications and has been a key aspect in the process of capturing application requirements. Furthermore, the system also incorporates techniques that provide support to monitor the state of a patient during the enactment of a treatment, and to dynamically and adaptively respond to exceptions arisen at execution time.

A demonstrator of OncoTheraper2.0 has been implemented and can be executed through a web interface[6].This application allows the introduction of some initial clinical data to describe a patient profile (stratification group, age, genre, body surface, initial blood test, etc.). Receiving as input this initial clinical context and a task networks-based representation of an oncology clinical protocol, OncoTheraper2.0 automatically generates a treatment plan, tailored for the patient profile provided as input. At present the system is capable of obtaining treatment plans based on two versions of the Hodgkin disease protocol [7]. The plans obtained may be visualized either as a list of clinical actions to be performed at a given time, or as Gantt charts, what helps clinicians to validate such treatment plans. Once a treatment plan is validated by clinicians, it is deployed on a BPM engine [8] which supports its interactive execution by clinical staff. In addition, a monitoring component has been developed which keeps track of the state of the patient, and detects any possible deviation from the expected evolution of the patient represented in the treatment plan. When an exception is detected by the monitoring process, a repair and replanning process determines whether the plan can be repaired (if a low criticality error is detected) or a replanning episode has to be triggered.

A detailed description of all these functionalities will be given in the following sections, but firstly we will briefly describe the main issues concerned with the development and integration of the CDSS using IActive Knowledge Studio.

[6] `http://tratamientos.iactive.es`

[7] A version released on 2003, EH-SEOP.2003, and a revised version on 2008, Euronet-PHL-C. Both versions coexist in the pediatrics oncology units that participate in this project. A description of this protocol and how can be represented with Hierarchical Task Networks can be found in [1]

[8] The console chosen for experimental purposes is the Nova Bonita console (http://www.bonitasoft.com/). Besides that both is an open source project and accepts XPDL process as input, the main reason for selecting this tool is its interactive execution model of tasks and its configuration and customization features.

3 GOAL-BASED MODELING OF ONCOLOGY TREATMENT PROTOCOLS BASED ON IACTIVE KNOWLEDGE STUDIO

IActive Knowledge Studio[9] incorporates several planning techniques (see Figure 1) that support the following steps in the whole cycle of clinical knowledge and plans management: clinical protocol authoring (based on an extended Hierarchical Task Networks representation which incorporates a graphical notation [6] inspired by BPM standards [7]), integration with external clinical data (based on transformations on standard UML models), protocol validation and generation of patient-tailored treatment plans (both based on a goal and knowledge-driven planning process that takes as input the clinical protocol modeled and patient data which are seamlessly integrated into the plan).

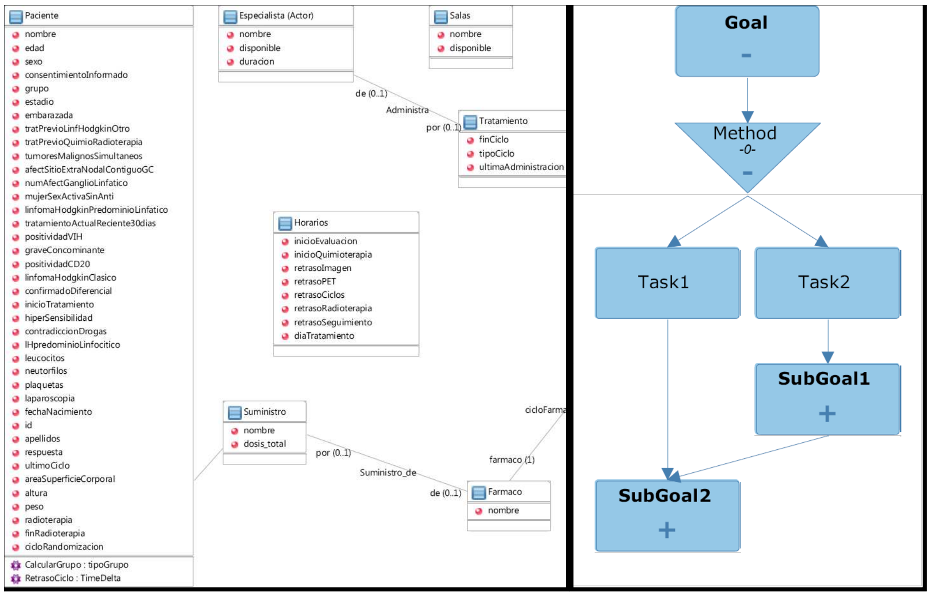


Fig. 2. Main parts of a clinical knowledge model: (Left-hand side) A UML model of some clinical object, (Right-hand side) basic concepts of EKMN.

