Automated Planning for Supporting Knowledge-Intensive Processes

Sheila Katherine Venero[1(✉)], Bradley Schmerl[2], Leonardo Montecchi[1], Julio Cesar dos Reis[1], and Cecília Mary Fischer Rubira[1]

[1] University of Campinas, Campinas, SP, Brazil
[2] Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. Knowledge-intensive Processes (KiPs) are processes characterized by high levels of unpredictability and dynamism. Their process structure may not be known before their execution. One way to cope with this uncertainty is to defer decisions regarding the process structure until run time. In this paper, we consider the definition of the process structure as a planning problem. Our approach uses automated planning techniques to generate plans that define process models according to the current context. The generated plan model relies on a metamodel called METAKIP that represents the basic elements of KiPs. Our solution explores Markov Decision Processes (MDP) to generate plan models. This technique allows uncertainty representation by defining state transition probabilities, which gives us more flexibility than traditional approaches. We construct an MDP model and solve it with the help of the PRISM model-checker. The solution is evaluated by means of a proof of concept in the medical domain which reveals the feasibility of our approach.

Keywords: Knowledge-intensive process · Business process modeling · Case management · Automated planning · Markov Decision Process · Business process management systems

1 Introduction

In the last decades, the business process management (BPM) community has established approaches and tools to design, enact, control, and analyze business processes. Most process management systems follow predefined process models that capture different ways to coordinate their tasks to achieve their business goals. However, not all types of processes can be predefined at design time—some of them can only be specified at run time because of their high degree of uncertainty [18]. This is the case with *Knowledge-intensive Processes (KiPs)*.

This work is partially supported by CAPES and CNPq scholarships, by the Mobility Program of the Santander Bank, and by the São Paulo Research Foundation (FAPESP) with grants #2017/21773-9 and #2019/02144-6. The opinions expressed in this work do not necessarily reflect those of the funding agencies.

S. Nurcan et al. (Eds.): BPMDS 2020/EMMSAD 2020, LNBIP 387, pp. 101–116, 2020.
https://doi.org/10.1007/978-3-030-49418-6_7

KiPs are business processes with critical decision-making tasks that involve domain-specific knowledge, information, and data [4]. KiPs can be found in domains like healthcare, emergency management, project coordination, and case management, among others. KiP structure depends on the current situation and new emergent events that are unpredictable and vary in every process instance [4]. Thus, a KiP's structure is defined step by step as the process executes, by a series of decisions made by process participants considering the current specific situations and contexts [13]. In this sense, it is not possible to entirely define beforehand which activities will execute or their ordering and, indeed, it is necessary to refine them as soon as new information becomes available or whenever new goals are set.

These kinds of processes heavily rely on highly qualified and trained professionals called *knowledge workers*. Knowledge workers use their own experience and expertise to make complex decisions to model the process and achieve business goals [3]. Despite their expertise, it is often the case that knowledge workers become overwhelmed with the number of cases, the differences between cases, rapidly changing contexts, and the need to integrate new information. They therefore require computer-aided support to help them manage these difficult and error-prone tasks.

In this paper, we explore how to provide this support by considering the process modeling problem as an automated planning problem. Automated planning, a branch of artificial intelligence, investigates how to search through a space of possible actions and environment conditions to produce a sequence of actions to achieve some goal over time [10]. Our work investigates an automated way to generate process models for KiPs by mapping an artifact-centric case model into a planning model at run time. To encode the planning domain and planning problem, we use a case model defined according to the METAKIP metamodel [20] that encloses data and process logic into domain artifacts. It defines data-driven activities in the form of tactic templates. Each tactic aims to achieve a goal and the planning model is derived from it.

In our approach, we use Markov decision processes (MDP) because they allow us to model dynamic systems under uncertainty [7], although our definition of the planning problem model enables using different planning algorithms and techniques. MDP finds optimal solutions to sequential and stochastic decision problems. As the system model evolves probabilistically, an action is taken based on the observed condition or state and a reward or cost is gained [7,10]. Thus, an MDP model allows us to identify decision alternatives for structuring KiPs at run time. We use PRISM [11], a probabilistic model checker, to implement the solution for the MDP model.

We present a proof of concept by applying our method in a medical treatment scenario, which is a typical example of a non-deterministic process. Medical treatments can be seen as sequential decisions in an uncertain environment. Medical decisions not only depend on the current state of the patient, but they are affected by the evolution of the states as well. The evolution of the patient state is unpredictable, since it depends on factors such as preexisting patient

illnesses or patient-specific characteristics of the diseases. In addition, medical treatment decisions involve complex trade-offs between the risks and benefits of various treatment options.

