Integrating plans into BPM technologies for Human-Centric Process Execution

Juan Fdez-Olivares and **Inmaculada Sánchez-Garzón** and **Arturo González-Ferrer** and **Luis Castillo**
Department of Computer Science and Artificial Intelligence
University of Granada

Abstract

This work presents a translation process from a standard representation of plans into a standard executable format for Business Process Management (BPM). This translation is conceived as a Knowledge Engineering for Planning process that bridges the existing gap between AI Planning and Busines Process Management and provides support for the direct execution of plans playing the role of Human-Centric processes.

Motivation

Human-Centric processes (Dayal, Hsu, and Ladin 2001) are collections of tasks, mainly organized in sequential and/or parallel control flows, which necessarily require human interaction in order to control and manage their execution. They are very common in any organization and they can be seen as complementary to System-Centric processes, which are devoted to exhaustively automate the data flow and processes of an organization, reducing human intervention to the minimum. This work is focused on a special kind of Human-Centric processes, concretely those oriented to *knowledge workers*(Myers et al. 2007): highly qualified personnel, like experts or decision makers, who need and produce knowledge in their daily work. These processes commonly support decisions and help to the accomplishment of workflow tasks in several application domains. Examples of such processes are a forest fire attack plan devoted to fire fighting technical staff, a medical treatment plan for a clinician, an e-learning course for a teacher, a military forces deployment plan for a commander, etc. For the sake of simplicity, we will designate these processes as *Smart Processes*.

The management and execution of *Smart Processes* demand special technological requirements, due to its main features(WorkflowManagementCoalition 2010): first, these processes respond to very complex, interacting sets of procedures and doctrine which reside either in an unstructured form in experts' mind or in partially structured documents, what makes difficult to generate and execute tasks in conformance with those constraints; second, they are unpredictable in the sense that both their composing tasks and order relations cannot be easily devised prior to their execution, since they strongly depend on the context of the organization and do not respond to a fixed pattern.

Hence they need to be somehow *modeled* and *dynamically generated*, their generation must be *adaptable to the context* of the organization and, finally, smart processes have to be *flexibly and interactively executed* by humans. In summary, they require some kind of intelligent management since they are very difficult to foresee and need to be adaptively generated depending on the context (current state) of an organization.

AI P&S has showed to be very suitable in many applications ((Fdez-Olivares et al. 2006; Castillo et al. 2007; Bresina et al. 2005; Fdez-Olivares et al. 2010)) as a technology that fulfills the above requirements. The role of AI P&S in this area, fundamentally HTN-based paradigms (Sacerdoti 1975; Castillo et al. 2006), is well known: starting from a planning domain where expert knowledge (in the form of actuation protocols or operating procedures) is *modeled* as a hierarchy of tasks networks, a plan (representing a course of actions to be accomplished at a given time) is *adaptively generated* as the result of a planning process. Then, the plan is *executed* by humans who, depending on the application may be ground operators, military personnel, experts in forest fire fighting, clinicians, etc., and this execution is supported by ad-hoc task visualization and execution models and tools (Fdez-Olivares et al. 2006; 2010).

On the other hand, a leading industrial area that has showed to be successful in the management and execution of Human-Centric processes is BPM (Business Process Management)(wfm), devoted to the modeling, deployment, execution and monitoring of business processes. From the concrete point of view of process execution, BPM technology provides: (1) *runtime engines* that support the execution of tasks based on robust task execution models, and (2) *visual consoles* (at present based on web portals) that support user interaction for the control of human-centric tasks. However, at the time being, BPM technology is mainly focused on static, repetitive, even perfectly predictable tasks/processes, mostly devoted to low qualification operators(WorkflowManagementCoalition 2010).

This is a widely known weakness in the BPM community (either industrial or academic) and, because of this, it is also recognized that new techniques must be devel-

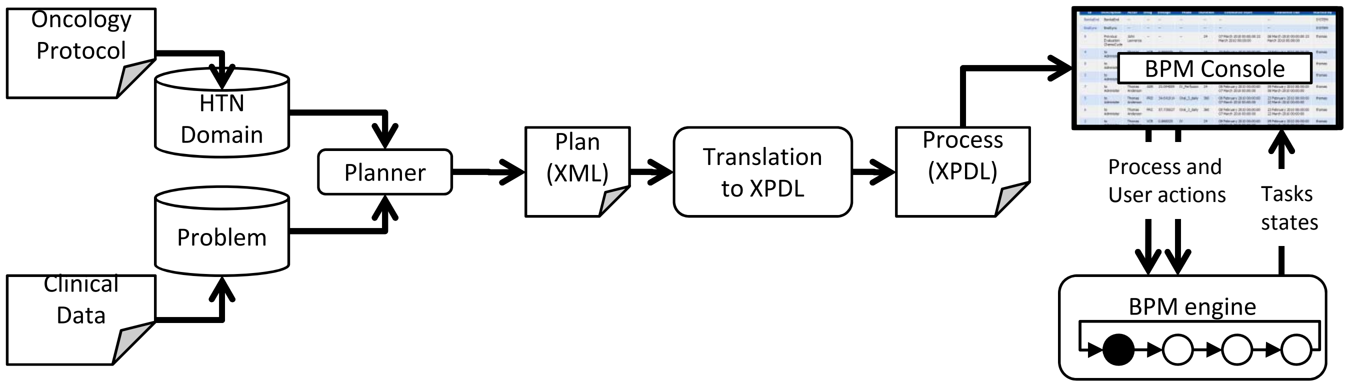


Figure 1: Integrating oncology treatment plans into both, a BPM console and a BPM engine