The most important issue of IActive Knowledge Studio is that it supports to visually model clinical knowledge, particularly clinical protocols, focused on two general concepts: the *Context Model (CM)*, devoted to model clinical data (i.e. domain objects, properties and relations which somehow are involved in a clinical protocol, like patient, drugs, etc.), and the *Expert Knowledge Model (EKM)*, devoted to visually encode clinical decisions and procedures found in a clinical protocol. This is done by using an extended HTN graphical notation, named EKMN (Expert Knowledge Modeling Notation, described in [6]), that is an evolution of the former textual language HTN-PDDL

[9] Download at http://www.iactive.es/productos/iactive-knowledge-studio/

(initially described in [3]) used to encode clinical knowledge in the proof of concept [1] above pointed out in the Introduction.

Domain objects are visually modeled in the Context Model (see Figure 2) following the standard recommendations of UML (Unified Modeling Language [8]), a general-purpose modeling language in the field of object-oriented software engineering. The Context Model is based on five key UML concepts: *Class* (to represent hierarchies of objects types), *Attribute* (to represent object properties), *Operation* (special attributes of objects that need to be computed by a procedure, thus allowing to manage and represent resources) , *Association and Generalization* (to represent relationships between objects).

EKMN is based on four basic concepts: *Compound Task* (or *Goal*), *Decomposition Method*, *Primitive Task* and *Relation* (see Figure 2). Compound Tasks represent high-level processes or goals to be accomplished in possibly alternative ways depending on the context. Methods are used to specify the alternative ways in which either a high-level process may be decomposed or a high-level goal may be achieved. Primitive tasks represent concrete actions the execution of which involve a change in the objects modeled. Relations may be either *hierarchical* (showing dependencies between tasks at different levels) or *compositional* (between a compound task and its decomposing methods), and they may also be used to specify *ordering constraints* (either sequential or parallel). EKMN is inspired by BPMN [7], the current standard notation for process modeling and it allows to intuitively represent a clinical protocol in terms of a compositional hierarchy of tasks representing compound and primitive tasks at different levels of abstraction (see Figure 2). This graphic notation supports a collaborative modeling process (between knowledge engineer and expert) that starts with the development of a visual skeleton of a task/goal hierarchy which represents a clinical protocol. Then, the knowledge engineer details this skeleton by filling out more detailed knowledge in successive refinement steps. A more detailed description on the expressiveness of EKMN and the UML-based model of data can be found in [6].

3.1 Dynamic generation of patient-tailored oncology treatment plans

The generation of patient-tailored oncology treatment plans is based on a knowledge-driven, temporal hierarchical planning process [10] (described in [1]) that receives the following inputs (see Figure 1): the Expert Knowledge Model, the Context Model and a set of object instances, that both accomplish the UML specification of the Context Model and represent the clinical context on which the treatment will be applied. The output of this process is a patient-tailored treatment plan, that is to say, a set of instantiated and temporally annotated primitive tasks which incorporate and integrate patient clinical data. The keystone for obtaining "tailored" plans is that object instances, representing the clinical context, are provided as input and incorporated natively into the plan by the planning process (since it is a reasoning process based, among others, on matching and unification techniques which allows to incorporate object instances into

[10] It is worth to note that this is a temporally extended HTN planner, formerly used in our previous proof of concept, which has evolved as a commercial product developed by our start-up , now called *Decisor* (http://www.iactivecompany.com/products/iactive-intelligent-decisor/).

