

Triggering Replanning in an Integrated Workflow Planning and Enactment System

Hilmar Schuschel and Mathias Weske

Hasso-Plattner-Institute at the University of Potsdam,
Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam, Germany,
{Schuschel,Weske}@hpi.uni-potsdam.de

Abstract. Workflow management systems support and automate the enactment of business processes. For this purpose, workflow management systems use process definitions that have been manually planned and modeled at build time. Recent research approaches try to enhance this concept by automating the creation of process definitions, using planning algorithms. This avoids the need for predefined process definitions and thus increases flexibility and allows to save costs. An important aspect of flexibility is the ability to react to unanticipated events that might occur during runtime. This Reaction can imply replanning and dynamically adapting the process. This paper shows how replanning can be triggered automatically in an integrated workflow planning and enactment system. Triggers for monitoring process executions are presented and events are defined which lead to the evaluation of corresponding conditions for deciding when replanning is necessary. Finally, advantages, limitations and areas of application of this approach are discussed.

1 Introduction

In today's dynamic markets, organizations have to take advantage of information technology to improve their business and stay competitive. A crucial point for the competitiveness of organizations is the performance of their processes [7, 22]. Workflow management systems [6, 9, 10] are applied to support and automate process enactment. For this purpose, process definitions are used that have been manually planned and modeled at build time. Recent research approaches propagate the feature to plan process definitions automatically to improve quality and flexibility [1, 11, 12, 18, 25]. An important aspect of flexibility is the ability to react to unanticipated events that might occur during runtime, by replanning and dynamically adapting the process. This paper shows how replanning can be triggered automatically in an integrated workflow planning and enactment system. Triggers for monitoring process executions are presented. Therefore, events are defined which lead to the evaluation of corresponding conditions for deciding when replanning is necessary.

A *process* is a defined set of partially ordered steps intended to reach a goal [4]. It is the means to change a given situation in order to fulfill a company's goal. The information about this situation including all relevant documents is

called a *case*. Examples of cases are an incoming purchase order that has to be handled or a sick patient who has to be cured. From an organizational point of view, the life cycle of a process includes the phases planning and enactment. *Planning* is the generation of a description called a *process definition* of what has to be done in which order to reach a particular goal. The subsequent phase is *enactment*, which is the organizational task to schedule the work in accordance to the process definition. Workflow management systems take a process definition as input and use it to support and automate process enactment. Traditionally, planning and supplying the process definition to the workflow management system has to be done manually. Recently, advances in the automation of business process planning has been made. Automated planning allows to generate individual process definitions for every case. Thus, quality and flexibility can be improved. An important aspect of flexibility is the ability to react to unanticipated events that might occur during runtime, by generating a new process definition and dynamically adapting the process. Basically, there are three main steps in replanning:

1. Trigger replanning when necessary
2. Generate a new process definition
3. Adapt process enactment to the changed process definition

To fully automate replanning, all three steps have to be taken into account. While research in workflow management concentrates on step 3 to enhance flexibility, the integration of automated process planning allows to fully support replanning.

This paper is based on the integrated planning and enactment system presented in [18] and adds a concept for automated initiation of replanning. Triggers are specified to monitor process executions and start replanning if necessary. In principle, replanning is necessary, if the process definition assigned to a case is no longer adequate for further enactment. This paper defines two degrees of adequateness and identifies events that may threaten this adequateness. To each event a condition is assigned, specifying the exact circumstances under which the event makes replanning necessary.

Related work is presented in Section 2. Section 3 introduces the relevant foundations on AI planning algorithms and workflow management. Section 4 describes the overall workflow planning and enactment system and adds a concept for automatically triggering replanning. Finally, in Section 5 areas of application, and unsolved problems are discussed.

2 Related Work

There are two major areas of related work: research on dynamic adaption of workflows and approaches to automatically generate workflow process definitions.