oped at the process modeling/generation step in order to fully cover the needs of knowledge workers on Smart Processes. With respect to AI P&S weak points, it is necessary to recognize that since most of the planning applications in Human-Centric processes are based on ad-hoc developments for task interactive execution (Bresina et al. 2005; Fdez-Olivares et al. 2006; Tate, Drabble, and Kirby 1994; Wilkins 1990), these developments are still far from being so stable, mature and usable like in BPM.

In summary, while AI P&S (concretely HTN paradigm) has proven to be successful on supporting the knowledge workers' *effort* (by modeling their expertise and helping them to adaptively produce plans to support their decisions), it can be seen that BPM is much more appropriate to support *the result* of this effort (by providing technological infrastructures in order to interactively execute and monitor processes). As a conclusion, it becomes relevant to analyze in what extent AI P&S technology might cover the lack of capability of BPM regarding the modeling and adaptive generation of plans. In addition, AI P&S may take advantage of an already tested, developed technology in order to enhance the user experience of a planning application at the execution and monitoring stage.

Consequently, this work faces the problem of integrating the capability of adaptive, dynamic generation of plans that we can find in AIP&S with the high-performance of BPM with respect to interactive execution of Human-centric processes. Concretely, we have interpreted the solution to this problem as the development of Knowledge Engineering techniques, focused on plan representation and postprocessing, in order to make the output of an AI planner understandable by a BPM runtime engine. The convergence of both technologies leads into an integrated environment for *Smart Process Management*, providing support for *modeling* (based on the representation of Hierarchical Task Networks), *adaptive generation* (based on Planning and Scheduling process) and *execution* (based on BPM runtime engines and consoles) of Smart Processes.

The adoption of this approach has many advantages for AIP&S in practical applications: the integration of a plan previously generated by a planning engine into an already developed, BPM standard environment for interactive execution of processes might support a rapid prototyping development life-cycle, saving development time at the first stages in the development of a AI P&S application. This also would allow to carry out a reliable acquisition of user requirements based on a rapid-prototyping methodology. In addition, user experience may be improved, helping to reduce/eliminate a constant bottle-neck in the adoption of AIP&S as a widely spread technology. From the BPM point of view, integrating a planner into its functional life-cycle, will leverage any BPM system allowing to fulfill all the requirements imposed by needs of knowledge workers.

In the following sections, in order to bring this arguments into reality, we will introduce a Knowledge Engineering approach based on the postprocessing and translation of plans into a BPM standard representations of processes. The result of this translation will be considered as the input of a BPM runtime engine that, highly coupled with a web console, will support the interactive execution of smart processes. In order to demonstrate the suitability of this approach, we have performed some experiments in the medical domain. Concretely we have achieved to execute, by using a commercial BPM runtime engine, pediatrics oncology therapy plans previously generated by a hierarchical planner. The therapy plans obtained by our planner are a clear example of what human-centric smart processes are, since they are primarily useful to support clinical decision making and need to be interactively executed by oncologists. Technical aspects of the plan representation and the translation process are detailed in last sections. Previously, the case study on therapy planning and some necessary background concepts on BPM are introduced.

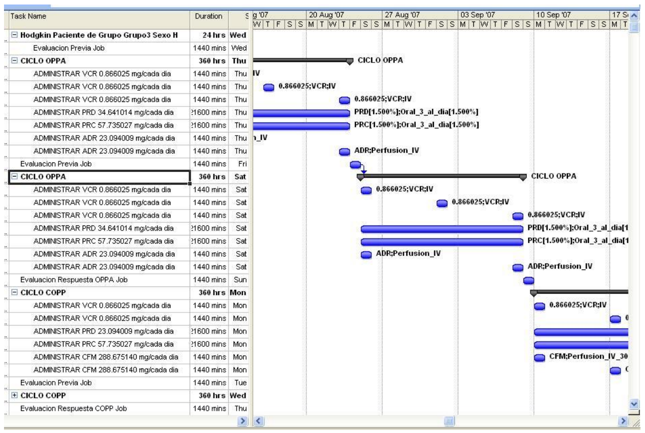
Therapy planning case study

The work presented in this paper is being carried out in the framework of a research project aimed at developing a Clinical Decision Support System (called *OncoTheraper*), based on planning and scheduling techniques, in the pediatrics oncology area. OncoTheraper is intended to support oncologists's effort (they are the knowledge workers in this case study) when they deal with the problem of planning an oncology treatment for a given patient. These experts make their decisions following *oncology treatment protocols*, a set of evidence-based operating procedure and policies that are gathered in partially structured documents. The system is based on a temporally extended HTN paradigm (see (Fdez-Olivares et al. 2010) for more details) that, on the one hand, supports to model treatment protocols on the basis of an HTN temporal planning language. On the other hand, it allows to dynamically generate user-acceptable treatment plans, adapted to a context defined by a concrete patient profile, by following a planning process driven by the expert knowledge modeled in the planning domain.