We show that it is possible to generate different optimal treatment plans according to the current patient state and a target goal state, assuming that we have enough data to accurately estimate the transition probabilities to the next patient state. The resulting process models could help knowledge workers to make complex decisions and structure execution paths at run time with more probability of success and optimizing constraints, such as cost and time.

The remainder of this paper is organized as follows: Sect. 2 presents a motivating medical scenario. Section 3 introduces the theoretical and methodological background. Section 4 describes the proposed method to encode a case model as a planning model. Section 5 reports on the application of the methodology in a scenario. Section 6 discusses the obtained findings and related work. Finally, Sect. 7 wraps up the paper with the concluding remarks.

2 Motivating Example

This section presents a motivating medical case scenario. Suppose we have the following medical scenario in the oncology department stored in the Electronic Medical Record (EMR).

Mary, 58 years old, married, two children. She was diagnosed with a lymphoma non-Hodgkin admitted on 20/07/2019 and is receiving R-ICE Chemotherapy. R-ICE is named after the initials of the drugs used: rituximab, ifosfamide, carboplatin, etoposide. R-ICE is applied as a course of several sessions (cycles) of treatment over a few months. On 02/10/2019, Mary is supposed to receive the second cycle of R-ICE. However, on admission, she is febrile at 38 °C *and presents severe nausea (Level 4).*

In order to receive the second cycle of R-ICE, it is necessary to stabilize Mary's health status as soon as possible. Thus, at this time the goal is to decrease her body temperature to $36.5\,^{\circ}\text{C} \leq Temp \leq 37.2\,^{\circ}\text{C}$ and reduce the level of nausea to zero $LN = 0$. For that, physicians need to choose from vast treatment strategies to decide which procedures are the best for Mary, in her specific current context.

Assume that we have statistical data about two possible tactics for achieving the desired goal: fever (Fvr) and nausea (Nausea) management, shown in Table 1 adapted from [2]. Each of these tactics can be fulfilled through multiple activities that have different interactions and constraints with each other, as well as to the specifics of the patient being treated. For example, (a) treating nausea with a particular drug may affect the fever, (b) administration of the drug may depend on the drugs that the patient is taking, (c) drug effectiveness may depend on the patient history with the drug, or (d) giving the drug may depend on whether the drug has already been administered and how much time has elapsed since the last dose. These issues make manual combination of even this simple case challenging, and it becomes much harder for more complex

treatments and patient histories. Support is therefore needed that can take into account patient data, constraints, dependencies, and patient/doctor preferences to help advise the doctor on viable and effective courses of treatment.

Table 1. Tactics templates for fever (FVR) and nausea (NAUSEA) management

Tactic: Fever Management (FVR)	**Tactic:** Nausea Management (NAUSEA)
Definition: Management of a patient with hyperpyrexia caused by non-environmental factors	**Definition:** Prevention and alleviation of nausea
Goal: Thermoregulation ($36.5\,^{\circ}\text{C} \leq \ Temp \ \leq 37.2\,^{\circ}\text{C}$)	**Goal:** Stop Nausea (LoN = 0)
Metric: Temperature (Temp)	**Metric:** Level of Nausea (LoN)
Preconditions: $Temp > 37.2\,^{\circ}\text{C}$	**Preconditions:** LoN > 0
Activities:	**Activities:**
A1. Administer ORAL antipyretic medication,as appropriate **A2.** Administer INTRAVENOUS antipyretic medication, as appropriate **A3.** Administer medications to treat the cause of fever, as appropriate **A4.**Encourage increased intake of oral fluids, as appropriate **A5.** Administer oxygen, as appropriate	**B1.** Ensure that effective antiemetic drugs are given to prevent nausea when possible (except for nausea related to pregnancy) **B2.** Control environmental factors that may evoke nausea (e.g., aversive smells, sound and unpleasant visual stimulation **B3.** Give cold, clear liquid and odorless and colorless food, as appropriate

3 Background

This section presents the underlying concepts in our proposal. Section 3.1 provides an overview of the METAKIP metamodel; Sect. 3.2 introduces basic concepts of automated planning; Sect. 3.3 explains Markov decision process (MDP). Section 3.4 describes the PRISM tool and language.

3.1 METAKIP: A Metamodel for KiPs Definition

Our previous work proposed an artifact-centric metamodel [20] for the definition of KiPs, aiming to support knowledge workers during the decision-making process. The metamodel supports data-centric process management, which is based on the availability and values of data rather than completion of activities.

In data-centric processes, data values drive decisions and decisions dynamically drive the course of the process [18]. The metamodel is divided into four major packages: case, control-flow, knowledge, and decision, in such a way that there is an explicit integration of the data, domain, and organizational knowledge, rules, goals, and activities.