Dynamic adaption of workflows deals with adjusting running process instances to changes in the process definition [3, 15, 17, 8]. Nevertheless, all this

work is concentrated on step 3 of replanning – the adaption of process enactment – as described in Section 1. Triggering replanning and generating a new process definition is expected to be done manually.

Recent research approaches present options to automate the generation of processes definitions: For instance, in [1] ontologies of domain services and domain integration knowledge are used that serve as a model for workflow integration rules. *DYflow* [25] avoids the use of predefined process definitions and allows the dynamic composition of Web services to business processes by applying backward-chain, forward-chain and data flow inference. Another approach dealing with automated composition of web-services is described in [14]. Semantics for a subset of DAML-S [21] are defined in terms of a first-order logical language, to enable automated planning. In [11] the composition of single tasks or subgraphs of workflows is embedded in a case-based framework for workflow model management. In [12] the application of contingent planners to existing workflow management systems is discussed. To generate process definitions some of these approaches use Artificial Intelligence (AI) planning algorithms, while others use proprietary developed algorithms. Although AI planning algorithms have been in research for more than 30 years [5], advances in recent years make their application on real business domains promising [13, 23]. The issue of replanning is discussed by some of these approaches, but step 1 of replanning – triggering replanning – is not explicitly taken into account.

3 Preliminaries

There has already been done a lot of research on planning and workflow enactment in the areas of AI planning algorithms and workflow management systems respectively. This section presents the concepts from both research areas that are important for an integration of workflow planning and enactment. In the remainder of this paper, common concepts of both areas are described using one continuous terminology, instead of applying the area specific terminology at each case. The example introduced in this section will be used throughout the paper to illustrate the integration to an overall system.

3.1 Planning Algorithms

In this section, foundations on AI planning are presented which are relevant for triggering replanning in an integrated workflow planning and enactment system. Furthermore, an example is introduced that is used throughout the paper.

Planning algorithms [5, 24] take a description of the current state of a case, a goal, and a set of activity definitions as input. The *goal* constitutes the desired state of the case and where required a metric to optimize. The means to transform the case from its actual state to the desired state are the activities. An *activity* is a piece of work that forms one logical step within a process. An *activity definition* defines the conditions under which an activity can be executed, called *preconditions* and its impacts on the state of the case, called *effects*:

Definition An *activity definition* d consists of

- a set $prec_d$ of preconditions
- a set eff_d of effects

A *domain* is a set of activities definitions. To sum up, the input of a planner is a *planning problem* defined as follows:

Definition A *planning problem* P consists of

- a domain dom_P
- an initial state $init_P$
- a goal $goal_P$

The initial state and the goal are logical descriptions of the state of the case. *Planning* is the task of finding a partially ordered set of activities that, when executed, transforms the case from its current state to the goal. The description of this set of activities and their ordering – the *process definition* – is the output of the planning algorithm. To illustrate the basic procedure of a planning algorithm and the relationship between its input and output, the planning of a simple process is presented. The example domain D consists of five activity definitions d_1 to d_5 that are described using the Planning Domain Definition Language [16] (PDDL). The corresponding definition given in extracts:

```
(define (domain D)
...
  (:action d1
    :precondition (and (v) (t))
    :effect (and (y) (increase (costs) 50)))

  (:action d2
    :precondition (x)
    :effect (and (z) (u) (increase (costs) 20)))

  (:action d3
    :precondition (x)
    :effect (and (v) (t) (not (z)) (increase (costs) 30)))

  (:action d4
    :precondition (z)
    :effect (and (w) (increase (costs) 10)))

  (:action d5
    :precondition (x)
    :effect (and (w) (increase (costs) 40)))

  (:action d6
    :precondition (x)
    :effect (and (t) (increase (costs) 10)))
)
```

