Towards the Use of XPDL as Planning and Scheduling Modeling Tool: the Workflow Patterns Approach

No Author Given

No Institute Given

Abstract. Given the absence of simple tools for modeling Planning and Scheduling problems, as well as the inability of Business Process Management for the anticipated planning of processes, this paper presents a transformation from XPDL process model diagrams into a hierarchical extension of the PDDL planning language, using the concept of workflow patterns as base of the translation process. The proposed architecture is evaluated within a specific teamwork project management scenario: the allocation of human resources and web services for the development of online courses in an e-learning center.

1 Introduction

The integration of Planning and Scheduling (P&S) and Business Process Management (BPM) is a significant challenge for enterprise environments. On one hand, BPM tools are able to deal with goals specification, environmental analysis, design, implementation, enactment, monitoring and evaluation of business processes[7], and they are adquiring an increasing business value for the efficient and intelligent knowledge management on these organisations. However, they lack of power for the anticipated planning and decision making of organisational processes. On the other hand, automated P&S, defined as "the process of generating possible representations of future behaviour *prior to the use of such plans* to constrain or control that behaviour"[16], is a technology that, due to the absence of simple modeling tools, has not explode its great potential in enterprise environments. Therefore, a common framework including these technologies would be interesting from both points of view.

Introducing automated P&S for the management of a company, based on an initial workflow representation would contribute to cover these missing goals, as it provides support for the generation of action plans, whatever these actions are carried out by humans or by remote calls to web services[5], as well as support for decision-making on key issues like tasks organization and resources allocation. Furthermore, if we were able to automatically transform the information present in a process model (usually described by a BPM diagram created by business analysts) to its corresponding P&S representation, we would overcome the traditional obstacle for P&S technology: the use of complex languages to model the domain. This paper gives further insights into this integration process.

Prior work on the use of P&S for the automatic generation of business process models is exposed in [10] (using an augmentation of STRIPS representation). As opposite to this work, our approach shows how to use an existing process model diagram stored in a standard BPM language, to automatically generate P&S representations using the HTN[12] planning paradigm (detailed later in this paper). Afterwards, we use the output of this transformation as input of an intelligent planner in order to prepare action plans and resources allocation that, after the corresponding validation by a business manager, would be helpful for decision making on risk management[7], mainly to avoid unwanted situations, like the detection of high loads of work in a specific period, or the incapacity to complete a project before a deadline, which is specially useful in ad-hoc workflows with human intervention. Some authors worked in similar problems as coordinating the design of airplanes[1], or workflow coordination in a mobile environment[11].

So, the main contributions of this paper can be summarized as:

1. To stablish the cornerstone for a new way to model P&S problems using a standard business process modeling tool, overcoming the need of new ad-hoc tools or languages[15] as well as the need for staff training on P&S languages.
2. To introduce the concept of workflow patterns decomposition for the automatic transformation of BPM diagrams into HTN planning representations.
3. To show how to integrate P&S technology at a low cost within a BPM framework, to make the most of an existing process model for anticipated management planning and resources allocation.

The paper is structured as follows. Section 2 describe the real case the paper is focused on. Section 3 provides an overview of the technologies used. Section 4 details the architecture overview and the mapping algorithm. Section 5 expose some results and Section 6 contains a conclusion of the contribution made and future work.

2 E-learning Management Scenario

In order to expose the contributions mentioned before, the paper is centered around the process of creating an online course within a Learning Management System. This process implies the participation and the correct interaction of different roles (instructional designers, graphic designers, HTML developers, sysadmins, tutors, students, etc). The corresponding process model will be customised accordingly to the operation of an specific e-learning center. In this case we have been supported by the teamwork experience of CEVUG[1]. The resulting process model diagram can be observed at Figure 1.