The Case Package defines the base structure of the metamodel, a *Case*. A case model definition represents an integrated view of the context and environment data of a case, following the artifact-centric paradigm. This package is composed of a set of interconnected artifacts representing the logical structure of the business process. An *artifact* is a data object composed of a set of items, attributes, and data values, defined at run time.

The Knowledge Package captures explicit organizational knowledge, which is encoded through *tactic templates*, *goals*, and *metrics* that are directly influenced by business rules. Tactics templates represent best practices and guidelines. Usually, they have semi-structured sequences of activities or unstructured loose alternative activities pursuing a goal.

The Control-flow Package defines the *behavior* of a case. It is composed of a set of data-driven activities to handle different cases. Activity definitions are made in a declarative way and have *pre-* and *post-conditions*. The metamodel refines the granularity of an *activity* that could be a step or a task. A *task* is logically divided into *steps*, which allows better management of data entry on the artifacts. Step definitions are associated with a single attribute of an artifact, a resource, and a role type at most. This definition gives us a tight integration between data, steps and resources.

These packages are used to model alternative plans to answer emergent circumstances, reflecting environmental changes or unexpected outcomes during the execution of a KiP. The Decision Package represents the structure of a collaborative decision-making process performed by knowledge workers. We proposed a representation of how decisions can be made by using the principles of strategic management, such as, looking towards goals and objectives and embracing uncertainty by formulating strategies for the future and correct them if necessary. The strategic plan is structured at run time by goals, objectives, metrics and tactic templates.

3.2 Automated Planning

Planning is the explicit and rational deliberation of actions to be performed to achieve a goal [7]. The process of deliberation consists of choosing and organizing actions considering their expected outcomes in the best possible way. Usually, planning is required when an activity involves new or less familiar situations, complex tasks and objectives, or when the adaptation of actions is constrained by critical factors such as high risk. Automated planning studies the deliberation process computationally [7].

A conceptual model for planning can be represented by a state-transition system, which formally is a 4-tuple $\Sigma = (S, A, E, \gamma)$, where $S = \{s_1, s_2, ...\}$ is a finite or recursively enumerable set of states; $A = \{a_1, a_2, ...\}$ is a finite or

recursively enumerable set of actions; $E = \{e_1, e_2, ...\}$ is a finite or recursively enumerable set of events; and $\gamma : S \times A \times E \rightarrow 2^S$ is a state-transition function.

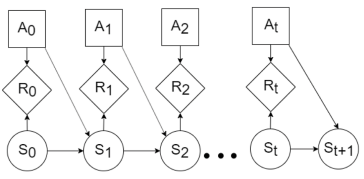
Actions are transitions controlled by a plan executor. Events are unforeseen transitions that correspond to the internal dynamics of the system and cannot be controlled by the plan executor. Both events and actions contribute to the evolution of the system. Given a state transition system Σ, the purpose of planning is to deliberate which actions to apply into which states to achieve some goal from a given state. A plan is a structure that gives the appropriate actions.

3.3 Markov Decision Process (MDP)

A Markov decision process (MDP) is a discrete-time stochastic control process. It is a popular framework designed to make decisions under uncertainty, dealing with nondeterminism, probabilities, partial observability, and extended goals [7].

In MDPs, an agent chooses action a based on observing state s and receives a reward r for that action [10]. The state evolves probabilistically based on the current state and the action taken by the agent.

Figure 1(a) presents a decision network [10], used to represent a MDP. The state transition function $T(s'|s, a)$ represents the probability of transitioning from state s to s' after executing action a. The reward function $R(s, a)$ represents the expected reward received when executing action a from state s. We assume that the reward function is a deterministic function of s and a.



(a)

```
mdp                                  // type of model
module <Module_name>
  // the values of var_1 and var_2 represent the state of the module
<var_1>: [0..n]  init 0;       // local variable
<var_2>: bool   init false;
//commands
[<action_1>] <guard> --> prob_1 : update_1 + ... + prob_n :
update_n;
[<action_2>] <guard> --> prob_1 : update_1 + ... + prob_n :
update_n;
endmodule

rewards
[ ] true : 1;
[<action_1>] true : a;
[<action_2>] true : 2*b;
endrewards
```

(b)

Fig. 1. (a) MDP representation [10] and (b) Example syntax of *mdp* PRISM [11] module and rewards

An MDP treats planning as an optimization problem in which an agent needs to plan a sequence of actions that maximizes the chances of reaching the goal. Action outcomes are modeled with a probability distribution function. Goals are represented as utility functions that can express preferences on the entire execution path of a plan, rather than just desired final states. For example, finding the optimal choice of treatment optimizing the life expectancy of the patient or optimizing cost and resources.