In a previous experimentation, reported in (Fdez-Olivares et al. 2010; Fdez-Olivares, Czar, and Castillo 2009), a model of a concrete oncology clinical trial protocol (the one followed at present for planning the treatment of Hodgkin's disease and elaborated by the Spanish Society on Pediatrics Oncology) has been encoded in the temporally extended HTN planning language, following a knowledge elicitation process based on interviews with experts. This model contains knowledge about wokflow control structures included in the treatment protocol, temporal constraints to be observed between chemotherapy cycles, periodic patterns to administrate drugs as well as the representation of oncologists' working shifts.

In the experiments performed, the planner received the following inputs: a planning domain, representing this protocol; an initial state representing some basic information to describe a patient profile (age, sex, body surface, etc.) as well as other information needed to apply administration rules about drugs (dosage, frequency, etc.); and a high-level task representing the goal (apply the protocol to the patient) with temporal constraints representing the start date of the treatment plan. The output of the planner are plans that contain collections of (partially) ordered tasks representing drug administration actions to be accomplished on a patient. Since temporal information is crucial for oncology treatments, all the actions in a plan are temporally annotated with constraints on start and end dates which specify deadlines either for the estimated beginning and finalization of tasks. These plans are represented in a standard XML representation that allows to display them as Gantt charts in standard tools devoted to project management (like MS Project, see Figure 2)

Furthermore, OncoTheraper is also intended to support the execution of the treatment plan, that is, the *result* of the process followed by oncologists (now supported by the AI planning system) when planning a treatment. In the work here presented, we are exploring how the treatment plans, dynamically generated on the basis of medical knowledge, can be made executable in order to support the deployment and supervision, step by step, of all the planned treatment tasks. Clearly, the treatment plan generated by AI P&S techniques becomes a human-centric process and oncologists need a platform to visualize and interact with the plan generated, controlling the execution of the tasks defined in the treatment plan. Figure 1 illustrates this idea: since BPM consoles and runtime engines have shown to be successful in the execution of human-centric processes, it seems



Task Name	Duration	S
Hodgkin Paciente de Grupo Grupo3 Sexo H	24 hrs	Wed
Evaluacion Previa Job	1440 mins	Wed
CICLO OPPA	360 hrs	Thu
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Thu
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Thu
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Thu
ADMINISTRAR PRD 34.641014 mg/cada dia	21600 mins	Thu
ADMINISTRAR PRC 57.735027 mg/cada dia	21600 mins	Thu
ADMINISTRAR ADR 23.094009 mg/cada dia	1440 mins	Thu
ADMINISTRAR ADR 23.094009 mg/cada dia	1440 mins	Thu
Evaluacion Previa Job	1440 mins	Fri
CICLO OPPA	360 hrs	Sat
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Sat
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Sat
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Sat
ADMINISTRAR PRD 34.641014 mg/cada dia	21600 mins	Sat
ADMINISTRAR PRC 57.735027 mg/cada dia	21600 mins	Sat
ADMINISTRAR ADR 23.094009 mg/cada dia	1440 mins	Sat
ADMINISTRAR ADR 23.094009 mg/cada dia	1440 mins	Sat
Evaluacion Respuesta OPPA Job	1440 mins	Sun
CICLO COPP	360 hrs	Mon
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Mon
ADMINISTRAR VCR 0.866025 mg/cada dia	1440 mins	Mon
ADMINISTRAR PRD 23.094009 mg/cada dia	21600 mins	Mon
ADMINISTRAR PRC 57.735027 mg/cada dia	21600 mins	Mon
ADMINISTRAR CFM 288.675140 mg/cada dia	1440 mins	Mon
ADMINISTRAR CFM 288.675140 mg/cada dia	1440 mins	Mon
Evaluacion Previa Job	1440 mins	Tue
CICLO COPP	360 hrs	Wed
Evaluacion Respuesta COPP Job	1440 mins	Thu

Figure 2: A temporally annotated and automatically generated therapy plan represented as a gantt chart. The plan represents the treatment for a patient following the Hodgkin's Disease Protocol. Start and end dates of every action are shown in the left-hand side. Drugs and their dosage are shown in the bars of the chart.

that a plan (in our example a treatment plan) adequately transformed into a standard business process representation, might be interactively executed by a standard/commercial runtime engine together with a BPM web console. In order to test this hypothesis we have considered the following steps:

1. The structure and content of plans generated by the hierarchical planner are postprocessed into an XML representation for plans.
2. The XML representation of plans has been translated into XPDL, a widely known standard representation of business processes. The result of this translation is the input to a BPM runtime engine.
3. The XPDL process is deployed into the BPM console in order to display its tasks and provide appropriate visual "gadgets" to support the interactive control of the execution of tasks.
4. The execution of tasks is fully accomplished by a BPM runtime engine,following a task execution model based on a state-based automaton. The engine is also in charge of capturing user actions, sent by the console, an changing accordingly the states of the tasks. New states of tasks are sent back and visualized into the BPM console.

From a Knowledge Engineering for Planning point of view, the translation of a plan into a different model raises the question of which categories of information a plan should contain in order to be executable by a standard BPM engine. Next section is devoted to clarify this question, since an important part of the answer comes from the analysis of the information model of the target language of the translation process, as well as from the analysis of the execution model carried out by the runtime engine and from the requirements about visualization and interaction of the BPM console.