Realise that some of the tasks needed to complete the process can only be done by a specific worker/role, and they can have some previous dependencies. For example, the task "content authoring" (A2) could only be completed by a content author and the task of "training authors on instructional design" (A1)

_____
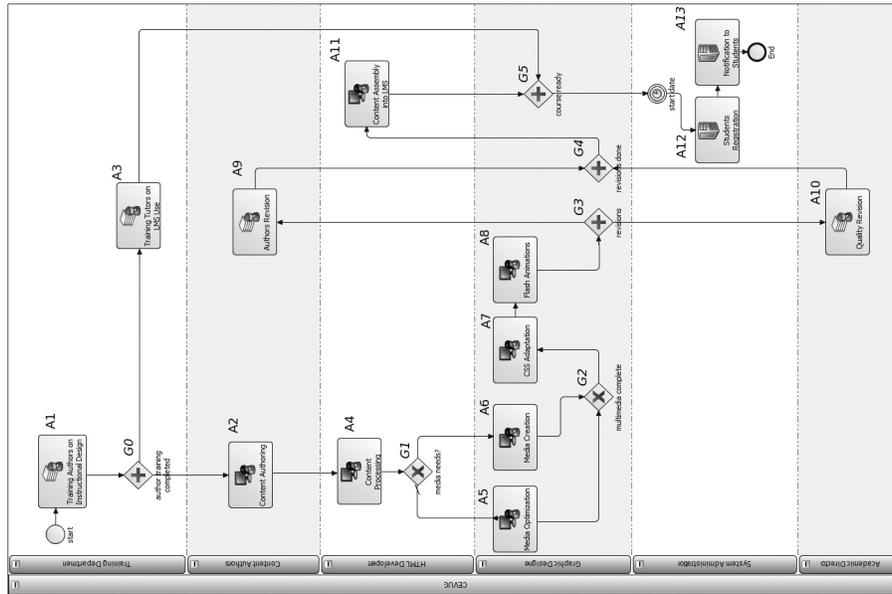[1] University of Granada E-learning Center, http://cevug.ugr.es

Fig. 1. BPM representation for "e-learning course creation" process (rotated).

had to be previously completed by the training department. So, the election of a specific worker for a specific task will be conditioned by his *ability* to complete it, and possibly by his temporal *availability* (or any implicit condition of the problem, i.e. the worker cannot work in more that n courses simultaneously).

Note that almost all the activities described can be considered as subflows themselves (i.e. they might be decomposed in sub-activities and so on recursively), so the subprocess diagram should be also specified for every of them. This decomposition-based representation of tasks make the HTN planning paradigm very appropriate in order to capture the model information on a BPM diagram, as better explained in section 4. Thus, the proposed scenario considers different workers that have to cooperate together to achieve a final goal.

For each **activity** we could identify:

- Duration: Estimated duration for the activity to be completed.
- Web Service: web service associated to the automated execution of the activity (in case it is needed).
- Dependencies and requirements: activities to be completed before the possible execution of the activity, as well as other requirements stablished (i.e. a specific worker ability).

For each **worker** we could identify:

- Abilities: a list of abilities to achieve the requirements mentioned before.
- Lane: The department or lane the worker belongs to.

- Number of courses: number of courses the worker is working on.
- Availability dates: dates in which the worker is available to be assigned a task.

This information will help us to establish the preconditions for the execution of actions, and to represent temporal knowledge associated to the problem (that might also be automatically generated[2], though not addressed in this paper). Anyway, the task planning notification to every worker has to be done after a validity check by the manager. The BPM language chosen in order to capture all the mentioned information is XPDL, which is detailed in the next section.

3 Technical Background

In this section XPDL and the concept of workflow patterns are defined. We also introduce the HTN planning paradigm, as well as some conceptual equivalences with Business Process Models.

3.1 XPDL and Workflow Patterns

The goal of XPDL[3] language is to store and exchange the process diagram. An XPDL file offers a one-to-one representation of the original Business Process Management Notation (BPMN)[8] process diagram. The main advantage of using XPDL as modeling language is that it is a common language used among business analysts, and it can be used to represent the organisation activity easily, storing it in an XML standard format. There are a lot of modeling tools that already incorporate XPDL natively or as an additional plug-in. Some of the tools evaluated store process models either using a propietary format or directly BPEL[8], but ideally this should be done in XPDL, as it was thought for modeling, not for execution[9]. We have used "TIBCO Bussiness Studio"[2], since it supports XPDL v2.0 and it is offered for free.