3.4 PRISM

PRISM [11] is a probabilistic model checker that allows the modeling and analysis of systems that exhibit probabilistic behavior. The PRISM tool provides support for modeling and construction of many types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs), and probabilistic timed automata (PTAs). The tool supports statistical model checking, confidence-level approximation, and acceptance sampling with its discrete-event simulator. For nondeterministic models it can generate an optimal adversary/strategy to reach a certain state.

Models are described using the PRISM language, a simple, state-based language based on the reactive modules formalism [1]. Figure 1(b) presents an example of the syntax of a PRISM module and rewards. The fundamental components of the PRISM language are modules. A module has two parts: variables and commands. Variables describe the possible states that the module can be in at a given time. Commands describe the behavior of a module, how the state changes over time. A command comprises a guard and one or more updates. The guard is a predicate over all the variables in the model. Each update describes a transition that the module can take if the guard is true. A transition is specified by giving the new values of the variables in the module. Each update has a probability which will be assigned to the corresponding transition. Commands can be labeled with actions. These actions are used for synchronization between modules. Cost and rewards are expressed as real values associated with certain states or transitions of the model.

4 Dynamic Plan Generation for KiPs Execution

In our approach, plans are fragments of process models that are frequently created and modified during process execution. Plans may change as new information arrives and/or when a new goal is set. We advocate the creation of a planner to structure process models at run time based on a knowledge base. The planner synthesizes plans on-the-fly according to ongoing circumstances. The generated plans should be revised and re-planned as soon as new information becomes available. Thereby, it involves both computer agents and knowledge workers in a constant interleaving of planning, execution (configuration and enactment), plan supervision, plan revision, and re-planning. An interactive software tool might assist human experts during planning. This tool should allow defining planning goals and verifying emerging events, states, availability of activities and resources, as well as preferences.

4.1 Model Formulation

The run-time generation of planning models according to a specific situation in a case instance requires the definition of the planning domain and then the planning problem itself.

Definition 1. *Let the case model be represented according to the METAKIP metamodel. The planning domain is derived from the case model that can be described using a state-transition system defined as a 5-tuple* $\Sigma = (S, A, E, \gamma, C)$ *such as that:* S *is the set of possible case states.* A *is the set of actions that are represented by activities inside tactics that an actor may perform.* E *is the set of events in the context or in the environment.* $\gamma : S \times A \times E \rightarrow 2^S$*, is the state-transition function, so the system evolves according to the actions and events that it receives.* $C : S \times A \rightarrow [0, \infty)$ *is the cost function that may represent monetary cost, time, risk or something that can be minimized or maximized.*

The state of a case is the set of values (available data) of the attributes contained in artifacts of the *context* and the *environment*. However, since the number of attributes of the artifacts is very large, it is necessary to limit the number of attributes to only the most relevant ones, which determines the current state of the case at a given time t.

Definition 2. *A state* s_t *is the set of values corresponding to a set of relevant attributes* $\{v_1, v_2, \ldots v_r\}$*, with* $r \geq 1$*, contained in the business artifacts at a given time* t*.*

Actions in the METAKIP metamodel are represented by the activities within a tactic. Tactics represent best practices and guidelines used by the knowledge workers to make decisions. In METAKIP, they serve as tactic templates to be instantiated to deal with some situations during the execution of a case instance. Tactics are composed of a finite set of activities pursuing a goal. A tactic can be structured or unstructured. A tactic is a 4-tuple $T = (G, PC, M, A)$, where: G is a set of variables representing the pursuing goal state, PC is a finite set of preconditions representing a state required for applying the tactic, M is a set of metrics to track and assess the pursuing goal state, and A is a finite set of activities.

In METAKIP, an activity could be a single step or a set of steps (called a task). An activity has some preconditions and post-conditions (effects). We map activities into executable actions. An executable action is an activity in which their effects can modify the values of the attributes inside business artifacts. These effects can be deterministic or non-deterministic.

Definition 3. *An action is a 4-tuple* $a = (Pr, Eff, Pb, c)$ *where:* Pr *is a finite set of preconditions.* Eff *is a finite set of effects.* Pb *is a probability distribution on the effects, such that,* $P_{ef}(i)$ *is the probability of effect* $ef \in Eff$ *and* $\sum_{ef \in Eff} P_{ef}(i) = 1$*.* c *is the number which represents the cost (monetary, time, etc.) of performing* a*.*

As the state-transition function γ is too large to be explicitly specified, it is necessary to represent it in a generative way. For that, we use the planning operators from which it is possible to compute γ. Thus, γ can be specified through a set of planning operators O. A planning operator is instantiated by an action.