Activity definition d_1 has the precondition v and the effect y . For instance, v is the fact that a customer name is determined and y is the fact that the address of the customer is determined. Thus, activity definition d_1 determines the address of a customer based on his name. Besides these functional properties, non-functional properties are specified: The costs for the execution of d_1 are 50. To define a concrete planning problem, not only the domain has to be specified, but also the initial state and the goal. The corresponding PDDL definition given in extracts:

```
(define (problem P)
  (:domain D)
  ...
  (:init (x) (= (costs) 0))
  (:goal (and (w) (y)))
  (:metric minimize (costs))
)
```

The planning problem P is specified as follows: The initial state is $x \wedge (costs = 0)$ and the goal is $w \wedge y$ with the costs to minimize. To complete the input for a planning algorithm, two extra activity definitions are introduced: an activity definition *Start* without preconditions and whose effect is the initial state of the case and an activity definition *Finish* with the goal as a precondition and no effects. The task of a planning algorithm is to find a process definition that solves P . Process definition p_1 depicted in Fig. 1 is a solution for P . The activities are partially ordered in a way that the effects of preceding activities satisfy the preconditions of subsequent activities. For example, d_2 has the effect z that is needed by d_4 as a precondition. Such a link between the effect of one activity that satisfies the precondition of another activity is expressed by a *causal link*, depicted as a dashed arrow. Every causal link implies an *ordering constraint*. For example, the ordering constraint from d_3 to d_1 is derived from the causal link between these activities. Additionally, ordering constraints are also inserted to protect causal links. For example, the ordering constraint from d_3 to d_2 is inserted to protect the causal link from d_2 to d_4 . If there would be no ordering constraint from d_3 to d_2 , d_3 could be executed between d_2 and d_4 . This would be problematic, because d_3 negates the effect z of d_2 that is needed by d_4 as a precondition.

Process definition p_1 is a solution of P . The minimized, overall costs are 110. There exists another process definition p_2 depicted in Fig. 1 that also produces the facts w and y , if started in an initial state in which fact x holds. Since the overall costs of p_2 are not minimal, it is no solution of P .

3.2 Workflow Management

The purpose of workflow management systems [6, 9, 10, 19] is to support and automate the enactment of business processes. During workflow enactment pieces of work have to be passed to the right participant at the right time with the support of the right tool. A central property of a workflow management system is

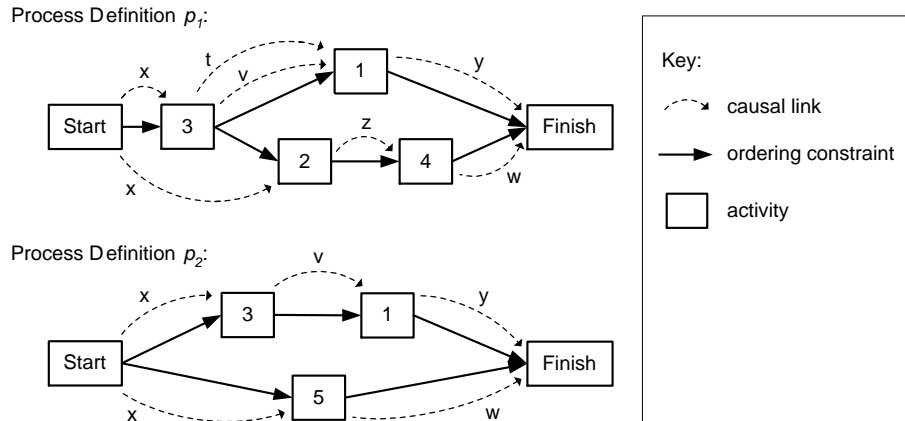


Fig. 1. Process Definitions p_1 and p_2

that this functionality is not hard coded for a specific process, but implemented in a way that the system can take any process definition as input. For a workflow management system, the relationship between effects and preconditions as the causal origin of the ordering of the activities is not relevant. Therefore, a process definition as input for a workflow management system contains only ordering constraints and no information on preconditions, effects nor causal links. Next to the process definition, the workflow management system needs information on the potential participants of the process and the available tools.