XPDL offers some standard `tags` to represent business processes. The definition of a `WorkflowProcess` consists of one or more `Activities` (also called Implementation Activities), each comprising 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 may be conditional (involving expressions which are evaluated to permit or inhibit the transition) or unconditional, and may result in the sequential or parallel operation of individual activities within the process. In graphical terms, a transition is a connection between two nodes. Special activities, referred to as `Route Activities`, are used to implement decisions that affect the sequence flow path through the process. An activity may also be a subflow (`ActivitySet`), being itself a container for the execution of a (separately specified) process. `Lanes` are used to organise and categorise activities, often used for specifying roles, departments, etc. XPDL assumes a number of standard `DataTypes` (string, integer, float,

---

[2] available at http://www.tibco.com

date, etc) that are relevant to `DataFields`, environmental or participant data, and that may be used to define expressions, to support conditional evaluations and assignment of new values to DataFields or `Parameters`. In case these tags are not expressive enough to represent all the information wanted, XPDL provides the `extendedAttribute` element, that can be appropiate in most cases. For further details about XPDL, refer to its specification[3].

On the other hand, Workflow Patterns[14] are those generic structures that represent frequently-used relationships between tasks in a process, and that are tipically nested to form the whole process model[4]. As clearly exposed in the next subsection, a workflow pattern decomposition of any XPDL process can be easily converted into an HTN domain definition. However, XPDL lacks of some power for the correct representation of some complex patterns[13]. Therefore, only the most basic ones has been studied in this paper, those that can be well represented and are expressive enough for the definition of most processes.

3.2 HTN Planning

The Hierarchical Task Network (HTN) planning paradigm was developed in order to express planning problems in a structured way, by means of the definition of compound tasks that are reduced into a network of lowest level activities, or primitive non-decomposable actions whose execution represents a state change. HTN planners use as input two different files:(a) the `problem`, which encodes the *initial state* (literals that are true at the beginning of the problem) and the *task-goal* (a partially ordered set of tasks that need to be carried out), and (b) the `domain`, which encodes the reduction schema (through the definitions of *tasks* and primitive *actions*) as well as a set of *predicates* and *constants* definitions.

HTN-PDDL Notation. The hierarchical extension of PDDL used[2] has the following notation:

- (`:types`), (`:constants`), (`:predicates`) and (`:functions`).
- (`:task`) to express compound tasks. Its definition can include *:parameters*, and different *(:methods* with associated *:preconditions* and *:tasks* that represents its corresponding lowest level task decomposition.
- (`:durative-action`) to express primitive actions, composed of *:parameters*, *:duration*, *:condition* and *:effects* caused by the execution of this action.
- (`:objects`) to define objects that are present in the problem.
- (`:init`) to define the set of literals that are initially true.
- (`:task-goals`) to define the set of high level tasks to achieve.

Therefore, the HTN paradigm is able to represent the hierarchical structure of the domain and it is also expressive enough to capture the expert knowledge in order to drive the planner to a desirable solution. We have used the HTNP[3] planner for this paper, as we already know how to translate workflow patterns for semantic web services composition[5], as well as its adaptation to temporal knowledge[2]. Therefore, plans obtained using the task representation described above could be interpreted as workflow instances.

---

[3] Formerly named SIADEX, refer to [2] for details.