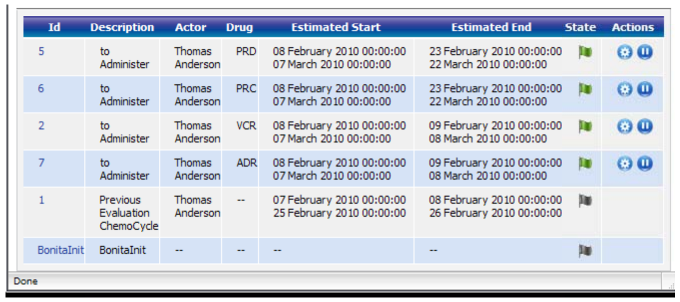
Technological Environment

A great part of the ideas developed in this work need to know in some extent the terminology, standard languages and concepts involved in the area of BPM. A business process is a collection of activities with order relations and control structures that define their execution flow. Processes are defined using a standard BPMN notation, through a Business Modeling tool. The result of such definition takes the form of a XPDL file. XPDL (*XML Process Definition Language*) is a standard language (wfm) , based on XML, devoted to promote process exchange between BPM engines. A BPM runtime engine is in charge of executing a process, usually represented in XPDL, by following the execution flow described. Since human-centric processes require the interaction of human users during its execution, most commercial runtime engines have also coupled a web console that support user interaction. The most relevant XPDL entities and attributes considered in our work are:

Activities. They comprise a logical, self-contained unit of work, which will be carried out by *participants* and/or computer applications. Activities are related to one another via *transitions*. **Transitions**. They result in the sequential or parallel operation of individual activities. **Participants**. They are used to define the organizational model over which a process is deployed and can be allocated to one or more activities.**Parameters and DataFields**. These entities are used to define the process data model. Information that is internal to the process is represented as *Data Fields* and information required outside the process is represented by *Parameters*.

In addition the information model of any XPDL entity may be extended by using *Extended Attributes*.

BPM runtime engines



Id	Description	Actor	Drug	Estimated Start	Estimated End	State	Actions
5	to Administer	Thomas Anderson	PRD	08 February 2010 00:00:00 07 March 2010 00:00:00	23 February 2010 00:00:00 22 March 2010 00:00:00	⚑	⚙ ⏸
6	to Administer	Thomas Anderson	PRC	08 February 2010 00:00:00 07 March 2010 00:00:00	23 February 2010 00:00:00 22 March 2010 00:00:00	⚑	⚙ ⏸
2	to Administer	Thomas Anderson	VCR	08 February 2010 00:00:00 07 March 2010 00:00:00	09 February 2010 00:00:00 08 March 2010 00:00:00	⚑	⚙ ⏸
7	to Administer	Thomas Anderson	ADR	08 February 2010 00:00:00 07 March 2010 00:00:00	09 February 2010 00:00:00 08 March 2010 00:00:00	⚑	⚙ ⏸
1	Previous Evaluation ChemoCycle	Thomas Anderson	--	07 February 2010 00:00:00 25 February 2010 00:00:00	08 February 2010 00:00:00 26 February 2010 00:00:00	⚑	
BonitaInit	BonitaInit	--	--	--	--	⚑	

Done

Figure 3: A typical BPM console showing a collection of tasks. For each one temporal information, execution state and execution controls are shown.

Most BPM systems include three main components: a Business Process Modeler (a tool oriented to IT professionals who visually design a business process), a BPM Runtime Engine (in charge of executing the activities represented in a XPDL process that, as said above, is a serialization of the business process visually designed) and a BPM console. From the user point of view, the BPM console is the most important component and it is closely related with the runtime engine, being responsible of (1) providing user interaction in order to deploy a business process previously defined by the Business process Modeler, (2) visualizing the process activities to be carried out, and (3) providing visual ”gadgets” to interactively control the execution of process activities. Figure 3 shows a snapshot of the console used in the experiments of this work.[1]

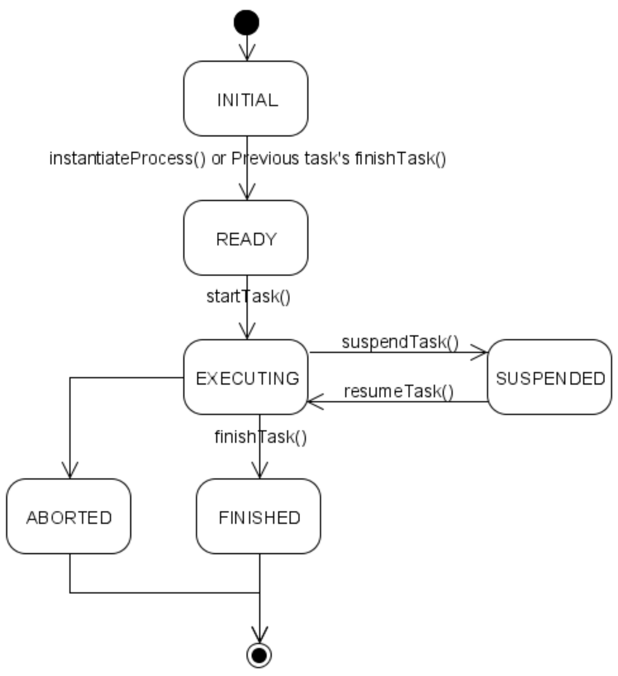


Figure 4: A BPM runtime execution model for Human-Centric tasks.

