2018 ACM/IEEE 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems

RE-STORM: Mapping the Decision-Making Problem and Non-Functional Requirements Trade-off to Partially Observable Markov Decision Processes

Luis H. Garcia Paucar
ALICE, SARI
Aston University, UK B4 7ET
garciapl@aston.ac.uk

Nelly Bencomo
ALICE, SARI
Aston University, UK B4 7ET
nelly@acm.org

ABSTRACT

Different model-based techniques have been used to model and underpin requirements management and decision-making strategies under uncertainty for self-adaptive Systems (SASs). The models specify how the partial or total fulfilment of non-functional requirements (NFRs) drive the decision-making process at runtime. There has been considerable progress in this research area. However, precarious progress has been made by the use of models at runtime using machine learning to deal with uncertainty and support decision-making based on new evidence learned during execution. New techniques are needed to systematically revise the current model and the satisficement of its NFRs when empirical evidence becomes available from the monitoring infrastructure. In this paper, we frame the decision-making problem and trade-off specifications of NFRs in terms of Partially Observable Markov Decision Processes (POMDPs) models. The mathematical probabilistic framework based on the concept of POMDPs serves as a runtime model that can be updated with new learned evidence to support reasoning about partial satisficement of NFRs and their trade-off under the new changes in the environment. In doing so, we demonstrate how our novel approach RE-STORM underpins reasoning over uncertainty and dynamic changes during the system's execution.

KEYWORDS

Requirements, specification, self-adaptation, decision-making, uncertainty, POMDPs, bayesian decision theory

ACM Reference format:
Luis H. Garcia Paucar and Nelly Bencomo. 2018. RE-STORM: Mapping the Decision-Making Problem and Non-Functional Requirements Trade-offto Partially Observable Markov Decision Processes. In *Proceedings of SEAMS '18: 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems , Gothenburg, Sweden, May 28–29, 2018 (SEAMS '18),* 7 pages.
https://doi.org/10.1145/3194133.3195537

1 INTRODUCTION

A self-adaptive system (SAS) is a system that can adapt its behaviour in response to its own changes and environmentalfl uctuations at runtime [29]. The decision-making process of a SAS is challenged by the underlaying uncertainty [12]. Different sources of uncertainty in SASs have been identified and studied [1, 13, 35].

In this paper we focus on the kind of uncertainty associated with the satisficement of non-functional requirements (NFRs) given a set of design decisions reflected in a SAS configuration [4, 5]. Specifically, we are interested on the specification and runtime handling of the uncertainty related to the levels of saticeficement of the NFRs when new evidence is collected, and that may create the need of adaptation or reconfiguration. Several authors have approached different issues related to uncertainty and NFRs trade-offs [9, 12, 13, 18, 27, 30–32, 36, 38]. However, critical challenges need to be further explored. One of the issues is that current approaches to deal with uncertainty focus mainly on the design-time activities for the specification of NFRs [11, 13, 20, 21, 32].
On the other hand, recent progress on POMDPs implementations [28] have shown promising results in the AI research community for decision-making under uncertainty and dynamic environments [3, 14, 19, 33, 37]. Some scalability issues [37] have been overcome for the decision-making planning when models and assumptions change at runtime [19, 37]. For instance, in [3], an autonomous driving vehicle performs decision-making tasks in real time for a complex and dynamic environment. However, the models used scarce of support for specific definition and further representation, monitoring and trade-off of NFRs as the environment changes.
This paper tackles these challenges with RE-STORM, a novel approach that allow us to (i) explicitly deal with the uncertainty associated with the current runtime context (i.e., the current belief expressed as a probability distribution) over the system's NFRs satisficement, (ii) balance different conflicting NFRs, (iii) maintain the definition of uncertainty over time as new evidence arrives in a consistent way with the past using Bayesian learning and (iv) incorporate risk preferences (i.e., rewards and penalties) that properly address the current situation modelled. Difference from our previous work [4, 5], in this paper we use POMDPs and avoid the "curse of history" problem [37] making the runtime models able to deal with the uncertainty in a scalable way over time by controling the exploration-exploitation trade-off during the online look ahead search provided by DESPOT [37]. The remainder of this paper is organized as follows: Section 2 presents a background on POMDPs. Section 3 presents how the specification of requirements can be performed using POMDPs. Finally, Section 4 concludes the paper.

2 BACKGROUND

This section briefly overviews POMDPs and their relevance for decision-making in SASs.

2.1 POMDPs

Partially Observable Markov Decision Processes (POMDPs) provide a principled approach to make rational decisions in the face of uncertainty within changing environments [19]. A POMDP models an agent acting in a partially observable stochastic environment. A rational agent is one that acts so as to achieve the best expected outcome [29]. Fig. 1 shows a POMDP with its main elements.