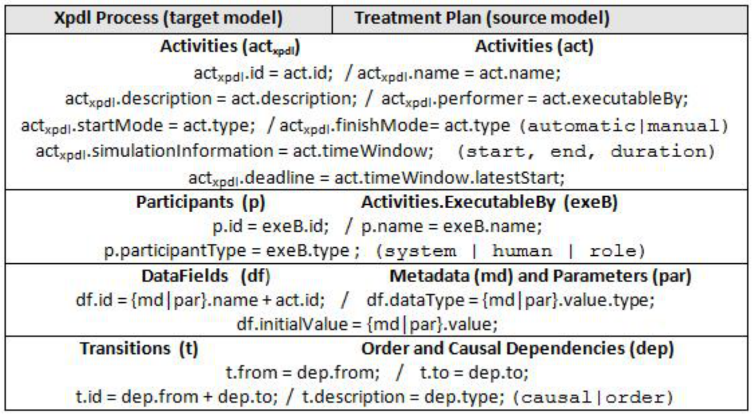
the final treatment plan). Indeed, external EHR data can also be automatically integrated into the planning process, by following an automated process that, previous to the reasoning process, translates the data model of the EHR into the data model of the Context Model.

3.2 Validation of treatment protocols and plans

IActive Knowledge Studio also provides an environment with the necessary functionalities to perform a validation-by-inspection process on the plans generated. On the one hand, it allows to simulate and show the intermediate states produced by the planner when reasoning about how to apply a protocol to a concrete patient. On the other hand, it also provides powerful tools for plan analysis and validation. The treatment plans contain a collection of partially-ordered activities that represents evaluation and drug administration activities (grouped by chemotherapy cycles) to be accomplished on a patient. Each activity is specified by its set of parameters and metadata (defining additional knowledge, as the mode of administration of a drug) and may be allocated to a specific actor for controlling its execution. Moreover, each activity is annotated by temporal information representing its duration and its estimated dates for the start and end. These time points are represented as time intervals with the *earliest* and *latest* dates in which actions are allowed to be executed. Finally, treatment plans contain a collection of *order* and *causal dependencies* generated by the reasoning process of the planner. All the information included in a treatment plan has been verified by the temporal reasoning process and, in addition, it can be visualized either as a sheet or as a Gantt diagram (what allows the intuitive analysis of order dependencies between actions and visually check inconsistencies). In addition, there is a section to show several statistics about the resulting plan (resource usage, like drugs dosages, or actions duration, like chemotherapy cycles, etc.). This can be used by knowledge engineers or experts in order to simulate and analyze different situated treatment cases from the formal representation. This information is used by oncologists in order to analyze the adequacy and adherence of the automatically personalized treatments to the protocol, thus allowing to refine or correct the knowledge modeled.

4 EXECUTION OF TREATMENT PLANS

According to the clinical decision life-cycle above described, after the automatic generation of treatment plans and the validation of such plans by clinical staff, the next step is to enact them in the real world. As previously said, plans can also be seen as patient-centered clinical pathways (or in other words, time annotated workflows instantiated for a given patient). This vision has the advantage that these workflows might be interactively executed based on standard BPM runtime engines and visual consoles. Moreover, there are several standardized formats, like XPDL, which facilitate interchange of workflows (business processes) between different tools. In order to take advantage of the infrastructure already offered by BPM technology, we have integrated the treatment plans with the input of a standard BPM tool based on XPDL (see Figure 1). This integration has additional advantages: on the one hand, rapid prototyping and agile requirements capture, since there is no need to develop ad-hoc execution engines nor interactive interfaces to rapidly obtain an initial prototype that can be evaluated by clinicians. On the other hand, easy integration of the CDSS (at the output) with already existing Clinical Management Systems, many of them based on Workflow and BPM technologies that support administrative and organizational aspects of clinical work. All these issues are based on a transformation from the plan representation into XPDL described in the remainder of this section.



Xpdl Process (target model)	Treatment Plan (source model)
Activities (act_{xpdl})	**Activities (act)**
act_{xpdl}.id = act.id; / act_{xpdl}.name = act.name;	
act_{xpdl}.description = act.description; / act_{xpdl}.performer = act.executableBy;	
act_{xpdl}.startMode = act.type; / act_{xpdl}.finishMode= act.type (automatic|manual)	
act_{xpdl}.simulationInformation = act.timeWindow; (start, end, duration)	
act_{xpdl}.deadline = act.timeWindow.latestStart;	
Participants (p)	**Activities.ExecutableBy (exeB)**
p.id = exeB.id; / p.name = exeB.name;	
p.participantType = exeB.type ; (system | human | role)	
DataFields (df)	**Metadata (md) and Parameters (par)**
df.id = {md|par}.name + act.id; / df.dataType = {md|par}.value.type;	
df.initialValue = {md|par}.value;	
Transitions (t)	**Order and Causal Dependencies (dep)**
t.from = dep.from; / t.to = dep.to;	
t.id = dep.from + dep.to; / t.description = dep.type; (causal|order)	