Regarding process execution, BPM engines are commonly endowed with the necessary machinery in order to execute every task in a process following an execution model based on state-based automata. Fig 4 illustrates the states and transitions of the automaton underlying task execution for the engine that we have chosen to test our concepts. It follows a standard life-cycle for task execution, and similar ones can be found in the literature for BPM engines as well as for clinical plans execution. Thus, there is not lost of generality on the concepts here explained and they can be applied to another different BPM engine. As shown int the figure, the engine allows to start, finish, suspend or abort any task, always upon user request. A task may be in a READY state if all its previously ordered tasks have been finished. Then, it can be started by the user and the engine changes its state to EXECUTING. At this point a task may change

[1]The console chosen for experimental purposes is Nova Bonita console that also includes the Nova Bonita runtime engine (http://www.bonitasoft.com/). Besides that both are Open Source projects and accept XPDL as input, the main reason for selecting these tools is that they support the interactive execution of tasks based on a configurable, simple yet expressive execution model.

either to SUSPENDED, ABORTED or FINISHED, depending on user actions. Every change of state has associated a *trigger* (a java method) that can be customized by the developer. This provides support to define the behaviour of the engine as required by users. Furthermore, triggers opens the possibility of communicating the engine with external systems like a plan monitoring service: the monitor may receive information on critical changes in the execution of a process and respond to them accordingly, for example raising a re-planning process.

Finally, on the basis on this execution model, the basic principles of an acceptable execution of plans can be obtained, with the following considerations:

1. Though most BPM engines support the execution of processes with conditional and repetitive control structures, due to the nature of clinical treatment plans, only sequential and parallel control structures are addressed.
2. BPM engines do not provide full support for the execution of processes with tasks incorporating time constraints(Gagné and Trudel 2009). Indeed, the engine used in this work is only capable of directly manage deadlines for the termination of tasks. This is a really weak point that forces to develop special monitoring services to provide full temporal information management like, for example rescheduling of dates upon user request. Therefore, the full treatment of temporal constraints falls out of the scope of this paper.

Next the XML plan representation used in this approach is explained.

Plan representation

The plans generated by the planner are represented in XML as collection of *Task nodes* (see Figure 5) where every Task node contains information about:

- Activities (*id* and *name*) and their parameters (*type*, *name* and the *value* assigned at planning time), its preconditions and effects.
- Temporal information of activities (*earliest Start* and *earliest End*), representing the estimated time (obtained at planning time) for the start and end of every task in the plan (start, end, duration). Indeed, the plan obtained includes richer temporal information, since it is deployed over a temporal constraint network that assigns to every task a start and end time points represented as time intervals with the earliest and latest start and end dates at which an action is allowed to be executed ($[a_{earlieststart}, a_{lateststart}]$ and $[a_{earliestend}, a_{latestend}]$). However, the information represented in the xml plan is enough, given the above explained execution model.
- Order dependencies. Every action a contains a collection of order dependencies, one for each action b ordered before a, which allow to establish sequential an parallel runtime control structures. This is a crucial item since its analysis will lead to inform the runtime engine about the set of actions that are immediately ordered after a given one.
- Metadata, which allow to represent additional knowledge required by either the user, the console or the runtime execution engine. They are syntactic structures that are managed (created, assigned, etc.) at planning time, and are intended to be interpreted by external systems. Indeed, they are the keystone to enrich a plan in order to facilitate plan postprocessing steps and integration with other systems. For a given action, the meta-data field is a collection of items of the form: `<metadata <name> <valuetype> <value> >`. The plan representation used in this approach embodies the following metadata:

  Description, a string containing user-friendly information about the task.

  Type, used to represent whether the execution of an action necessarily requires human intervention to be initiated ($Type = Manual$) or might be initiated by the engine ($Type = Auto$). This is a very common categorization of actions in Human-Centric processes.

  Actor, considering that we are focused on human-centric processes, it is mandatory that one of the parameters be considered as the resource (either a person or a system) that accomplishses the action.

  Performer, its value is the participant in the process in charge of executing the action from the BPM console.

Meta-data are a very convenient way to encode information that is not directly related with the reasoning process, but which is strongly required in practical applications. Furthermore, they can also be used to extend the knowledge model of actions with additional items requires for a given domain. Indeed, meta-data are encoded in the planning domain, as special tags associated to actions. It is important to note that they can be encoded independently from the BPM processes intended to be executed, as it will be shown in next sections.