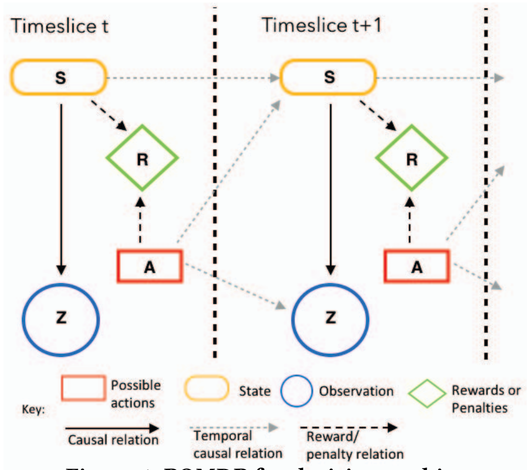


Figure 1: POMDP for decision-making

2.1.1 *POMDP representation.*

POMDPs constitute a useful framework for modelling beliefs about the world, associating rewards with states of the world in order to make decisions. A POMDP is specified as a 6-tuple (S, A, Z, T, O, R), where:

- S represents a set of states that the agent can achieve. A state is a description of the environment at a specific point in time. It should capture all information that is relevant to the decision-making process in a system [28]. All relevant information is provided to S through observations of the system's environment.
- Z is a set of observations related to the set of states S. An assumption in POMDPs is that the states s ∈ S are not directly observable [28] due to errors in control and sensing under uncertain and dynamic environments [2]. Instead, the agent receives observations z ∈ Z that provide partial information of the state.
- A is a set of actions that the agent choose to perform.
- T:SxAxS → [0,1] is the transition model. It describes an agent taking action a ∈ A in state s ∈ S, and updatesthe new state s' ∈ S in the next time slice with probability P(s' |s, a) = T (s, a, s').
- O:SxAxZ → [0,1] is the observation model. It describes the probability P(z|s', a) = O(s', a, z) of observing z ∈ Z when the action a is performed and the resulting state is s'.
- R: SxA → ℝ is the reward function. R(s,a) is the reward obtained for performing action a under the current state s. The R(s,a) values are usually defined at design time based on information provided by the system's stakeholders or based on previous experiences.

As the states in a POMDP are not directly observable, the agent cannot choose its actions based on the states, instead it is based on belief states and policies related to these beliefs. A policy maps from beliefs to the actions [28]. Both concepts are explained next.

2.1.2 *POMDP based decision-making process.*

- Belief state. A belief state b is a probability distribution over the states space S. The agent starts with an initial belief state b_0, representing its knowledge about the starting state of the environment. At any slice time t, the belief b may be updated according to the Bayes' rule, by incorporating information from the action a, the previous state and the observation z [29]. The updated belief b' is represented as b'=r(b,a,z) for some a ∈ A and z ∈ Z. The next step is to choose an action based on the belief state.
- Policies and decision-making. A policy π: $\beta \rightarrow$ A specifies a mapping from the belief space β to the actions space A, i.e., an action a=π(b) for any belief b. A policy defines the system strategy for all possible situations it canfi nd [28]. The objective is to maximize the amount of reward earned over afi nite or infinite time horizon. The optimal policy, denoted by π^*, produces the highest expected reward value for each belief state, represented by the optimal value function V*(b). The value function V*(b) is solution to the Bellman's principle of optimality [29] as is shown below:

$$V^*(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s)R(s,a) + \gamma \sum_{z \in Z} p(z|b,a)V^*(r(b,a,z)) \right\} \quad (1)$$

The action that produces the highest or optimal Expected Value (EV) in the equation above, will be the chosen action a* for adaptation. The action a* will be executed and eventually a new observation z will be monitored.

The constant $\gamma \in [0, 1)$ is the discount factor, which expresses preferences for immediate rewards over future ones. By using the equation in (1), an agent searches for an optimal action a*=V*(b) at the current belief b. Therefore, the decision-making process in a POMDP involves (i) choosing an action a using a policy a=π(b) based on the current belief b (ii) updating the current belief b to b'=r(b,a,z) after receiving and observation z. The process then repeats [37].

RE-STORM is supported by an approximate POMDP solver [37] that allow us to overcome the curse of history during the decision-making based on new runtime evidence. Overcoming the curse of history means that treating all the possible future states while making projections is tractable.

3 REQUIREMENTS AND DECISION-MAKING WITH POMDPS

Next, we explain how POMDPs offer a frame for modelling NFRs trade-offs to underpine the decision-making process when new evidence is collected from the environment.

3.1 Motivating example

As an example to show the specification and representation of NFRs by using POMDPs, consider the fragment of a simple model of the robot vacuum cleaner for a domestic apartment [4, 34] shown in Fig. 2. The vacuum cleaner has as functionality to clean apartment and two NFRs; to avoid causing danger to people within the house (i.e., Avoid Tripping Hazard - ATH) and to be economical to run (i.e.,Minimization of Energy Costs - MEC). The functionality clean apartment can be satisfied by two different realization strategies; Clean at night (CN) or Clean when empty (CE). Cleaning at night partially denies tripping hazard avoidance but completely satisfies energy cost

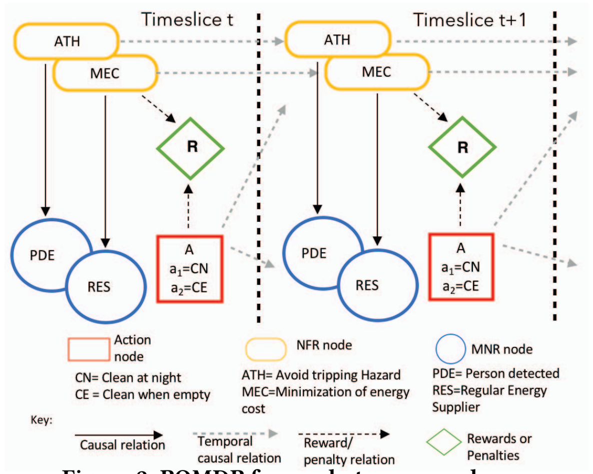


Figure 2: POMDP for a robot vacuum cleaner

minimization, while cleaning when empty partially denies energy cost minimization but completely satisfies tripping hazard avoidance.