Before the execution of a process, all activities are in the state *init*. The execution of a process starts with the activities in the process definition that have no predecessors. For example in process definition p_1 depicted in Fig. 1 activity 3 is scheduled first. When an activity is executed the workflow management systems automatically starts the appropriate tool with the data that has to be processed in this activity. By starting an activity its state changes to *running*. After execution completes, the state of an activity changes to *done* and its subsequent activities are started. This iterates until all activities are in the state done and the process is completed.

4 Replanning in an Integrated Planning and Enactment System

This section is divided into three subsections: In Section 4.1, the overall framework for an integrated workflow planning and enactment system is presented. Section 4.2 carries on by adding a concept for replanning. Finally, Section 4.3 describes triggers for replanning in detail.

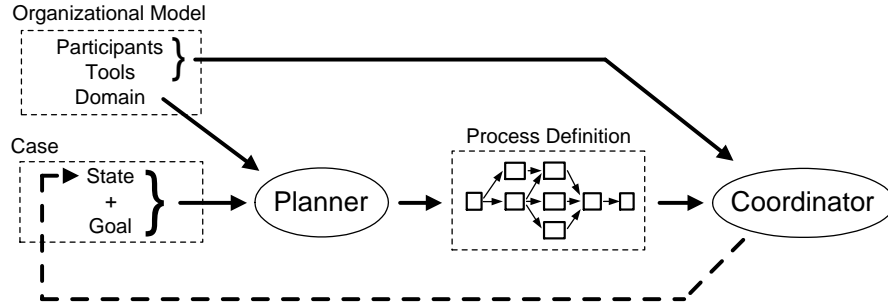


Fig. 2. Integrated Planner and Coordinator

4.1 Framework

The two central sub-systems of the framework are the planner and the coordinator. The planner generates a process definition for each case. The coordinator then uses the process definition to schedule the activities. In the following, the basic functional and behavioral aspects of the interaction of these components are described. Fig. 2 gives an overview of the interrelation of input and output between planner and coordinator. The basis for the overall system is the formal description of the domain, the participants and the tools. They describe the means of the company to process individual cases. A case is defined as follows:

Definition A case c consists of

- a state $state_c$
- a goal $goal_c$

To plan a process definition to handle a case c , a corresponding problem definition P has to be specified. This is done by mapping $state_c$ to $init_P$ and $goal_c$ to $goal_P$. The domain of the problem definition is the domain of the organizational model. The planner takes the problem definition as input and generates a process definition as output. A process definition is defined as follows:

Definition A process definition p consists of

- a set $inst_p$ of activity instances
- a set $conn_p \subseteq inst_p \times inst_p$ of control connectors

Definition An activity instance i based on activity definition d consists of

- an execution state $exec_i \in \{\text{init, running, done}\}$
- a set act_i of actual effects
- a set $rel_i \subseteq eff_d$ of relevant effects

An activity instance is the representation of an activity definition in a process definition. In addition to the information in its corresponding activity definition, an activity instance contains information on its execution state and its effects. This information is important for triggering replanning as it is explained in Section 4.3. The limitation to three different execution states of activity instances is a strong simplification for the purpose of this paper. The set act_i contains the effects the activity actually had during execution. The set rel_i contains the effects of eff_d that are relevant for the process. An effect $e \in eff_d$ is relevant iff, ceteris paribus, the planner would generate another process definition, if $e \notin eff_d$. The relevance of an effect in eff_d depends on the problem definition. Consider the example introduced in Section 3.1. The activity definition d_2 has the effects $z \wedge u \wedge \text{increase costs by } 20$. The effect u is not relevant for p_1 , because the same process definition would be generated, no matter if d_2 has the effect u or not. Thus, the relevant effects rel_2 are $z \wedge \text{increase costs by } 20$.