Fig. 3. Matching process between a treatment plan and a xpdl process

The most relevant entities of a XPDL process are: *Activities*, comprising a logical and self-contained unit of work carried out by participants and/or computer applications; *Participants*, used to define the organizational model over which a process is deployed and which can be allocated to one or more activities. *Transitions*, used to relate activities in sequential or parallel operations; *dataFields*, used to define the process data model. Figure 3 shows how XPDL entities can be matched with the entities of a treatment plan: XPDL activities are directly matched with plan actions (by directly mapping their corresponding internal properties like name, action type or temporal constraints), XPDL participants correspond to participants of plan actions (represented by the property *ExecutableBy*), XPDL data fields are matched with either parameters or special metadata of plan actions, and, finally, XPDL transitions are matched with order and causal dependencies represented in plans. This mapping has been developed as a transformation model, using Model Driven Engineering standard techniques. Concretely, we have selected the domain-specific language **ATL** (ATLAS Transformation Language - [9]), a language that is compatible with the *Ecore* metamodel [10] which, in turn, has been used to model the representation of the treatment plans generated.

The workflow obtained from this transformation is provided as input to a BPM runtime engine [11] in charge of executing every activity following an execution model that

[11] The console chosen for experimental purposes is the Nova Bonita console (http://www.bonitasoft.com/). Besides that both is an open source project and accepts

relies on a state-based automaton. Activities may have different states (*initial, ready, executing, suspended, aborted* or *finished*) and transitions are mainly related to user requests. This BPM engine include a BPM console (based on web interface) for deploying the process, displaying the activities to be carried out and providing visual *gadgets* to interactively control the execution of them. The execution of a treatment plan is a collaborative process carried out by different members of a clinical team, and platforms such as BPM consoles are an excellent support for such distributed work.

5 Dynamic Response to Exceptions

The high dynamism of the clinical environment makes difficult to exactly predict all the activities of a careflow, so long-term plans have to be frequently adapted according to the health conditions and the progress of the patient. Therefore, a monitoring process (in order to check that everything is going as planned and to detect exceptions) and an adaptation strategy (for dynamically responding to such unexpected events) are also required during the execution of treatment plans. Both functionalities are described next.

5.1 Monitoring Process

The representation of the treatment plans generated include some useful information from the monitoring activity standpoint, since every activity (apart from the information above described) contains a list of *preconditions* and *effects*. They are encoded in the protocol modeling step, in order to describe, respectively, which conditions have to be accomplished in order to execute a clinical task, and which properties and relations of objects are affected by its execution. Hence the monitoring activity may be summarized as: (1) Checking that the list of preconditions is met in the real world (before executing an activity) and supervising that the list of effects of an action has been correctly applied (after its correct execution). (2) Confirming that the activities have been initiated and finished their execution according to their *temporal information*. (3) Supervising that the causal structure of the plan is consistent, i.e., it does not have any failed causal links. Since an error in a causal link involves a future failed precondition, the check of this structure allows us to detect exceptions as soon as possible (thus having additional time to suggest a valid response). (4) Finally, the monitoring component supports both decisions made by healthcare professionals in runtime (e.g. decreasing the dosage of a drug to be administered to a patient by a bad reaction) and exceptions detected by such experts (e.g. the progress of the patient is not the expected one and a new treatment subplan is required). So far, we have described how to interactively execute treatment plans and how to develop a monitoring activity. The next step is to suggest a flexible and dynamic response after an exception is detected by the monitoring process and this topic is dealt in the next section.

XPDL process as input, the main reason for selecting this tool is its interactive execution model of tasks and its configuration and customization features.

5.2 Dynamic Repair & Replanning of treatment plans

A strategy for tackling any exception during the execution of a treatment plan should fulfill the following desirable features. On the one hand, 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 the patient's parameters as his body mass or age) 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 therapy plan, thus promoting the *plan stability* [11]. The reason of this conservative attitude is that, when the execution of a therapy 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 can be costly. Finally, the strategy should exploit the reasoning and decision-making process performed by the AI 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.