Definition 4. *A planning operator O is a pair (id, a) where a is an action and id is a unique identifier of action a.*

At this point, we are able to define the planning problem to generate a plan as a process model.

Definition 5. *The planning problem for generating a process model at a given time t is defined as a triple $P = (OS_t, GS_t, RO_t)$, where: OS_t is the observable situation of a case state at time t. GS_t is the goal state at time t, a set of attributes with expected output values. RO_t represents a subset of the O that represents only available and relevant actions for a specific situation during the execution of a case instance at a given time t.*

Definition 6. *The observable situation of a case instance C state at a given time t is a set of attributes $OS_t = \{v_1, v_2, \ldots, v_m\}$, with $m \geq 1$, such that $v_i \in S_t \cup I_t$ for each $1 \leq i \leq m$, where the state of C is S_t and the set issues in the situation of C is I_t.*

Definition 7. *The goal state of an observable situation of case instance C at a given time t is the set of attributes $GS_t = \{v_1, v_2, \ldots, v_m\}$, with $m \geq 1$, such that, for $1 \leq i \leq m$, v_i is an attribute with an expected output value, v_i belongs to an artifact of C. These attributes are selected by the knowledge workers. Some metrics required to asses some goals inside tactics can be added to the goal. GS_t represents the expected reality of C.*

GS_t serves as an input for searching an execution path for a specific situation. Different goal states can be defined over time.

Definition 8. *Let $P = (OS_t, GS_t, RO_t)$ be the planning problem. A plan π is a solution for P. The state produced by applying π to a state OS_t in the order given is the state GS_t. A plan is any sequence of actions $\pi = (a_1, ..., a_k)$, where $k \geq 1$. The plan π represents the process model.*

Our problem definition enables the use of different planning algorithms and the application of automatic planning tools to generate alternatives plans. As we are interested in KiPs, which are highly unpredictable processes, we use Markov Decision Processes for formulating the model for the planner. MDPs allows us to represent uncertainty with a probability distribution. MDP makes sequential decision making and reasons about the future sequence of actions and obstructions, which provides us with high levels of flexibility in the process models. In the following, we show how to derive an MDP model expressed in the PRISM language from a METAKIP model automatically.

4.2 PRISM Model Composition

Algorithm 1 shows the procedure to automatically generate the MDP model for the PRISM tool, where the input parameters are: OS_t, GS_t, set of domain

Tactics, t is the given time, PP minimum percentage of preconditions satisfaction, and PG minimum percentage of goal satisfaction, both PP and PG are according to the rules of the domain. As described in Sect. 3.4, a module is composed of variables and commands. Variables of the module are the set of attributes from the case artifacts that belong to $OS_t \cup GS_t$. Commands are represented for the relevant planning operators RO_t. The name of the command is the identifier of the action, the guards are the preconditions PC and the effects *Eff* are the updates with associated probabilities. Rewards are represented by the cost of actions c and are outside of the module of PRISM.

Algorithm 1. PRISM Model Generator

Require: OS_t, GS_t, $Tactics$, t, PP, PG
  $V \leftarrow OS_t \cup GS_t$ ▷ Attributes of OS_t *and* GS_t correspond to PRISM variables
  for all $T \in Tactics$ **do** ▷ For each tactic
    $p_1 \leftarrow |T.PC \cap OS_t|/|T.PC|$ ▷ Percentage of satisfied preconditions
    $p_2 \leftarrow |GS_t \cap T.G|/|GS_t|$ ▷ Percentage of achievable target goal
    if $p_1 \geq PP$ and $p_2 \geq PG$ **then** ▷ If percentages are acceptable
      $ST \leftarrow ST \cup T$ ▷ Add to the set of selected tactics
    end if
  end for
  $RT \leftarrow SelectRevevantTactics(ST)$ ▷ Relevant tactics for the current situation OS_t
  $A_t \leftarrow CheckAvailableActivities(RT, t)$ ▷ Select available activities at time t
  $RO_t \leftarrow CreatePlanningOperators(A_t)$
  $C \leftarrow CreateCommands(RO_t)$
  $R \leftarrow CreateRewards(RO_t)$
  $V \leftarrow V \cup \{T.M : T \in RT\}$ ▷ Add necessary metrics to evaluate
  $CreatePRISMModel(V, C, R)$

For finding the set of relevant planning operators RO_t, first, we select tactics whose preconditions must be satisfied by the current situation OS_t and whose goal is related to the target state GS_t. This can be done by calculating the percentages of both the satisfied preconditions and achievable goals. If these percentages are within an acceptable range according to the rules of the domain, the tactics are selected. Second, this first set of tactics is shown to the knowledge workers who select the most relevant tactics. The set of the selected relevant tactics is denoted as RT. From this set of tactics, we verify which activities inside the tactics are available at time t. Thus, the set of available actions at time t is denoted by $A_t = a_1, a_2, \ldots, a_n$. Finally, the relevant planning operators, RO_t, are created by means of A_t.