After the process definition is planned, it is assigned to the case. The process definition combined with information on participants and tools, is the input of the coordinator. This allows the coordinator to schedule the execution of the activities in accordance to the process definition until the goal is reached. The dashed arrow from the coordinator to the state of the case depicted in Fig. 2 illustrates the effects of the activities on the case, which are scheduled by the coordinator.

To illustrate the behavior of the overall system the example from Section 3.1 is carried on. Let us assume a case c should be handled that is specified as follows: $state_c$ is x and $goal_c$ is $w \wedge y$ with the costs to optimize. First, a corresponding problem definition P has to be specified. Therefore, $state_c$ is mapped to $init_P$ and $goal_c$ is mapped to $goal_P$. A solution for P is the process definition p_1 depicted in Fig. 1. The rectangles stand for the activity instances i_1 to i_4 that represent the activity definitions d_1 to d_4 in p_1 . All activity instances are in the state *init*. p_1 is given to the coordinator. Execution begins by starting i_3 . While running, i_3 has the anticipated effects $v \wedge t \wedge \neg z$. After i_3 completes, the subsequent activity instances i_1 and i_2 are started. The execution continues until all activity instances are in the state *done* and the goal is reached.

4.2 Replanning

An integrated planning and enactment system allows to automatically replan process definitions if necessary and adapt the process enactment. Therefore, the interaction between planner and coordinator becomes more interlaced. Consider the example above. Let us assume that i_3 completes without having all anticipated effects. For example it has the effects v and $\neg z$ but not the effect t . In this case replanning becomes necessary, because a precondition of the subsequent activity instance i_1 is not satisfied. The coordinator that monitors the execution of the process, notes the missing effect and triggers replanning. Replanning is done by planning a new process definition based on the current state of the case. In the example, the current state of the case $state_c$ is $x \wedge v \wedge \neg z$. A new planning problem P' is defined and the planner generates a process definition p_3 shown in

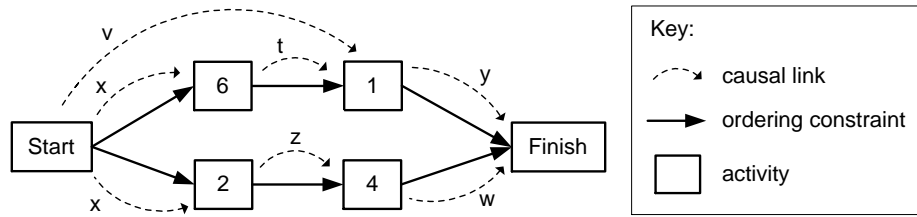


Fig. 3. Process Definition p_3

Fig. 3 i_3 is not part of p_3 , because the effect v is not needed any more, and the missing effect t can also be produced by i_6 at a lower cost. p_3 is assigned to the case and given to the coordinator, which then adapts the process enactment to the new process definition. In the example above replanning became necessary, because an activity instance completed and an anticipated effect was missing. Next to anticipated effects that are missing, other events can occur that make replanning necessary. In the following, a general approach to trigger replanning automatically is presented.

As a basic principle, replanning is necessary, if the process definition that is assigned to a case becomes inadequate. While initially the planner generates an optimal process definition, the example above shows that certain circumstances can make it inadequate for further enactment. To state the adequateness of a process definition more precisely, the properties *working* and *optimal* are defined:

Definition A process definition p is *working* for case c , iff applied on c , it transforms $state_c$ to $goal_c$.

Definition A process definition p is *optimal* for case c , iff it is working for c and no other process definition working for c is better in relation to the optimization metric.