Taking all these requirements into account, we propose a *repair* and *replanning* strategy (R&R) that, for the time being, is in its early stage of development. Such strategy is organized into a multilevel architecture (see figure 4(a)) according to the criticality (obtained as measure related to the number of actions affected by the exception) and the nature of the exception detected. Hence the two lower layers are aimed to carry out *repairs*, i.e., applying local changes or a *repair rule*. If this alternative is not effective enough, a *replanning* episode is initiated for generating a new sub-plan according to the current context. A *mixed-initiative* approach is also considered, as last alternative.

Depending on the level in charge of managing an exception handling episode, the R&R strategy will exhibit different degrees of autonomy. The exception handling process always starts at the lowest levels (first and second one) which are intended to be fully autonomous. As explained below, these levels are endowed with both appropriate automated planning machinery and exception handling rules in order to repair a treatment plan without human intervention. If the plan cannot be repaired by the techniques of that levels, the third level will be responsible of making replanning recommendations, based on the knowledge encoded in the Hierarchical Task Networks domain. Finally, the R&R process will exhibit the lowest degree of autonomy at the fourth level, since it will be guided by the intervention of a human expert. The degree of autonomy of the proposal and the degree of user involvement depend on the complexity of the failure to solve, i.e., the resolution of complex exceptions (known as *hazard* exceptions - [12]) requires more expert participation than solving an *obstacle* error (that is simpler). For example, errors identified by clinicians at runtime (which commonly are more difficult to solve) require more expert participation, since the way to respond to these error is not explicitly modeled in the planning domain and, therefore, the monitoring component is unaware of how to solve them autonomously.

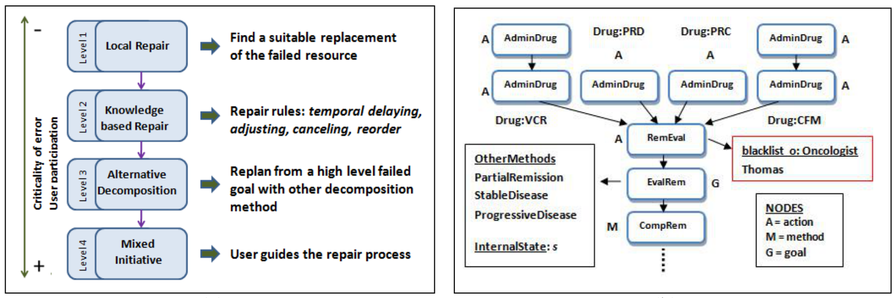


Fig. 4. R&R Strategy and extract of a Decision Graph

In order to avoid the replanning from scratch, the proposal relies on the previous reasoning and decision-making process carried out by the planner for the construction of the original plan. The previous reasoning process gives as result the **causal structure** of the plan and its analysis makes possible to know the set of actions affected by an error (called *impact* of the failure). On the other hand, the sequence of decisions taken by the planner during the plan construction are recorded in an additional structure called **decision graph**. Taking the HTN paradigm into account [13], we identify the following decisions: (1) the list of *resources* whose attributes made true the preconditions of an activity or of a decomposition method (list of *unifications*) and (2) the method selected to decompose a compound task (the way chosen to achieve a goal). This structure also records the failed decisions that did not lead to a solution state (called *black-list*). In the case of goals (compound tasks), the decision graph records the list of alternatives (other decomposition methods) for reducing them. All these entities are useful to make a repair or to initiate a replanning episode as seen bellow. Figure 4(b) shows an extract of a decision graph related to an oncology treatment plan. In this example, the resource *Thomas* belongs to the black-list related to *oncologists* resources in the node of the action *RemEval*, i.e., in planning time *Thomas* could not be assigned to this action. In the next sections, we explain briefly each level of the R&R strategy:

Level 1: Local Repair. This level attempts to solve errors caused by a failed resource that makes the preconditions of an activity fail (e.g. the oncologist is not available). The following assumptions must be considered: the selected resource may affect the duration of the activity in which is used and it may be selected according to a sorting criterion (useful for establishing priorities among resources, e.g., choose the doctor with the best availability). The repair is based on finding (among the total resources of the same type) a suitable replacement of the resource that satisfies the preconditions of the failed action, mets its temporal restrictions and does not cause any *threats* (not to be used simultaneously).