To implement the scenario above, the model involves a monitoring infrastructure to monitor: (i) people at home and, (ii) the use of an alternative sources of energy. Person detected (PDE) is a monitorable to detect a person that could be in tripping hazard when the robot vacuum cleaner is working, while Regular Energy Supplier (RES) is a second monitorable to detect the use of a more expensive source of energy at the apartment.

During runtime, the POMDP model is kept in memory to support reasoning and to monitor the current satisficement of the system's NFRs [7]. Let us suppose that during the execution and when the vacuum cleaner is cleaning the apartment, the monitoring infrastructure senses that a person is at home. In this case, tripping hazard avoidance could be in disadvantage and the runtime reasoning engine (supported by the POMDP model) would evaluate the consequences and suggest an adaptation from cleaning at night strategy to cleaning when empty. In a different context, when the monitoring infrastructure senses the use of a more expensive source of energy, minimization of energy cost may be in disadvantage and again, the reasoning engine would evaluate the effects of actions and suggest the most suitable adaptation accordingly.

The focus of this paper is to present the mapping from NFRs onto POMDPs and we will use the vacuum cleaner example for this. The mapping is explained as follows.

3.2 Mapping from NFRs onto POMDPs

First, a POMDP model is designed to support a functionality and its related NFRs by providing:

- **NFR nodes**. Each NFR node is associated with a conditional probability distribution to represent its current satisficement (i.e., its current state in POMDP terms). A NFR node is influenced by the state of its parent nodes, i.e., an Action node and a NFR node, both at the previous time slice [4, 29] (See Fig. 2). NFR nodes are related to the states s ∈ S in a POMDP.
- **Monitorable (MNR) nodes**. They represent the observations collected from the environment at runtime. Each MNR node, similar to the NFR nodes, is associated with a conditional probability distribution that is influenced by the state of its parent nodes.
- **An Action node**. It represent a set of actions a ∈ A (i.e., adaptations) that a system could execute. The execution of an action may have an impact on the satisficement of the NFR nodes and the collected information from the environment by the MNR nodes (See Fig. 2). MNR nodes are related to the observations z ∈ Z in a POMDP.
- **A Reward node and its corresponding reward function**. It represents the reward obtained after performing the action a under the current state of the NFR nodes.

Next, we explain the mapping process from the elements above to the POMDP model and their dependencies.

3.2.1 From Requirements to POMDPs. Fundamentals concepts and equivalences in the mapping are explained next using a set of mapping rules.

a) **NFR nodes and their satisficement values**. NFR nodes represent the NFRs to be satisfied [10]. In theory, each NFR node has two possible states or values: True or False. However, it does not mean that the satisficement of a NFR is measured by only two absolute values. As the states in a POMDP, the states of the NFR nodes are not directly observable. What we have are beliefs about these states and therefore their levels of satisficement are described by the probability distribution P(NFR_i =True) and P(NFR_i = False), where NFR_i represents any NFR. We measure at runtime the current satisficement of a NFR node by using these probability distributions, which allow us to model the inherent uncertainty about the true state of the levels of satisficement of the NFRs underpined by the POMDP [28].

b) **NFR nodes and POMDP states s ∈ S**. NFR nodes represent states s ∈ S that a POMDP can achieve. Each state s in a POMDP is represented by a set of possible values (True or False) of the NFR nodes from NFR_1 to NFR_n, where n is the number of NFR nodes in the model. Given the identified relationship between NFR nodes and states s in a POMDP, we have the following mapping rule:

Mapping Rule 1. *A state s ∈ S represents a set of values (combinations of True or False) of the NFR nodes NFR_1,...,NFR_n. The values of the NFR nodes are not directly observable. They have associated a probability distribution.*

In the case of the example, two NFR nodes exist: Avoid Tripping Hazard (ATH) and Minimization of Energy Cost (MEC). The states s of the POMDP associated are shown in Table 1. The number

Table 1: Vacuum cleaner NFR nodes and states *s*

S	NFR_1=ATH	NFR_2=MEC
s_1	True	True
s_2	False	True
s_3	True	False
s_4	False	False

of states s ∈ S is represented as |S| and depends on the number of NFR nodes |NFR| and the number of values $|NFR_{sv}|$ for each NFR node. |S| is computed as follows:

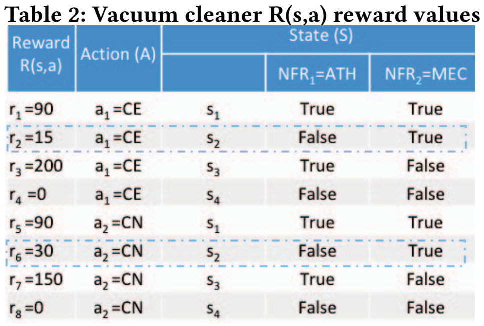
$$|S| = |NFR_{sv}|^{|NFR|} \quad (2)$$

In the example, we have 2 NFR nodes and there are 2 possible values: True and False per each NFR node, therefore, the number of possible states |S| is $(2)^2$=4.

c) **NFR nodes and the reward function R(s,a)**. A reward function is a scalar that assigns a cardinal scale to each decision made during a specific state of the system [15], i.e., each action a and state s in a POMDP indicating its desirability. The state s in a reward function R(s,a) is represented by the values of the NFR nodes (Following Mapping Rule 1).