For example, initially the process definition p_1 is optimal for a case c . Process definition p_2 is also working for c , but it is not optimal. It has to be considered that the properties working and optimal also refer to process definitions that have already been started: The state of the case may already have changed and that the execution state of the activity instances may already be running or done. In this case the the properties working and optimal refer to the unfinished part of the process definition and the current state of the case. For example, let us assume that the process definition p_1 is already partially executed. i_3 has completed and has all anticipated effects. p_1 is still optimal, because it transforms the current state $state_c: x \wedge v \wedge t \wedge \neg z$ to the goal state $goal_c: y \wedge w$. If i_3 completes without having the anticipated effect t , p_1 is not working for the case

c any more. As will be shown, process definitions can also loose the property optimal without loosing the property working.

The purpose of replanning is that a case is always assigned to an optimal process definition. It is assumed, that a planner initially generates an optimal process definition for a case. The process definition is then assigned to a case for execution. During execution, the process definition may loose the property optimal. In this case, replanning has to be triggered to generate a new, optimal process definition. If a new process definition is found, proess enactment is adapted. If the planner is unable to find a new process definition, the goal can not be reached any more and human intervention becomes necessary.

4.3 Triggers for Replanning

In this subsection a detailed description of triggers for replanning is given. The presented set of triggers does not claim to be complete, but it covers typical causes for replanning. The triggers are specified in form of Event Condition Action (ECA) rules [2]. Events are identified that threaten the property optimal of the associated process definition. To avoid unnecessary replanning, conditions are defined for every event to specify the exact circumstances under which the event may threaten the property optimal. The triggers in detail:

Trigger 1:

Event: activity instance i has effect e

Condition: $e \notin eff_d$ with activity instance i based on activity definition d

Action: Replan

In the domain, effects are defined for each activity that are anticipated from its execution. For example, the effect y is anticipated from the execution of i_1 . The planner uses this information to plan an optimal process definitions for a case. During execution, an activity may have effects that are not anticipated. In this case, the input of the planner was incorrect. Thus, it can no longer be guaranteed that the process definition is optimal. For example, if i_3 has an unanticipated effect v , p_1 is still working for c , but it is not optimal any more. p_1 is working, because if the remaining activities are scheduled in accordance to p_1 , the goal will be reached. Nevertheless, p_1 is not optimal, because p_3 is also working for c and has lower costs. In contrast to this example, activities can also have effects threatening the property working. The trigger to handle unanticipated effects is defined as follows: The event for Trigger 1 is an activity instance i having an effect e . The condition of Trigger 1 checks, if the effect was anticipated. The domain defines the anticipated effect for each activity. Therefore, it is checked, whether the effect e was defined in the activity definition d of the activity instance i . If this is the case, the input of the planner was correct and it is guaranteed that the process definition is still optimal. Otherwise, the process definition may have lost this property and replanning is necessary.

Trigger 2:Event: activity instance i completesCondition: $rel_i - act_i \neq \emptyset$

Action: Replan

Next to unanticipated effects, an activity can have less effects than anticipated. This complies with the example given at the beginning of Section 4.2, where i_3 is missing the effect t . This can make the process definition loose the property working. In the example above, p_1 loses the property working, because a precondition of the subsequent activity instance i_1 is not satisfied. As an activity can have effects as long as it is running, the proper time to check if it had all anticipated effects is when it completes. Therefore, the event for Trigger 2 is an activity i that completes. If $eff_d - act_i \neq \emptyset$ there is at least one effect $e \in eff_d$ defined in the activity definition that did not occur during the execution of the activity instance. Thus, the input of the planner was not correct and it can not be guaranteed that the process definition is still optimal. Nevertheless, not all effects in the activity definition are always relevant for planning a concrete process definition as explained in Section 4.1. Given the set rel_i of relevant effects for each activity instance, unnecessary replanning can be avoided by defining $rel_i - act_i \neq \emptyset$ as the condition for Trigger 2. In the example above, the relevant effects rel_2 for i_2 are $z \wedge$ increase costs by 20. If i_2 completes and the effect u is missing, there is no need to replan. In contrast, if the effect z is missing, replanning is necessary. This example shows that additional information on relevant effects for each activity instance allows to avoid unnecessary replanning. For this reason the planner adds this information to each activity instance when generating the process definition. Please note that a check $act_i - eff_d \neq \emptyset$ is not necessary, because unanticipated effects have already been taken into account by Trigger 1.