4 Architecture Overview

A sketch of the proposed arquitecture can be observed in Figure 2. Firstly, we have to draw the model for a specific process of the company in close cooperation with its managers, specyfing all the needed information (in our case mentioned on Section 2) and storing it in XPDL. Afterwards, the basic workflow patterns present in that model are analysed, finding a decomposition tree for them (Fig. 3) that will be automatically mapped into an HTN-PDDL task network, taking into account the transformations mentioned in next subsection.
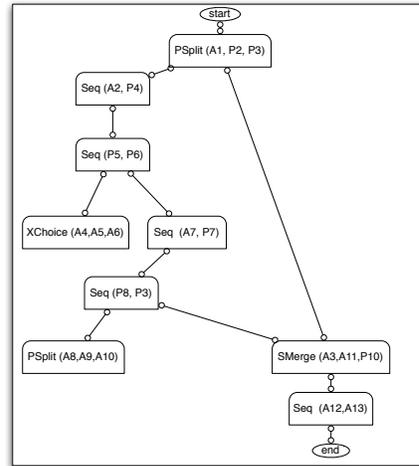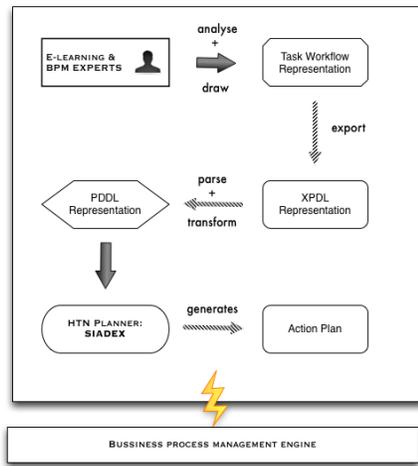

Fig. 2. Architecture for business anticipated management planning

Fig. 3. Workflow Patterns decomposition for the business process shown at Fig. 1

The generated code is used as input of the HTNP planner triggering a reasoning and search process that, guided by the knowledge included in the domain, finally returns a workers allocation and an action plan, to carry out the tasks defined as goal. Realise that we could obtain a solution according to input variables, temporal landmarks introduced in the diagram, or organisational information accesible through web services available at planning time. Finally, the resulting plans could also be converted again into BPEL, a language which is readable by most BPM engines, developed for the execution of processes and web services composition, and that has been subject of prior research related to our work[8]. This would help to complete the cycle, seamlessly introducing P&S technology within the BPM life cycle.

The underlying idea of the proposed architecture is that outcomes could be easily applied to different domains, developing a solution that maintains consistency and completeness for a wide subset of the possible inputs, as well as

readability of the generated code. That is one of the reasons for the prior work-flow patterns recognition phase[4], as well as the existing equivalences between both the decomposition tree and the HTN paradigm, detailed in next subsection.

4.1 XPDL to HTN-PDDL Mapping

What drives us to propose the use of XPDL to model P&S problems following the HTN paradigm, is the existing equivalence between both. The basis for this statement is the fact that most business processes modeled under XPDL can be decomposed as a set of related workflow patterns (defined in section 3.2). This set is what we call a *workflow patterns decomposition tree*. For example, the process diagram of Figure 1 can be expressed as the decomposition tree shown in Figure 3. As we can observe, the root node is a "Parallel Split" in which, after the execution of activity A1, the blocks P2 and P3 are executed in parallel. At the same time, block P2 is defined as `Sequence(A2, P4)` (left child node) and block P3 is defined as `SimpleMerge(A3,A11,P10)` (right child node) . Subsequently, every of the existing blocks, P_x, are recursively defined as a new decomposition, until we complete the whole diagram. This finally forms the decomposition tree shown at Figure 3. The mapping of this structure into an HTN-PDDL domain is intuitive, as exposed next.

Workflow activities as durative-actions. The execution of an activity in a workflow corresponds to the execution of a primitive action in an HTN-PDDL domain. XPDL Activities should also be mapped as HTN-PDDL constants, used as parameters of a predicate (`completed ?a -activity`), that indicates the completion of this activity upon its execution (defined as an :effect), and that can also be used in preconditions statements for the execution of other tasks or primitive actions. XPDL Participants and Lanes will be mapped as objects, making possible the allocation of participants to activities through the inclusion of a parameter (`?p - participant`) in the durative-action definition. Furthermore, the membership of a Participant P_x to a Lane L_y will be mapped as an init condition of the problem, using the previously defined predicate (`belongstolane` P_x L_y). This predicate can also be used as precondition (i.e. the activity A3 can only be done by participants that belongs to the training department). The duration of an activity is modeled using the XPDL *extendedAttribute* tag.