Table 2 shows the rewards R(s,a) related to the NFR nodes: ATH and MEC of our example. We observe that given the same state s_2 there is a bigger weight (i.e., preference) over action a_2 (i.e., r_6=30) in relation to action a_1 (i.e., r_2=15). These weights are used by a POMDP using the Bellman's principle of optimality (See section 2.1.2, Equation 1).

Table 2: Vacuum cleaner R(s,a) reward values



Reward R(s,a)	Action (A)	State (S)		
			NFR_1=ATH	NFR_2=MEC
r_1=90	a_1=CE	s_1	True	True
r_2=15	a_1=CE	s_2	False	True
r_3=200	a_1=CE	s_3	True	False
r_4=0	a_1=CE	s_4	False	False
r_5=90	a_2=CN	s_1	True	True
r_6=30	a_2=CN	s_2	False	True
r_7=150	a_2=CN	s_3	True	False
r_8=0	a_2=CN	s_4	False	False

Given the above we present this mapping rule:

Mapping Rule 2. *The reward values R(s,a) represent the preferences over the execution of an action a ∈ A given a set of values of the NFR nodes.*

The number of possible reward values R(s,a) is represented as |R| and depends on the number of NFR nodes |NFR|, the number of state values $|NFR_{sv}|$ for each NFR node and the number of actions |A|. |R| is computed using the equation 3 as follows:

$$|R| = |NFR_{sv}|^{|NFR|} * |A| \tag{3}$$

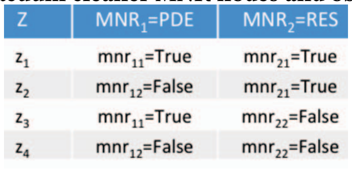
In the case of the vacuum cleaner, the system contains 2 NFR nodes, 2 possible values for each NFR node and 2 actions, therefore, the number of rewards $|R|$ would be $(2)^2$*2=8.

d) **MNR nodes and POMDP observations z ∈ Z.** MNR nodes provide the observations required for monitoring at runtime the satisficement levels of NFR nodes. In a POMDP the MNR nodes are represented as observations z from the environment. Each observation z ∈ Z represents a set of possible values obtained from MNR_1 to MNR_l, where l is the number of MNR nodes. Different from NFR nodes, MNR nodes may contain a bigger number of possible values, i.e., not only the values True and False. This characteristic allow us to enrich the quality of the information collected from the system's environment.

Mapping Rule 3. *An observation z ∈ Z represents a set of observation values of the MNR nodes.*

In the case of the example, two MNR nodes, PDE and RES, are involved. For simplicity, each of them has only two possible values: True and False. The observations z associated with these MNR nodes are shown in Table 3.

Table 3: Vacuum cleaner MNR nodes and observations



Z	MNR_1=PDE	MNR_2=RES
z_1	mnr_{11}=True	mnr_{21}=True
z_2	mnr_{12}=False	mnr_{21}=True
z_3	mnr_{11}=True	mnr_{22}=False
z_4	mnr_{12}=False	mnr_{22}=False

The number of observations z ∈ Z is represented as |Z| and depends on the number of MNR nodes |MNR| and the number of possible values $|MNR_{ov}|$ for each MNR node. The number of observations |Z| is computed as follows:

$$|Z| = \prod_{i=1}^{|MNR|} |MNR_{ov}| \tag{4}$$

In the case of the robot vacuum cleaner example, we have 2 MNR nodes and 2 possible values for each one, therefore the number of observations |Z| would be 2*2=4.

3.2.2 *Relationships among POMDP elements.*

They are represented by the transition and observation models of the system as well as the probability distributions related to them. We explain these concepts and their relationship to NFRs below.

a) **NFR nodes and the POMDP Transition model**. In a POMDP, the transition model T (s, a, s') = P(s' |s, a) represents the probability that the system makes a transition from state s to state s' when action a is executed in state s. In our approach, a state s represents a set of current values of NFR nodes (Mapping Rule 1), therefore, the transition model will represent the transition from a set of current values to a new one for the NFR nodes. NFR nodes can barely ever be labelled 100% satisfied or 100% unsatisfied in an unambiguous sense [5]. In the context of POMDPs, the satisficement of each NFR node is dictated by a probability distribution over its possible values. For instance, in Fig. 3, one possible probability distribution for the satisficement of the NFR node ATH at time slice t could be:

$P(ATH_t$=True)=0.32 and $P(ATH_t$=False)=0.68

However, at time slice t+1, based on the influence of its parent's nodes, we could get a different probability distribution:

$P(ATH_{t+1}$=True)=0.65 and $P(ATH_{t+1}$=False)=0.35

As is noticed above, the probabilities about the satisficement of NFR nodes may evolve at runtime, and this evolution is according to the following mapping rule.