Trigger 3:Event: $goal_c$ of case c changesCondition: $true$

Action: Replan

The event for Trigger 3 is a change in the goal of a case. As a result the associated process definition may lose the property optimal. For example, consider the execution of p_1 . Let us assume that i_3 is in the state running. Due to an external event, e.g. a customer request, the goal state $goal_c$ is changed from $w \wedge y$ to y . As a result, p_1 is not optimal any more, because i_2 and i_4 have become needless. Furthermore, if $goal_c$ is changed to m , p_1 even loses the property working. From this it follows that a change in the goal of a case must trigger replanning to assure to have an optimal process definition.

Trigger 4:Event: external effect on state $state_c$ of case c Condition: *true*

Action: Replan

Trigger 4 handles external effects on the state of the case. External effects are ad hoc effects that can not be assigned to the execution of an activity instance. For example, p_1 is executed and i_3 is running. The coordinator has assigned i_3 to a participant for execution. Now external events may occur that have an effect w on the case, for example: a customer calls, pieces of information are lost or a new law is implemented in the company. Due to the effect w , i_2 and i_4 become needless. Thus, p_1 is not optimal any more. Generally speaking, an external effect is always an unanticipated effect and thus replanning becomes necessary.

Trigger 5:Event: activity definition d changesCondition: \exists an activity instance i based on d with $exec_i \in \{\text{init, running}\}$

Action: Replan

The triggers 5 and 6 deal with changes in the domain. The event for Trigger 5 is a change in an activity definition d . The preconditions or effects of d may change or d becomes unavailable. This event make replanning necessary, if the process definition assigned to the case contains at least one activity instance based on d that is not already done. Changes in activity definitions of finished activity instances are not threatening the properties working and optimal, because the domain only specifies the anticipated behavior of activities instances. Thus, activity instances that are already finished are not affected by domain changes.

Trigger 6:

Event: additional activity definition becomes available

Condition: *true*

Action: Replan

If additional activities become available, the current process definition may not be optimal any more. For example, p_1 is executed and i_3 is running. Consider a new activity definition d_7 that is equal to d_5 with the exception that its costs are only 10. Thus, another process definition p_4 that replaces i_2 and i_4 is better in matters of costs. As new activities can make the current process definition lose the property optimal, replanning should be triggered. Please note that in contrast to Trigger 5 an additional activity definition only threatens the property optimal and not the property working. Thus, if it is sufficient to have a working process definition, Trigger 6 can be renounced to save replanning effort.

5 Discussion

In this paper we specified triggers that enable automated initiation of replanning. For this purpose, events were identified that threaten the adequateness of process definitions. Conditions were assigned to every event, to specify the exact circumstances under which the event makes replanning necessary. In particular, a set of relevant effects rel_i for an activity instance i was defined and it was shown, how it can help to avoid unnecessary replanning. It is important to mention that relevant effects are not part of the standard output of an AI planner. It is possible to check if an effect is relevant, by planning again without this effect and comparing the resulting process definition with the original – that is the way relevance was defined. This approach may not be feasible, because for every effect of every activity instance a process definition has to be planned to check relevance. Instead relevant effects can be determined directly during planning. Basically, the information on causal links and causal link protection generated during planning, has to be analyzed. A detailed description is beyond the scope of this paper. The presented approach is a step towards fully automating replanning. Especially, applications in which replanning is time crucial or has to be done without human interaction, can benefit from the ability to automatically trigger replanning. The presented triggers are part of the replanning concept of an integrated planning and enactment system called Plængine. Currently a first prototype is realized that is able to take a domain and problem definition in PDDL as input, to generate a process definition in the Business Process Execution Language for Web Services [20]. The process definition is then given to the coordinator for process enactment. Automated triggering of replanning is included in the second prototype that is currently in the design phase.