Level 2: Knowledge-based Repair. 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 error and even on the experience of clinicians on previous episodes. Hence the objective of this level is to define a collection of *generic repair rules* (such as *temporal delaying rule*, an *adjusting* or a *canceling rule*), that are selected in runtime by healthcare professionals for solving an unexpected event. We expect to increase this set of rules to cover more typical exceptions.

Level 3: Alternative Decomposition. The previous levels seen so far try to make local changes for solving the failed plan. But if the plan remains invalid to the current context, the solution here is to initiate a replanning episode trying to achieve the goal in a different way, i.e., invoking the planner from the compound task (to which the failed actions belongs) and with an alternative decomposition method (taken the records of the decision graph into account). For example, regarding the decision graph shown in figure 4(b), if the progression of the patient is not as planned (the tumour has a partial instead of a complete remission) we will invoke the planner from the goal *EvalRem* and with the decomposition method *PartialRemission*. If there are not any valid sub-plans, a *mixed-initiative* process will start in the next level.

Level 4: Mixed-Initiative Approach. Considering that an effective R&R mechanism should be a domain-independent strategy, it is worth to note that in the clinical context the type of repair to apply depends on the considered application domain, on the detected error, on the current health conditions of the patient and even on the experience of clinicians on previous episodes. For this reason, the aim of this level is to propose a framework in which healthcare professionals can conveniently interact to the R&R strategy in order to guide the plan adaptation process.

Although the repair and replanning strategy is in its early stage of development, we consider that it has several advantages as: *efficiency* (reuse the previous reasoning process), *flexibility* (local or global changes), *stability* (minimize the changes in the plan), *coverage* (the same strategy for solving a wide variety of errors), *interactivity* with the user (participating both in execution and in plan repair activities) and *acceptance* (reduction of his cognitive load at trying to make a repair before a replanning episode).

6 RELATED WORK

Regarding the interactive execution of personalized careflows, we mention the work [14]. All the approaches presented in this work incorporate execution engines that are specifically developed for the representation language used to model clinical knowledge. This feature hinders the integration of the treatments managed with external Clinical Management Systems, since these systems also include proprietary (yet standard) execution engines for organizational activities. As opposite to this, the approach here presented for the execution of treatment plans opens the way to seamlessly integrate the treatment plans obtained with other standard BPM execution engines. Other approaches are being developed in this direction (like [15]), but this work is not yet fully implemented, thus showing that execution and monitoring of careflows is an ongoing research area. On the other hand, the idea of dynamically adapting a failed treatment plan has been discussed from different perspectives, as in [12], which presents a generic approach to handling unexpected events (classified into *hazards* and *obstacles*) at runtime. This work is based on a continual planning approach, that is, some parts of the treatment plan are defined as goals, allowing to delay clinical decision until more information about the patient state is available. In addition, if the clinician knows precisely how some actions of the personalized treatment have to be implemented, then those sections are defined as tasks and not as goals. Therefore, clinicians can not dispose of the *complete* personalized treatment plan because there may be pending goals to achieve in the initial treatment, and they are *refined* during plan execution. Our approach obtains a fully instantiated and personalized treatment plan, including all the expected actions and resources which should be used during its execution. This has been a hard requirement of oncologists, since they prefer having both, a complete initial prediction of the expected evolution of the patient, and the chance to change the treatment (based on the monitoring and repair/replanning process presented) once an unexpected, non foreseen exception occurs.

7 CONCLUSIONS

In this work we have described the most relevant issues addressed in the development of OncoTheraper2.0, a Knowledge Based Clinical Decision Support System aimed at supporting the full clinical decision life-cycle of oncology treatments. The system provides support for the modeling of clinical knowledge based on UML and EKMN (an extended HTN graphical knowledge representation), as well as the dynamic generation of patient-tailored treatment plans (based on a temporal planning process, guided by the knowledge so encoded, which integrates clinical information from external EHRs). In addition, it supports the validation of clinical protocols and treatment plans (performing use-case analysis based on the same reasoning process), as well as the execution of the patient-tailored plan generated (allowing to integrate the output of the system with external BPM tools). Finally, it incorporates a state-based monitoring process which tracks the patient evolution and detect fails (analyzing the preconditions and effects represented in the actions of the treatment plan) and, finally, it is capable of responding adaptively to the exceptions detected during the execution based on a plan repair/replanning strategy.