Mapping Rule 4. *The transition model T (s, a, s') = P(s' |s, a), represents a system taking an action a under the current values of its related NFR nodes and updating the next time slice to a new set of values of its NFR nodes.*

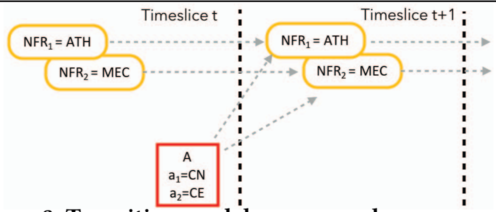


Figure 3: Transition model - vacuum cleaner example

Next, we present the transition model derived from mapping rule 4, to represent the model as a function of the NFR nodes:

$$T(s, a, s') = P(NFR'_1 ... NFR'_n | NFR_1 ... NFR_n, a) \tag{5}$$

By using conditional independence property [26] and Bayes theorem [17, 25], the transition model can also be factored into a product of smaller conditional distributions:

$$\begin{aligned} T(s, a, s') = P(NFR'_1 | NFR_1, a) P(NFR'_2 | NFR_2, a) \\ ... P(NFR'_n | NFR_n, a) \end{aligned} \tag{6}$$

Note that the term conditional independence property in the context above, we are not implying that the NFRs are required to be independent. RE-STORM is oblivous to the their independence. Let's use equation (6) in the example of the vacuum cleaner. Two possible actions: Clean when empty (CE) and Clean at night (CN) exist. They affect the NFR nodes Avoid tripping hazard (ATH) and Minimization of energy costs (MEC). The factored transition model for this example is:

T(s,a,s')=P(ATH'|ATH,a) P(MEC'|MEC,a)

As is observed in equation (6), the transition model requires factored conditional probabilities related to NFR nodes. The conditional probability tables (CPTs) for each factor of example are shown below in Tables 4 and 5.

Table 4: CPT ATH: P(ATH' |ATH,a)

NFR_1: Avoid Tripping Hazard (ATH)				
Action A_t		ATH_t	$P(ATH_{t+1}=False)$	$P(ATH_{t+1}=True)$
a_1=Clean when empty (CE)		True	0.45	0.55
a_1=Clean when empty (CE)		False	0.47	0.53
a_2=Clean at night (CN)		True	0.11	0.89
a_2=Clean at night (CN)		False	0.21	0.79

Key: ATH_t = ATH at time t ATH_{t+1} = ATH at time t+1 A_t = set of actions *a* at time t

Table 5: CPT MEC: P(MEC' |MEC,a)

NFR_2: Minimization of Energy Consumption (MEC)				
Action A_t		MEC_t	$P(MEC_{t+1}=False)$	$P(MEC_{t+1}=True)$
a_1=Clean when empty (CE)		True	0.25	0.75
a_1=Clean when empty (CE)		False	0.35	0.65
a_2=Clean at night (CN)		True	0.1	0.9
a_2=Clean at night (CN)		False	0.2	0.8

Key: MEC_t = MEC at time t MEC_{t+1} = MEC at time t+1 A_t = set of actions *a* at time t

The initial probabilities above are defined by domain experts [6, 16]. They evolve at runtime and are used to compute the probabilities of the states s in a POMDP.

b) **Probabilities of states s in a POMDP**. It was established in Mapping Rule 1 that a state s ∈ S represents a set of values of NFR nodes. To obtain the probability of any state s we should apply the joint probability property [17] over its related NFR nodes.

Mapping Rule 5. *The probability of a state s ∈ S in a POMDP corresponds to the joint probability over the current values of its related NFR nodes.*

For the case of the vacuum cleaner, the state s_2 in Table 1, represents the values False for the NFR node ATH and True for the NFR node MEC. By following Mapping Rule 5, the probability of the state s_2 is obtained using the equation:

P(s_2)= P(ATH=False)P(MEC=True).

However, there are several possibilities for solving the previous expression. In Table 4 we observe that P(ATH=False) could be equal to: 0.45, 0.47, 0.11 or 0.21, depending on the values of its parent's nodes. In Table 5 we observe that P(MEC=True) could be equal to: 0.75, 0.65, 0.9 or 0.8, depending also on the values of its parent's nodes.

The different possibilities mentioned above are visually explained in Fig 3. It shows that the NFR nodes at time t+1, depends on their parent's elements at time t. Based on the values of the parent's elements, we could get different effects, i.e., different valid probabilities P(ATH) and P(MEC) at time t+1. We call this relationship "parent's dependencies" for the transition model.

The number of parent's dependencies for the transition model is represented as $|PD_{TM}|$ and depends on the number of NFR nodes |NFR|, the number of values $|NFR_{sv}|$ for each NFR node and the number of actions |A|. The number $|PD_{TM}|$ is computed using the equation 7 as follows:

$$|PD_{TM}| = |NFR_{sv}|^{|NFR|} * |A| \qquad (7)$$

Table 6: Parent Dependencies Transition Model

Parent dependencies Transition model	Action (A)	State (S)	
		NFR_1=ATH	NFR_2=MEC
PD_{TM1}	a_1=CE	True	True
PD_{TM2}	a_1=CE	False	True
PD_{TM3}	a_1=CE	True	False
PD_{TM4}	a_1=CE	False	False
PD_{TM5}	a_2=CN	True	True
PD_{TM6}	a_2=CN	False	True
PD_{TM7}	a_2=CN	True	False
PD_{TM8}	a_2=CN	False	False

Table 7: States s given parent dependency

PD_{TM5}: Action a_t = CN ATH_t=True MEC_t=True			
NFR $nodes_{t+1}$		Joint Probability	State S_{t+1}
P(MEC)	P(ATH)	P(MEC,ATH)	
True=0.9	True=0.89	0.801	s_1
True=0.9	False=0.11	0.099	s_2
False=0.1	True=0.89	0.089	s_3
False=0.1	False=0.11	0.011	s_4
	Σ P(MEC,ATH)	1	