4.3 Plan Generation

To generate plans in PRISM, it is necessary to define a property file that contains properties that define goals as utility functions. PRISM evaluates properties over an MDP model and generates all possible resolutions of non-determinism in the model, state graphs, and gives us the optimal state graph. The state graph describes a series of possible states that can occur while choosing actions aiming to achieve a goal state. It maximizes the probability to reach the goal state

taking into consideration rewards computed, that is maximizing or minimizing rewards and costs.

In our context, a property represents the goal state GS_t to be achieved while trying to optimize some criteria. Then, PRISM calculates how desirable an executing path is according to one criterion. Thus, plans can be customized according to knowledge workers' preferences (costs and rewards). To generate a plan, we need to evaluate a property. The generated plan is a state graph that represents a process model to be executed at time t. The generated process model shows case states as nodes and states transitions as arcs labeled with actions which outcomes follow probability distribution function. According to this state graph, the knowledge worker could choose which action to execute in a particular state. This helps knowledge workers to make decisions during KiPs execution.

5 Proof of Concept

This section formulates a patient-specific MDP model in PRISM for the medical scenario presented in Sect. 2. In the area of health care, medical decisions can be modeled with Markov Decisions Processes (MDP) [5,17]. Although MDP is more suitable for certain types of problems involving complex decisions, such as liver transplants, HIV, diabetes, and others, almost every medical decision can be modeled as an MDP [5]. We generate the PRISM model by defining the observable situation OS_t, Goal state GS_t, and the set of relevant planning operators RO_t.

Table 2. Activity modeling

Activity A1: Administer Oral antipyretic medication, as appropriate	**Activity B1: Ensure that effective antiemetic drugs are given to prevent nausea when possible**
Pre-condition: ((Temp $>$ 37.2) and (LN $=$ 0 or LN $=$ 1)) and (allergic $=$ false) and (conflict with current medications $=$ false) and (medication is available $=$ true)	**Pre-condition:** Pregnancy(FALSE) and (LN $>$ 2) and (allergic $=$ false) and (conflict with current medications $=$ false)
Effects:	**Effects:**
E1: p $=$ 0.6 Respond to treatment (Temp $=$ 37)	**E1**: p $=$ 0.7 Respond to treatment (LN $=$ 0)
E2: p $=$ 0.3 Partial Respond to treatment (Temp $=$ Temp $-$ 0.5)	**E2:** p $=$ 0.2 Partially respond to treatment (LN $=$ LN $-$1)
E3: p $=$ 0.1 Not Responding to treatment (Temp $=$ Temp $+$ 0.5)	**E3:** p $=$ 0.1 Not Responding to treatment (LN $=$ LN $+$1))
Task execution time : 5 min	**Task execution time :** 5 min
Cost: 0.08	**Cost: 0.08**

Taking in consideration the medical scenario, the observable situation is $OS_0 = \{Temp_0 = 38\,°, LN_0 = 4\}$ and the goal state is $GS_0 = \{36\,°\text{C} \leq Temp \leq 37.2°\text{C}, LN = 0\}$ where: Temp is the temperature of the patient and LN is the level of nausea, both attributes of the *Health Status* artifact. We assume that the set of relevant tactics RT according to the current health status of the patient are fever and nausea management, presented in Sect. 2.

Table 2 shows the specification of one activity of each tactic, showing their preconditions, effects with their probability, time, and cost of execution. We modeled the activity effects with probabilities related to the probability of the patient to respond to the treatment. For example, the possible effects of applying the activity *Administer ORAL antipyretic medication* are: (E1) the patient successfully responds to treatment, occurring with a probability 0.6; (E2) 30% of the time the patient partially responds to treatment where their temperature decreases by 0.5° or more fails to reach the goal level; and (E3) the patient does not respond at all to treatment or gets worse (occurring with a probability of 0.1). The other activities are similarly modeled according to the response of the patient. Assuming that all activities from both tactics are available, the set of executable actions is $A_t = \{A1, A2, A3, A4, A5, B1, B2, B3\}$. Then, it is possible to model the set of relevant planning operators RO_t. Having OS_t, GS_t and RO_t, it is possible to generate the MDP model in the language PRISM.