Acknowledgments The authors would like to thank Jens Hündling for his valuable suggestions on this paper.

References

1. S. A. Chun, V. Atluri, and N. R. Adam. Domain knowledge-based automatic workflow generation. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 81–92. Springer-Verlag, 2002.
2. U. Dayal, B. T. Blaustein, A. P. Buchmann, U. S. Chakravarthy, M. Hsu, R. Ledin, D. R. McCarthy, A. Rosenthal, S. K. Sarin, M. J. Carey, M. Livny, and R. Jauhari. The HiPAC Project: Combining Active Databases and Timing Constraints. *SIGMOD Record*, 17(1):51–70, 1988.
3. C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In *Proceedings of conference on Organizational computing systems*, pages 10–21. ACM Press, 1995.
4. P. H. Feiler and W. S. Humphrey. Software process development and enactment: Concepts and definitions. In *Proceedings of the Second International Conference on Software Process*, pages 28–40. IEEE CS Press, 1993.
5. R. E. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

6. D. Georgakopoulos, M. F. Hornick, and A. P. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.
7. M. Hammer and J. Champy. *Reengineering the corporation*. Harper Collins Publishing, New York, 1993.
8. S. Horn and S. Jablonski. An approach to dynamic instance adaption in workflow management applications. In *Conference on Computer Supported Cooperative Work (CSCW)*, 1998.
9. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, 1996.
10. F. Leymann and D. Roller. *Production workflow: concepts and techniques*. Prentice Hall, 2000.
11. T. Madhusudan and J. L. Zhao. A case-based framework for workflow model management. In *Proceedings of the Business Process Management Conference*, volume 2678 of *LNCS*, pages 354–369. Springer, 2003.
12. MD R-Moreno, D. Borrajo, and D. Meziat. Proces modelling and AI planning techniques: A new approach. In *Proceedings of the 2nd International Workshop on Information Integration and Web-based Applications Services*, 2000.
13. K. Myers and P. Berry. The boundary of workflow and ai. In *Proceedings of the AAAI-99, Workshop on Agent-Based Systems in the Business context*, 1999.
14. S. Narayanan and S. McIlraith. Simulation, verification and automated composition of web services. In *11th International World Wide Web Conference*, 2002.
15. G. J. Nutt. The evolution towards flexible workflow systems. *Distributed Systems Engineering*, 3:276–294, Dec 1996.
16. Planning Competition Committee. PDDL – The planning domain definition language. AIPS-98, 1998.
17. M. Reichert and P. Dadam. ADEPT flex -supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
18. H. Schuschel and M. Weske. Integrated workflow planning and coordination. In *14th International Conference on Database and Expert Systems Applications*, volume 2736 of *LNCS*, pages 771–781. Springer, 2003.
19. A. P. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. C. Wileden, and A. L. Wolf. Report from the NSF workshop on workflow and process automation in information systems. *SIGMOD Record*, 25(4):55–67, 1996.
20. S. Thatte, T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, version 1.1. Published on the WWW by BEA Corp., IBM Corp., Microsoft Corp., SAP AG and Siebel Systems, March 2003.
21. The DAML Services Coalition. DAML-S: Web service description for the semantic web. In *The First International Semantic Web Conference (ISWC)*, 2002.
22. D. Wastell, P. White, and P. Kawalek. A methodology for business process redesign: experiences and issues. *Journal of Strategic Information Systems*, 3(1):23–40, 1994.
23. D. S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.
24. Q. Yang. *Intelligent Planning: A Decomposition and Abstraction Based Approach*. Springer, 1997.
25. L. Zeng, B. Benatallah, H. Lei, A. H. H. Ngu, D. Flaxer, and H. Chang. Flexible composition of enterprise web services. *Electronic Markets - Web Services*, 13(2), 2003.