For the example, after using equation 7, eight different options of "parent's dependencies" exist and they are shown in Table 6. We also show in Table 7 the probabilities of the states s_1 to s_4 for one of these options: parent dependency PD_{TM5}. At this stage we are able to compute the probability distribution for the states s ∈ S as a result of the application of Mapping Rule 5. This probability distribution is a design-time input for the execution at runtime of the POMDP transition model that allows us to monitor the probability of satisficement of the NFR nodes by following Mapping Rule 4.

c) **MNR nodes and the POMDP observation model**. Observations inform about the current state of the world [28]. O(s', a, z) = P(z|s', a) denote the probability that the system observes z after executing action a and transitions to state s'. In our approach, an observation z represents a set of observation values of MNR nodes (See Mapping Rule 3). MNR nodes should tackle the uncertainty related to lack of confidence of sensor's reports [4], therefore their observation values can hardly ever be 100% confident. In the context of POMDPs, the observations of each MNR node are dictated by a probability distribution over its possible values. As an example, in Fig. 4 one possible probability distribution for the values of the MNR node PDE at time slice t+1 could be:

$P(PDE_{t+1}$=True)=0.12 and $P(PDE_{t+1}$=False)=0.88

The probabilities about the values of MNR nodes are influenced by their parent's nodes (i.e., NFR nodes and Action node) and may evolve at runtime. This evolution is according to the following mapping rule.

Mapping Rule 6. *O(s', a, z) = P(z|s', a) represents a system getting MNR nodes observations under the current state of their related NFR nodes and after taking action "a" at the previous time slice.*

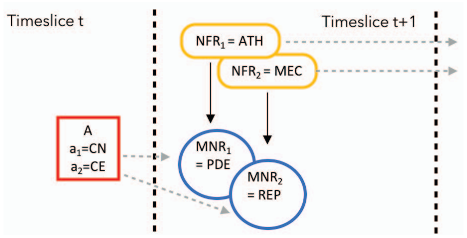


Figure 4: Observation model - Vacuum Cleaner

The observation model derived from mapping rule 6, is as follows:

$$O(z, a, s') = P(MNR_1...MNR_l|NFR'_1...NFR'_n, a) \qquad (8)$$

As with the transition model, the observation model can also be factored into a product of smaller conditional distributions.

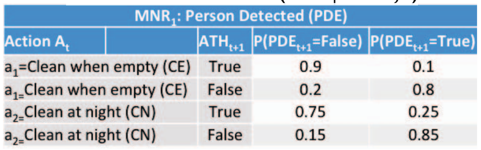
$$O(z, a, s') = P(MNR_1|NFR_1...NFR_n, a)$$
$$P(MNR_2|NFR_2...NFR_n, a)...P(MNR_l|NFR_n...NFR_n, a) \qquad (9)$$

In the case of the example of the vacuum cleaner, two NFR nodes: Avoid tripping hazard (ATH) and Minimization of energy costs (MEC) exist, that affect the MNR nodes Person detected (PDE) and Regular Energy Supplier (RES) respectively. The factored observation model for this example is:

O(z,a,s')=P(PDE|ATH',a) P(RES|MEC',a)

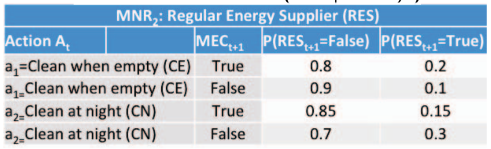
As is observed in (9), the observation model requires factored conditional probabilities related to the MNR nodes. The conditional probability tables (CPTs) for each factor of the vacuum cleaner example are shown below in Tables 8 and 9.

Table 8: CPT PDE: P(PDE|ATH',a)



MNR$_1$: Person Detected (PDE)			
Action A_t	ATH_{t+1}	P(PDE_{t+1}=False)	P(PDE_{t+1}=True)
a_1=Clean when empty (CE)	True	0.9	0.1
a_{1-}Clean when empty (CE)	False	0.2	0.8
a_{2-}Clean at night (CN)	True	0.75	0.25
a_{2-}Clean at night (CN)	False	0.15	0.85

Table 9: CPT RES: P(RES|MEC',a)



MNR$_2$: Regular Energy Supplier (RES)			
Action A_t	MEC_{t+1}	P(RES_{t+1}=False)	P(RES_{t+1}=True)
a_1=Clean when empty (CE)	True	0.8	0.2
a_{1-}Clean when empty (CE)	False	0.9	0.1
a_{2-}Clean at night (CN)	True	0.85	0.15
a_{2-}Clean at night (CN)	False	0.7	0.3

The initial probabilities above are used to compute the probabilities of the observations in a POMDP as is explained below.

d) **Probabilities of observations z in a POMDP**. It was established in Mapping Rule 3 that an observation z ∈ Z represents a set of current observation values of the MNR nodes. To obtain the probability of any observation z we should apply the joint probability property [17] over its related MNR nodes.