Once we created the MDP model, the following utility functions were evaluated: minimize time and cost while reaching the target state. The optimal plan to achieve the goal state GS_t while minimizing the cost shows that reachability is eight iterations. The resulting model has 13 states, 35 transitions, and 13 choices. The time for the model construction was 0.056 s.

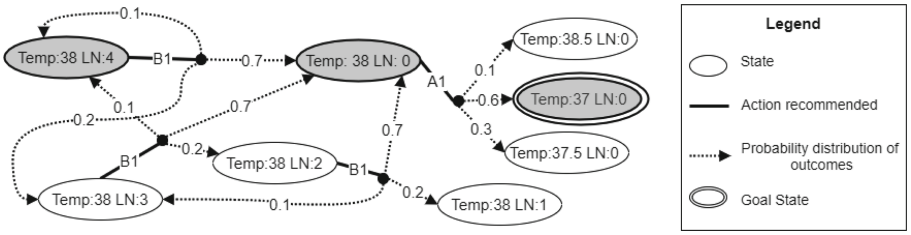


Fig. 2. Plan for reaching the goal state optimizing the cost

Figure 2 presents only a fragment of the model generated, highlighting the most probable path from the initial state to the goal state. The first suggested action is $B1$ (labeled arc) with possible outcome states with their probabilities. If the most probable next state is achieved, the next action to perform is $A1$ which has a probability of 0.6 to reach the goal state. Knowledge workers can use this generated plan to decide which is the next activity they should perform in a particular state. To make the plan readable to knowledge workers, they could be presented with only the most probable path, and this could be updated

according to the state actually reached after activity execution. Further studies are necessary to help guiding knowledge workers in interpreting and following the model.

6 Discussion and Related Work

In the last decades, there has been a growing interest in highly dynamic process management, with different types of approaches that deal with the variability, flexibility, and customization of processes at design time and at run time. Most approaches start from the premise that there is a process model to which different changes have to be made, such as adding or deleting fragments according to a domain model or to generate an alternative sequence of activities due to some customization option. A few approaches use automated planning for synthesizing execution plans. Laurent *et al.* [12] explored a declarative modeling language called Alloy to create the planning model and generate the plans. This approach seems to be very promising for activity-centric processes, but not effective enough for data-centric processes, as data is not well-enough treated to be the driver of the process as required in KiPs.

SmartPM [16] investigated the problem of coordinating heterogeneous components inside cyber-physical systems. They used a PDDL (Planning Domain Definition Language) planner that evaluates the physical reality and the expected reality, and synthesize a recovery process. Similarly, Marrella and Lespérance proposed an approach [15] to dynamically generate process templates from a representation of the contextual domain described in PDDL, an initial state, and a goal condition. However, for the generation of the process templates, it is assumed that tasks are black boxes with just deterministic effects. On the other hand, Henneberger *et al.* [8] explored an ontology for generating process models. The generated process models are action state graphs (ASG). Although this work uses a very interesting semantic approach, they did not consider important aspects such as resources and cost for the planning model.

There has been an increasing interest in introducing cognitive techniques for supporting the business process cycle. Ferreira *et. al.* [6] proposed a new life cycle for workflow management based on continuous learning and planning. It uses a planner to generate a process model as a sequence of actions that comply with activity rules and achieve the intended goal. Hull and Nezhad [9] proposed a new cycle Plan-Act-Learn for cognitively-enabled processes that can be carried out by humans and machines, where plans and decisions define actions, and it is possible to learn from it. Recently, Marrella [14] showed how automatic planning techniques can improve different research challenges in the BPM area. This approach explored a set of steps for encoding a concrete problem as a PDDL planning problem with deterministics effects.

In this paper we introduced the notion of the state of a case regarding data-values in the artifacts of a case instance. From this state, we can plan different trajectories towards a goal state using automated planning techniques. Our solution generates action plans considering the non-deterministic effects of the

actions, new emerging goals and information, which provides high levels of flexibility and adaptation. As we describe a generic planning model, it is possible to use different planning algorithms or combine other planning models, such as the classical planning model or the hierarchical task network (HTN), according to the structuring level of the processes at different moments. Thereby, we could apply this methodology to other types of processes, from well-structured processes to loosely or unstructured processes.

Our approach relies on MDP, which requires defining transition probabilities, which in some situations can be very difficult and expensive to get. Nowadays a huge amount of data is produced by many sensors, machines, software systems, etc, which might facilitate the acquisition of data to estimate these transition probabilities. In the medical domain, the increasing use of electronic medical record systems shall provide the medical data from thousands of patients, which can be exploited to derive these probabilities. A limitation in MDPs refers to the size of the problem because the size of the state-space explodes, and it becomes more difficult to solve. In this context, several techniques for finding approximate solutions to MDPs can be applied in addition to taking advantage of the rapid increase of processing power in the last years.