Translating plans into XPDL

The translation process is focused mainly in transforming the following pieces of knowledge from a XML plan: activities and its parameters, temporal information, order dependencies and metadata. This knowledge is enough to generate a human-centric process that can be fully executed by a user through a standard BPM engine, since with this knowledge it is possible to generate information in order for the execution model to manage order relations, either sequential or parallel, of tasks as well as execution deadlines. The translation process has three main steps:

1. **Generation of XPDL DataFields/Participants.**The goal of this step is to generate the data model used by the runtime engine. Basically, consists on translating the objects hierarchy, their properties and initial values (defined in the initial state) into XPDL *DataFields*. This step is at present subject to further analysis, in the experiments this has been done semi-automatically.
2. **Generation of XPDL Activities.** For each action a_{plan} in the XML representation of plan, a XPDL activity a_{xpdl} is generated with the following information items:

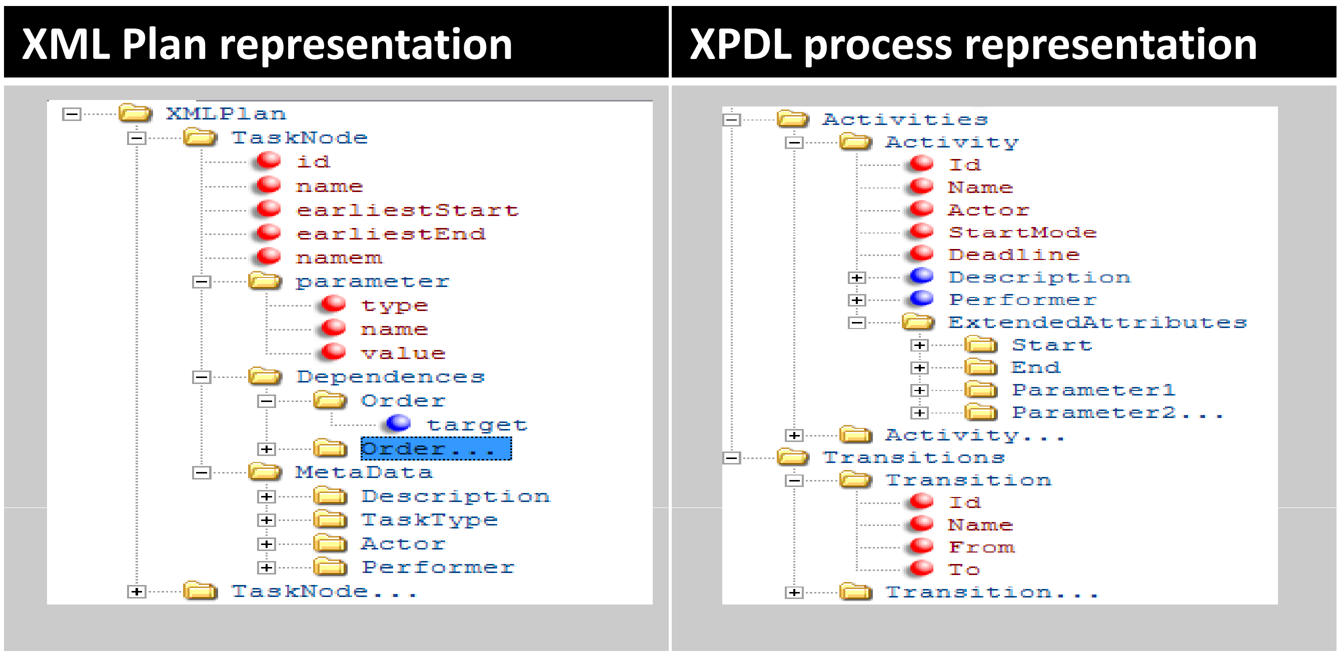


Figure 5: An XML structure of a plan and the XPDL schema of the relevant information for activities and transitions used in this work.

- $a_{xpdl}.name = a_{plan}.name$
- $a_{xpdl}.performer = a_{plan}.metadata.Performer$
- $a_{xpdl}.participant = a_{plan}.metadata.Actor$
- $a_{xpdl}.description = a_{plan}.metadata.Description$
- $a_{xpdl}.startmode = a_{plan}.metadata.Type$
- $a_{xpdl}.deadline = a_{plan}.duration$
- Start and end dates of a_{plan} are translated as XPDL *extended attributes* and added to the properties of a_{xpdl} `<ExtendedAttribute name=`*start* `value=`$start_{aplan}$`>` `<ExtendedAttribute name=`*end* `value=`end_{aplan}`>`
- Every paramenter p_i of a_{plan} is translated as a XPDL *extended attribute* of the form `<ExtendedAttribute name=`$p_i.name$ `value=`$p_i.value$`>`

3. **Generation of XPDL Transitions.** This step contains two stages:

(a) For each a_{plan} generate $SUCC(a_{plan})$ as the set of immediate succesors of a_{plan}. A task j is an immediate succesor of another task i when a dependence $i < j$ exists and there is no task k such that $i < k$ and $k < j$.

(b) For every a_{plan} and for each b in $SUCC(a_{plan})$, generate an XPDL transition `<Transition From=`a_{plan} `To=`b`>`

As said before, the XPDL so generated is given as input to the BPM console that deploys it upon user request. The information contained in the XPDL file is used by both, the console in order to show task items required by the user, and the runtime engine in order to execute tasks according to the information stored in the XPDL process. Thus, an XPDL file contains *informative-only* fields which are the following: *name, participant, description, parameters, and estimated start and end times*. These fields are usually showed in the console for user information purposes. *Operative fields* needed by the execution model are: *startmode* (used to determine whether a tasks starts automatically or upon user request), *deadline* (used to manage whether a task has finished correctly on time), *performer* (used to determine if the user is allowed to execute a given actions) and the *transitions* (used to determine the execution order of tasks).