Workflow patterns and subprocesses as composed tasks. A workflow pattern present in the XPDL will be mapped as an HTN-PDDL task (Table 1 shows the translation for the patterns considered). Furthermore, XPDL ActivitySets (subprocesses embedded into an XPDL description) will also be mapped as compound tasks. This is where really resides the power for decision making, considering the control structs that define the execution order and control flow between tasks. To do so, HTN-PDDL use the symbols `()` to express *sequencing* (see Table 1(a)), and the symbols `[]` to express *paralellism* (see Table 1(b)).

We will also map XPDL DataTypes as custom HTN-PDDL types, and XPDL Parameters and DataFields can be used in preconditions statements for the definition of conditional workflow patterns (see Table 1(c)), to define the control
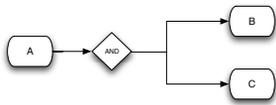
Sequence. *"A task in a process is enabled after the completion of a preceding task in the same process"*

```
(:task SeqA2P4
    :parameters()
        (:method A2P4
         :precondition ()
         :tasks ((A2 ?w1) (BlockP4))))
```


(a)

Parallel Split. *"The divergence of a branch into two or more parallel branches each of which execute concurrently."*

```
:task PSplitG3
 :parameters ()
        (:method G3
         :precondition (completed a7)
         :tasks ((A8 ?w1)
                 [(A9 ?w2) (A10 ?w3)])))
```


(b)

Exclusive Choice. *"The divergence of a branch into two or more branches such that when the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on a mechanism that can select one of the outgoing branches."*
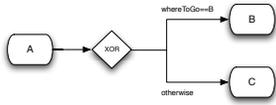
```
(:task XChoiceG1
 :parameters (?x - object)
    (:method ifA5
     :precondition (condition ?x)
     :tasks ((A5 ?w1) (SeqA7A8)))

    (:method elseA6
     :precondition (not
                    (condition ?x))
     :tasks ((A6 ?w2) (SeqA7A8))))
```


(c)

Simple Merge. *"The convergence of two or more branches into a single subsequent branch such that each enablement of an incoming branch results in the thread of control being passed to the subsequent branch."*
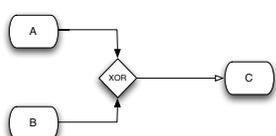
```
(:task SMergeG5
 :parameters ()
    (:method SMG5
     :precondition ()
     :tasks ([(A3 ?w1) (A11 ?w2)]
             (A12 ?w3))))
```


(d)

Table 1. Definition for the basic workflow patterns on the left side. A corresponding HTN-PDDL representation on the right side, using data from some of the patterns found on Figure 1. Note in (c) that a workflow pattern as XChoiceG1 is decomposed using two methods, executing a primitive task and then the next worfklow pattern found (SeqA7A8).

flow between two activities. Finally, the task goal for the planning problem correspond to the root node of the workflow patterns decomposition tree.

5 Results

We have used the process model shown at Figure 1 for the experimentation results. We achieved to transform this model into HTN-PDDL domain and problem files, using the workflow patterns decomposition shown at Figure 3, the transformations proposed at Table 1, and estimated durations for the activities.

The following action plan (that could also be generated as a gantt diagram) was returned by the planner, being $(A_x\ P_y)$ the corresponding allocation of activity A_x to participant P_y, and "Day1 Month1-Day2 Month2" the timeframe where that activity should be executed:

```
(A1 Emilio) 1st Jun-10th Jun  -> (A2 Storre) 10th Jun-15th Jun ->
(A4 Chesco) 15th Jun-20th Jun -> (A5 JoseBa) 20th Jun-22nd Jun ->
(A7 JoseBa) 22nd Jun-25th Jun -> (A8 JoseBa) 25th Jun-28th Jun ->
(A9 Storre) 28th Jun-1st Jul  -> (A10 FMoren) 1st Jul-3rd Jul  ->
(A3 Emilio) 10th Jun-8th Jul  -> (A11 Chesco) 8th Jul-14th Jul ->
(A12 Artur) 14th Jun-15th Jul -> (A13 Artur) 15th Jul-16th Jul ->
```

Results were evaluated by the manager of the e-learning center, outperforming the expected. However, some improvements are still necessary for the implementation of temporal restrictions[2], stressing on the translation of sinchronization/merge route gateways.