Mapping Rule 7. *The probability of an observation z ∈ Z in a POMDP corresponds to the joint probability over the current observation values of its related MNR nodes.*

For the case of the example, the state z_3 in Table 3, represents the observation values True for the MNR node PDE and False for the MNR node RES. By following Mapping Rule 7, the probability of the observation z_3 can be obtained from the following equation:

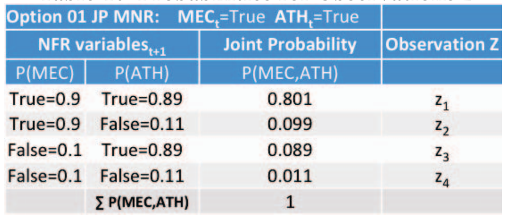
P(z_3)= P(PDE=True)P(RES=False).

Fig. 4 shows that the MNR nodes at time t+1, depends on their parent's elements at time t and time t+1. Based on the values of the parent's elements different effects are obtained, i.e., different valid probabilities P(PDE) and P(RES) at time t+1. This relationship is called 'parent's dependencies" of the observation model. The number of these parent's dependencies is represented as $|PD_{OM}|$ and depends on the number of NFR nodes |NFR|, the number of satisficement values $|NFR_{sv}|$ for each NFR node and the number of actions |A|. The number $|PD_{OM}|$ is computed using the equation 10:

$$|PD_{OM}| = |NFR_{sv}|^{|NFR|} * |A| \qquad (10)$$

In the case of our example, after using equation 10, similar to the transition model, eight different options of "parent's dependencies" exist. The probabilities of the observations z_1 to z_4 for one of these options are shown in Table 10.

Table 10: Probabilities for Observations z



Option 01 JP MNR: MEC_t=True ATH_t=True			
NFR variables$_{t+1}$		Joint Probability	Observation Z
P(MEC)	P(ATH)	P(MEC,ATH)	
True=0.9	True=0.89	0.801	z_1
True=0.9	False=0.11	0.099	z_2
False=0.1	True=0.89	0.089	z_3
False=0.1	False=0.11	0.011	z_4
	Σ P(MEC,ATH)	1	

4 CONCLUDING REMARKS

We argue that a SAS should exhibit explicit treatment of uncertainty by evaluation of new evidence using techniques such as machine learning. In this paper we have introduced RE-STORM, to assess the consequences of new evidence over the satisficement of the NFRs. RE-STORM is based on Bayesian methods and decision theory to account for uncertainty. RE-STORM works towardsfilling the lack of a time-history wise scalable platform to support NFR specification and runtime trade-off of the satisficement of NFRs to drive the decision-making in SASs. RE-STORM deals with uncertainty in an explicit way and updates its definition over time as new evidence arrives keeping consistency with the history of the execution. During runtime, RE-STORM learns and updates the probabilities of the satisficement level for each NFR (i.e., beliefs) over time when new evidence becomes available at runtime.

In this paper, we have mapped the decision-making of a SAS, including the specification of the trade-off between NFRs, onto a decision-problem solved by a POMDP. The POMDP implementation is the DESPOT algorithm [37]. Early experimental results performed using an remote data mirroring simulator [23] and other domains [8, 22] show that the decisions made about the trade-off of the satisficement levels of the NFRs are compatible with other experiences [1, 4, 5] and look sound. Further, RE-STORM presents a scalable solution with respect to our work in [4, 5]. Using the risk preferences in RE-STORM, we are working on a novel approach for the runtime update of preferences when the current preferences are not suitable to emerging situations detected at runtime [24].

REFERENCES

[1] A. Ramirez, A. J., andCheng , B. âĂIJa taxonomy of uncertainty for dynamically adaptive system,âĂİ. *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on, june 2012, pp. 99 âĂŞ108.* (2012).

[2] A. Somani, N. Ye, D. H., andLee , W. Despot: online pomdp planning with regularization. *In Advances in Neural Information Processing Systems (NIPS)* (2013).

[3] Bai, H., Cai, S., Ye, N., Hsu, D., andSun , W. Intention-aware online pomdp planning for autonomous driving in a crowd. *In Proc. IEEE Int. Conf. on Robotics & Automation* (2015).

[4] Bencomo, N., andB elaggoun, A. Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks. In *REFSQ - Best Paper Award* (2013).

[5] Bencomo, N., Belaggoun, A., andIssarny , V. Dynamic decision networks to support decision-making for self-adaptive systems. In *(SEAMS)* (2013).

[6] Bencomo, N., Cleland-Huang, J., Guo, J., andHarrison , R. First international workshop on artificial intelligence for requirements engineering (aire 2014). In *RE 2014*. IEEE, 2014.

[7] Blair, G., Bencomo, N., andFrance , R. B. Models@ run. time. *Computer 42*, 10 (2009), 22–27.

[8] Bocanegra, J., Garcia-Paucar, L., Pavlich-Mariscal, J., andBencomo , N. Improving the decision-making support in context-aware applications: The case of an adaptive virtual education learning management system. *XXI Ibero-american Conference on Software Engineering* (2018).

[9] Cheng, B. H., and et al. Software engineering for self-adaptive systems. Springer-Verlag, Berlin, Heidelberg, 2009, ch. Software Engineering for Self-Adaptive Systems: A Research Roadmap, pp. 1–26.

[10] Chung, L., Nixon, B. A., Yu, E., andMylopoulos , J. *Non-Functional Requirements in Software Engineering*, vol. 5. Springer, 1999.