The life-cycle of Smart Process Management

The translation process above introduced is the keystone of a process that allows to achieve a full connection between the output of an AI Planning system and the input of a BPM runtime engine. The convergence of both technologies leads into an integrated environment for *Smart Process Management*, providing support for *modeling* (based on the representation of Hierarchical Task Networks), *adaptive generation* (based on Planning and Scheduling process) and *execution* (based on BPM runtime engines and consoles) of Smart Processes.

Authors argue that this can be considered a contribution in the field of Knowledge Engineering for Planning due to the following reasons: first, it is a non-trivial transformation between a plan representation and a widely spread, standard language for business processes. Second, it also answers some questions about new information requirements that must be covered by domains and plans representations when they have to deal with plans that must be executed by using standard BPM technologies. The new pieces of knowledge like type of actions, actors and performers, are not usually considered as part of a planning model, but it has been shown that they necessarily have to be incorporated in domain and plan representations.

Furthermore, the field of Knowledge Engineering for Planning also deals with the study and development of techniques and methodologies that might advance the life-cycle for engineering planning systems. In this sense, the transformation process above described bridges a common, important gap that prevented to adequately carry out a fast prototyping strategy for developing practical planning applications since, in order to develop a first prototype, it was mandatory to develop also ad-hoc execution monitoring processes and underlying preliminary interfaces, in order to convince users about the advantages of the system. Therefore, under this circumstances, building a first prototype of planning application is very costly in time and human resources.

As opposite, we have carried out a proof of concept based on the translation of a treatment plan previously generated by the planner and its execution based on the console. The plans obtained are then transformed into XPDL processes, following the translation process above described and interactively executed by oncologists on a BPM console. Therefore, the development effort in building a first prototype for oncologists has been reduced. Instead of fully developing both a specific interface for user interaction and a execution monitoring system to support the execution, we have studied and used the configuration techniques provided in the user manuals of both the console and runtime engine. The result, from the oncologist point of view, is a web application, obtained in few weeks, that provides both information about the treatment tasks to be performed, their time constraints and interaction to start/finish/suspend/abort tasks.

Figure 3 shows the configured interface which has as default behaviour to show information about the pending treatment tasks to be executed. The information in the console is structured as a table where each row shows task information that may be divided into three blocks:

1. Information about its name, its parameters and its estimated start and end dates (recall that XPDL does not provide specific fields to represent task parameters, nor start and end dates, but this information is extracted from *extended attributes* as detailed above).
2. Information about its state, shown in the form of flags.
3. Active buttons to control the execution of tasks. These controls are easily configurable, and include buttons to start, finish and suspend a task.

As the execution of the process progress, new pending task are added to the console. In addition, a basic time management at execution can be achieved, based on the capability of representing deadlines for tasks, as explained above.

Oncologists are highly skilled knowledge workers, but it is understandable the they have not a clear idea about what are their real usability needs with respect to a challenging and novel Clinical Decision Support System. Therefore the basic functionality above described is enough to capture user requirements, on the basis of this prototype, that would be almost impossible to detect on interviews-based knowledge/requirements acquisition.

Apart from Knowledge Engineering for Planning, the connection between a planner and a BPM runtime engine through the translation process introduced has advantages in the field of BPM. Mainly, it contributes to leverage the BPM life cycle incorporating capabilities for dynamic generation of emergent processes. As said in the introduction of this paper, BPM engines are oriented to execute processes the execution flow of which is completely defined a priori, and there is no place to the management of adaptive, context dependent process generation. The experimental proof of concept carried out shows that, starting from a "smart process model" represented by an HTN planning domain, it is possible generate processes (originally plans and then translated into business processes) that vary on its execution flow depending on a variable context, defined by a patient profile. Then, these processes can be executed on the basis of standard BPM execution models.

Related work

The relationships between workflows (or business processes) and AI planning have been studied from different perspectives, but it is relevant for this work when considered as a technique to directly generate workflows executed by ad-hoc, application specific systems devoted to the interactive execution and monitoring of plans considered as workflows. For this last case, some works are oriented to autonomic computing (Srivastava, Vanhatalo, and Koehler 2005), others oriented to grid computing(Deelman et al. 2004), and we can also find works devoted to support knowledge workers in their daily work, in the form of intelligent task management assistants(Myers et al. 2007). Again, these approaches do not face the execution of plans, seen as smart processes, by using standard BPM technologies. AI P&S techniques have also been applied in the field of worflow generation for semantic web services. In this application area, AI planning techniques are mainly focused on the automated generation of sequences of semantic web services calls (that may be seen as semantic business processes)(S.A., T.C., and H. 2001). Some approaches in this field (P. and M. 2004) address the generation of BPEL code from plans representing web services processes. These approaches are focused on the management of System-Centric processes and, due to the nature of these processes, do not address the interactive execution of processes.

Regarding the concrete BPM field, the concepts described in this work are subject of study under the denomination of *Adaptive Case Management*(WorkflowManagement-Coalition 2010). Nowadays this is an emergent concept in BPM. It tries to analyze and explore either already developed or new promising techniques, susceptible to be integrated into present BPM systems, in order to fulfill the requirements imposed by knowledge worker processes.