6 Conclusions and Future Work

Designing P&S domains manually through the use of planning languages as PDDL[6] is a very difficult task, even for engineers with a prior training on these languages. This obstacle has been a traditional drawback for the introduction of P&S techniques in enterprise environments, where new technologies are often evaluated in terms of time, money and efforts needed. Successful technologies are usually based on nifty user interfaces that achieve to give easiness for obtaining a specific goal. Automated P&S definitely lacks of this kind of tools.

Therefore, the main goal of this paper has been to stablish the cornerstone for a new way to model P&S problems using standard BPM tools, that otherwise have demonstrated to be one of the most promising tools for business analysts, and that, furthermore, already consider the significance of web services for a new age of interconnected organisations. A transformation from the XPDL language is proposed, through the decomposition of the input diagram into workflow patterns that are directly mapped into an HTN-PDDL domain, that will be input of an intelligent planner. Thus, action plans and resources allocation can then be generated in an automated way through the use of simple modeling tools.

Future work will explore the HTN-PDDL representation of complex workflow patterns and will also study how to represent more demanding temporal restrictions that are usually present in business processes, so that we increase the capacity of the business analyst to represent real planning domains through the use of the same business modeling tools.

References

1. A. Bahrami. "Integrated Process Management: From Planning to Work Execution". In *IEEE Workshop on Business Service Networks*, pages 11–18, 2005.
2. L. Castillo, J. Fdez-Olivares, O. García-Pérez, and F. Palao. "Efficiently handling temporal knowledge in an HTN planner". In *16th ICAPS Conf.*, pages 63–72, 2006.
3. Workflow Management Coalition. "XML Process Definition Language Specification, v2.1". *WFMC-TC-1025*, pages 1–216, 2008.
4. T. Dirgahayu, D. A. C. Quartel, and M. J. van Sinderen. "Development of transformations from business process models to implementations by reuse". In *3rd International Workshop on Model-Driven Enterprise Information Systems*, pages 41–50, 2007.
5. J.Fdez-Olivares, T. Garzón, L. Castillo, O.García-Pérez, and F. Palao. "A Middleware for the automated composition and invocation of semantic web services based on HTN planning techniques". In *LNAI 4788*, pages 70–79. Springer, 2007.
6. D. Long and M. Fox. "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains". *Journal of Artificial Intelligence Research*, 20:61–124, 2003.
7. M.Z. Muehlen and D. Ting-Yi Ho. "Risk Management in the BPM Lifecycle". In *LNCS 3812*, pages 454–466. Springer, 2006.
8. C. Ouyang, M. Dumas, A. Hofstede, and W.M.P. van der Aalst. "Pattern-based translation of BPMN process models to BPEL web services". *International Journal of Web Services Research*, 5(1):42–62, 2007.
9. N. Palmer. "Workflow and BPM in 2007:Bussiness Process Standards see a new global imperative". pages 9–14, 2007.
10. M.D. R-Moreno, D. Borrajo, A. Cesta, and A. Oddi. "Integrating planning and scheduling in workflow domains". *Expert Systems with Applications*, 33(2):389–406, 2007.
11. R. Sen, G. Hackmann, M. Haitjema, G.C. Roman, and C. Gill. "Coordinating Workflow Allocation and Execution in Mobile Environments". In *Coordination Models and Languages, LNCS 4467*, pages 249–267. Springer, 2007.
12. E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. "HTN planning for web services composition using shop2". *Journal of Web Semantics*, 1(4), 2004.
13. W.M.P. van der Aalst. "Patterns and XPDL: A critical Evaluation of the XML Process Definition Language". *QUT report FIT-TR-2003-06*, pages 1–30, 2003.
14. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. "Workflow Patterns". *Distributed and Parallel Databases*, 14(1):5–51, 2003.
15. T.S. Vaquero, V.M.C. Romero, F. Tonidandel, and J.R. Silva. "itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains". In *17th ICAPS Conf.*, 2007.
16. R. Wilson and F.C. Keil. "*MIT Encyclopedia of Cognitive Science*", pages 652–654. MIT Press, 2001.