[11] Elahi, G., and Yu, E. Requirements trade-offs analysis in the absence of quantitative measures: A heuristic method. In *Proceedings of the 2011 ACM Symposium on Applied Computing* (New York, NY, USA, 2011), SAC '11, ACM, pp. 651–658.

[12] Esfahani, N., Kouroshfar, E., andM alek, S. Taming uncertainty in self-adaptive software. In *Proc.of the 19th ACM SIGSOFT Symp FSE* (2011), no. 11.

[13] Esfahani, N., andMalek , S. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems 2 (SEfSAS 2)*. Springer-Verlag, 2012.

[14] H. Kurniawati, D. H., andLee , W. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. *In Proc. Robotics: Science and Systems* (2008).

[15] Horvitz, E., Breese, J., andHenrion , M. Decision theory in expert systems and artificial intelligence. *Int. Journal of Approximate Reasoning 2, 247âĂŞ302 (1988)* (1998).

[16] Jureta, I. J., Borgida, A., Ernst, N. A., andMylopoulos , J. The requirements problem for adaptive systems. *ACM Trans. Manage. Inf. Syst. 5*, 3 (Sept. 2014), 17:1–17:33.

[17] Koller, D., andFriendman , N. Probabilistic graphical models: Principles and techniques.

[18] Krupitzer, C., Roth, F. M., VanSyckel, S., Schiele, G., andBecker , C. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing 17, Part B* (2015), 184 – 206.

[19] Kurniawati, H., and Yadav, V. An online pomdp solver for uncertainty planning in dynamic environment. *Proc. Int. Symp. on Robotics Research* (2013).

[20] Letier, E., Stefan, D., andBarr , E. T. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of ICSE* (New York, NY, USA, 2014), ICSE 2014, ACM, pp. 883–894.

[21] Liaskos, S., McIlraith, S. A., Sohrabi, S., andMylopoulos , J. Representing and reasoning about preferences in requirements engineering. *Requir. Eng. 16*, 3 (Sept. 2011), 227–249.

[22] Paucar, L. H. G., andBencomo , N. Runtime Models Based on Dynamic Decision Networks : Enhancing the Decision-making in the Domain of Ambient Assisted Living Applications. *MRT - Models@run.time at MODELS 2016, Saint-Malo, France* (2016).

[23] Paucar, L. H. G., andBencomo , N. The Reassessment of Preferences of Non-Functional Requirements for Better Informed Decision-making in Self-Adaptation. *AIRE - 3rd International Workshop on Artificial Intelligence for Requirements Engineering. Beijing* (2016).

[24] Paucar, L. H. G., Bencomo, N., andF ung Yuen, K. K. Juggling Preferences in a World of Uncertainty. *RE NEXT 2017, Lisbon, Portugal* (2017).

[25] Pearl, J. Probabilistic reasoning in intelligent systems: Networks of plausible inference. *Morgan Kaufmann* (1988).

[26] Poupart, P. Exploiting structure to efficiently solve large scale partially observable markov decision processes. *Thesis for the degree of Doctor of Philosophy. Graduate Department of Computer Science. University of Toronto* (2005).

[27] Ramirez, A., Cheng, B., Bencomo, N., andS awyer, P. Relaxing claims: Coping with uncertainty while evaluating assumptions at run time. *MODELS* (2012).

[28] Ross, S., Pineau, J., Paquet, S., andChaib-Draa , B. Online planning algorithms for pomdps. *J. Artificial Intelligence Research, 32(1), 663âĂŞ704* (2008).

[29] Russell, S. J., andNorvig , P. *Artificial intelligence - a modern approach: the intelligent agent book.* Prentice Hall series in artificial intelligence. Prentice Hall, 1995.

[30] Salama, M., Bahsoon, R., andBencomo , N. Managing trade-offs in self-adaptive software architectures: A systematic mapping study. In *Managing trade-offs in adaptable software architectures*, I. Mistrï£¡k, N. Ali, J. Grundy, R. Kazman, and B. Schmerl, Eds. Elsevier, 2016.

[31] Salehie, M., and Tahvildari, L. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst. 4*, 2 (May 2009), 14:1–14:42.

[32] Sawyer, P., Bencomo, N., Whittle, J., Letier, E., andFinkelstein , A. Requirements-aware systems: A research agenda for RE for self-adaptive systems. In *Proceedings of RE* (Washington, DC, USA, 2010), RE '10, IEEE.

[33] Silver, D., andVeness , J. Monte-carlo planning in large pomdps. *Advances in Neural Information Processing Systems (NIPS)* (2010).

[34] WadeB rittain, R., and D'Ambrosio, B. Autonomous vacuum cleaners must be bayesian. *AAAI Technical Report FS-93-03.* (1993).

[35] Welsh, K., andS awyer, P. âĂIJunderstanding the scope of uncertainty in dynamically adaptive system,âĂİ. *REFSQ* (2010).

[36] Welsh, K., Sawyer, P., andBencomo , N. Towards requirements aware systems: Run-time resolution of design-time assumptions. In *ASE* (2011), pp. 560–563.

[37] Ye, N., Somani, A., Hsu, D., andLee , W. S. Despot: Online pomdp planning with regularization. *Journal of Artificial Intelligence Research 58 (2017) 231-266* (2017).

[38] Yuan, E., Esfahani, N., andMalek , S. A systematic survey of self-protecting software systems. *ACM Trans. Auton. Adapt. Syst. 8*, 4 (Jan. 2014), 17:1–17:41.