Conclusions

This work should be considered as a step forward in the pursuit of a methodology for rapid prototyping in Knowledge

Engineering for Planning. The main contribution consists on the integration of AIP&S techniques and BPM technologies through a process that translates plans into executable business processes, thus allowing to directly execute those plans into BPM standard runtime engines. From the point of view of a BPM runtime engine, the information model of plans, represented in XML, contains enough information to be directly executed what allows a fully automated translation process.

With respect to the flexibility of this approach, it is important to note that, although most of the knowledge embodied by plans obtained by any state-of-art planner can be reused by BPM runtime engines, without the transformation process presented, the plans obtained could not be directly executed in BPM engines. Therefore, for any planning system, it is mandatory to transform both, the structure of the plans and their content, as already explained. In this sense, whenever the plans obtained by another state-of-art planner fits to the plan representation here presented, it will always be possible to use our translation process in order to transform the plans into XPDL and then execute them on a BPM runtime engine. However, it is necessary to say that the domain model must be able to represent "special tags" for actions, in order to finally obtain plans containing all the information required for execution. This technique is not new in planning, and many planners, specially HTN planners (Myers et al. 2007), allow to introduce additional knowledge in the model of actions for postprocessing purposes. Under these considerations, authors argue that it is possible obtain a fully automated process that leads from domain modeling to human-centric business processes execution.

However the approach here introduced presents some weak points that need to be deeply studied. The translation of the planning domain object model is not completely addressed, and it must be faced in order to achieve a fully automated translation process. Precondition and effects management is neither addressed. Although it is possible to achieve a user acceptable execution of plans, this functionality is only permitted for prototype-level versions. The development of a full application requires to develop a complete execution monitoring based on the causal rationale of plans. Finally, a full treatment of temporal constraints at execution time is needed. All these issues are being faced at present and will be incrementally added to the current approach, according to the user requirements analyzed on the basis of this first prototype.

Acknowledgements

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

References

Bresina, J. L.; Jonsson, A. K.; Morris, P.; and Rajan, K. 2005. Activity planning for the mars exploration rovers. In *Proceedings of the ICAPS05*, 40–49.

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

Castillo, L.; Fdez-Olivares, J.; Garca-Prez, O.; Garzón, T.; and Palao, F. 2007. Reducing the impact of ai planning on end users. In *ICAPS 2007, Workshop on Moving Planning and Scheduling Systems into the Real World*, 40–49.

Dayal, U.; Hsu, M.; and Ladin, R. 2001. Business process coordination: State of the art, trends, and open issues. In *Proceedings of the 27th VLDB Conference*.

Deelman, E.; Blythe, J.; Gil, Y.; Kesselman, C.; Mehta, G.; Vahi, K.; Blackburn, K.; Lazzarini, A.; Arbree, A.; Cavanaugh, R.; and Koranda, S. 2004. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing* 1:25–39.

Fdez-Olivares, J.; Castillo, L.; García-Pérez, O.; and Palao, F. 2006. Bringing users and planning technology together. Experiences in SIADEX. In *Proceedings ICAPS06*, 11–20.

Fdez-Olivares, J.; Castillo, L.; Cozar, J.; and Garcia-Perez, O. 2010. Supporting clinical processes and decisions by hierarchical planning and scheduling. *Computational Intelligence* To Appear.

Fdez-Olivares, J.; Czar, J.; and Castillo, L. 2009. *Knowledge Management for Health Care Procedures*, volume 5626 of *Lecture Notes on Computer Science*. Springer. chapter OncoTherapist: Clinical Decision Support for Oncology Therapy Planning Based on Temporal Hierarchical Tasks Networks, 25–41.

Gagné, D., and Trudel, A. 2009. "Time-BPMN". In *Proceedings of 1st International Workshop on BPMN*.

Myers, K.; Berry, P.; Blythe, J.; Conley, K.; Gervasio, M.; McGuinness, D.; Morley, D.; Pfeffer, A.; Pollack, M.; and Tambe, M. 2007. An intelligent personal assistant for task and time management. *AI Magazine* 28(2).

P., T., and M., P. 2004. Automated composition of semantic web services into executable processes. In *International Semantic Web Conference*.

S.A., M.; T.C., S.; and H., Z. 2001. Semantic web services. *IEEE Intelligent Systems* 2(16):46–53.

Sacerdoti, E. D. 1975. The nonlinear nature of plans. In *Proceedings of IJCAI 1975*, 206–214.

Srivastava, B.; Vanhatalo, J.; and Koehler, J. 2005. "Managing the Life Cycle of Plans". In *17th Innovative Applications of Artificial Intelligence Conference*, 1569–1575. AAAI Press.

Tate, A.; Drabble, B.; and Kirby, R. 1994. O-PLAN2: An open architecture for command, planning and control. In Zweben, M., and Fox, M., eds., *Intelligent scheduling*. Morgan Kaufmann.

Workflow management coalition. http://www.wfmc.org/.

Wilkins, D. E. 1990. Can AI planners solve practical problems? *Computational intelligence* 6:232–246.

WorkflowManagementCoalition. 2010. http://www.xpdl.org/nugen/p/adaptive-case-management/public.htm. Group on Adaptive Case Management.