Flexible processes could be easily designed if we replan after an activity execution. In fact, our approach suggests a system that has a constant interleaving of planning, execution, and monitoring. In this way, it will help knowledge workers during the decision-making process.

7 Conclusion

Process modeling is usually conducted by process designers in a manual way. They define the activities to be executed to accomplish business goals. This task is very difficult and prone to human errors. In some cases (*e.g.*, for KiPs), it is impossible due to uncertainty, context-dependency, and specificity. In this paper, we devised an approach to continually generate run-time process models for a case instance using an artifact-centric case model, data-driven activities, and automatic planning techniques, even for such loosely-structured processes as KiPs.

Our approach defined how to synthesize a planning model from an artifact-oriented case model defined according to the METAKIP metamodel. The formulation of the planning domain and the planning problem rely on the current state of a case instance, context and environment, target goals, and tactic templates from which we can represent actions, states, and goals. As our focus is KiPs management, we chose to use the MDP framework that allows representing uncertainty, which is one of KiPs essential characteristics. To automatically generate the action plan, we used the tool PRISM, which solves the MDP model and provides optimal solutions.

Future work involve devising a user-friendly software application for knowledge workers to interact with the planner and improve the presentation of plans in such a way that it is more understandable to them. Our goal is to develop

a planner which combines different types of planning algorithms to satisfy different requirements in business processes, especially regarding the structuring level. This planner will be incorporated into a fully infrastructure for managing Knowledge-intensive processes that will be based on the DW-SAArch reference architecture [19].

References

1. Alur, R., Henzinger, T.A.: Reactive modules. Form. Methods Syst. Des. **15**(1), 7–48 (1999)
2. Butcher, H.K., Bulechek, G.M., Dochterman, J.M.M., Wagner, C.: Nursing Interventions classification (NIC)-E-Book. Elsevier Health Sciences (2018)
3. Davenport, T.: Thinking for a Living. How to Get Better Performance and Results. Harvard Business School Press, Boston (2005)
4. Di Ciccio, C., Marrella, A., Russo, A.: Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches. J. Data Semant. **4**(1), 29–57 (2015)
5. Dıez, F., Palacios, M., Arias, M.: MDPs in medicine: opportunities and challenges. In: Decision Making in Partially Observable, Uncertain Worlds: Exploring Insights from Multiple Communities (IJCAI Workshop), vol. 9, p. 14 (2011)
6. Ferreira, H.M., Ferreira, D.R.: An integrated life cycle for workflow management based on learning and planning. Int. J. Cooper. Inf. Syst. **15**(04), 485–505 (2006)
7. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Elsevier (2004)
8. Henneberger, M., Heinrich, B., Lautenbacher, F., Bauer, B.: Semantic-based planning of process models. In: Multikonferenz Wirtschaftsinformatik (MKWI). GITO-Verlag (2008)
9. Hull, R., Motahari Nezhad, H.R.: Rethinking BPM in a cognitive world: transforming how we learn and perform business processes. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 3–19. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_1
10. Kochenderfer, M.J.: Decision Making Under Uncertainty: Theory and Application. MIT press, Cambridge (2015)
11. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
12. Laurent, Y., Bendraou, R., Baarir, S., Gervais, M.P.: Planning for declarative processes. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 1126–1133. ACM (2014)
13. Marjanovic, O.: Towards is supported coordination in emergent business processes. Bus. Process Manag. J. **11**(5), 476–487 (2005)
14. Marrella, A.: Automated planning for business process management. J. Data Seman. **8**(2), 79–98 (2019)
15. Marrella, A., Lespérance, Y.: A planning approach to the automated synthesis of template-based process models. SOCA **11**(4), 367–392 (2017)
16. Marrella, A., Mecella, M., Sardina, S.: SmartPM: an adaptive process management system through situation calculus, IndiGolog, and classical planning. In: Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2014), pp. 518–527 (2014)

17. Mattila, R., Siika, A., Roy, J., Wahlberg, B.: A Markov decision process model to guide treatment of abdominal aortic aneurysms. In: 2016 IEEE Conference on Control Applications (CCA), pp. 436–441. IEEE (2016)
18. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods Technologies. Springer, Heidelberg (2012)
19. Venero, S.K.: DW-SAAArch: a reference architecture for dynamic self-adaptation in workflows. Master's Thesis, UNICAMP, Campinas, Brazil (2015)
20. Venero, S.K., Dos Reis, J.C., Montecchi, L., Rubira, C.M.F.: Towards a metamodel for supporting decisions in knowledge-intensive processes. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 75–84. ACM (2019)