The modeling, generation and validation of treatment plans have been supported by IActive Knowledge Studio, a suite of tools devoted to the development of KB intelligent systems, and particularly, Clinical Decision Support Systems. The tool allows users (knowledge engineers or clinical staff) to generate situated treatment cases from the formal representation, to visualize them as sheets or as Gantt diagrams, as well as showing several statistics (about drug usage, milestones, etc.). This information is used by oncologists in order to analyze the adequacy and adherence of the automatically personalized treatments to the protocol, thus allowing to refine or correct the knowledge model. Moreover, the modeling and development life-cycle described also can be complemented with an automatic knowledge acquisition process from other CIGs representation languages (presented in [16]) into our knowledge model. This clearly opens the possibilities of standardization of our technology since it can be used to develop clinical decision support services where knowledge modeling can be represented in other standard formalism, and exploited through the techniques and tools here described. The implementation of the system is being completed through the incorporation of the replanning strategy, which constitutes its last part, that is being developed. Several proofs are being carried out about the integration with external information sources, through the development of a demonstrator in the Health Living Lab of Andalusia. At present, an ongoing real integration process is being carried out in the EHR and HIS of the Hospital of Jaén (Spain)[12].

References

1. Fdez-Olivares, J., Cózar, J., Castillo, L.: Oncotheraper: Clinical decision support for oncology therapy planning based on temporal hierarchical tasks networks. In Riaño, D., ed.: Knowledge Management for Health Care Procedures. Volume 5626 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2009) 25–41
2. Fdez-Olivares, J., Castillo, L., Cózar, J.A., Pérez, O.G.: Supporting clinical processes and decisions by hierarchical planning and scheduling. Computational Intelligence **27**(1) (2011) 103122
3. Castillo, L., Fdez-Olivares, J., García-Pérez, O., Palao, F.: Efficiently handling temporal knowledge in an HTN planner. In: Proceeding of ICAPS06. (2006) 63–72
4. Nau, D., Muoz-Avila, H., Cao, Y., Lotem, A., , Mitchel, S.: Total-order planning with partially ordered subtask. In: Proceedings of the IJCAI-2001. (2001)
5. González-Ferrer, A., Fdez-Olivares, J., Sánchez-Garzón, I., Castillo, L.: Smart process management: automated generation of adaptive cases based on intelligent planning technologies. Proceedings of the Business Process Management 2010 Demonstration Track (2010)
6. Palao, F., Fdez-Olivares, J., Castillo, L., García, O.: An extended htn knowledge representation based on a graphical notation. In: Submitted to Workshop Knowledge Engineering for Planning and Scheduling, ICAPS 2011. (2011)
7. White, S.: Introduction to BPMN. IBM Cooperation (2004) 2008–029
8. Booch, G., Rumbaugh, J., Jacobson, I.: The unified modeling language user guide. Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA (1999)
9. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: Atl: A model transformation tool. Science of Computer Programming **72** (2008) 31–39
10. (EMF), E.M.F.P.: Ecore metamodel. `http://www.eclipse.org/modeling/emf/?project=emf`
11. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan stability: Replanning versus plan repair. In: Proc. of ICAPS06. (2006) 212–221
12. Grando, A., Peleg, M., Glasspool, D.: A goal-oriented framework for specifying clinical guidelines and handling medical errors. Biomedical Informatics **43** (2010) 287–299
13. Sacerdoti, E.: The nonlinear nature of plans. In: Proc. of IJCAI. (1975) 206–214
14. Isern, D., Moreno, A.: Computer-based execution of clinical guidelines: A review. Medical Informatics **77**(12) (2008) 787–808
15. Miller, K., MacCaull, W.: Toward web-based careflow management systems. Emerging Technologies in Web Intelligence **1**(2) (2009) 137–145
16. González-Ferrer, A., Teije, A.T., Fdez-Olivares, J., Milian, K.: Careflow planning: from time-annotated clinical guidelines to temporal hierarchical task networks. In: Proceedings of AIME 2011. (2011)

[12] This work has been partially supported by the Andalusian Regional Ministry of Innovation under project P08-TIC-3572.