Decision Making for Self-Adaptation Based on Partially Observable Satisfaction of Non-Functional Requirements

LUIS GARCIA, SEA Group, Aston University, Birmingham, UK
HUMA SAMIN, AIHS Group, Durham University, Durham, UK
NELLY BENCOMO, AIHS Group, Durham University, Durham, UK

Approaches that support the decision-making of self-adaptive and autonomous systems (SAS) often consider an idealized situation where (i) the system's state is treated as fully observable by the monitoring infrastructure, and (ii) adaptation actions are assumed to have known, deterministic effects over the system. However, in practice, the system's state may not be fully observable, and the adaptation actions may produce unexpected effects due to uncertain factors. This article presents a novel probabilistic approach to quantify the uncertainty associated with the effects of adaptation actions on the state of a SAS. Supported by Bayesian inference and POMDPs (Partially-Observable Markov Decision Processes), these effects are translated into the satisfaction levels of the non-functional requirements (NFRs) to, therefore, drive the decision-making. The approach has been applied to two substantial case studies from the networking and Internet of Things (IoT) domains, using two different POMDP solvers. The results show that the approach delivers statistically significant improvements in supporting decision-making for SAS.

CCS Concepts: • **Computing methodologies** → **Uncertainty quantification**; **Partially-observable Markov decision processes**; • **Mathematics of computing** → *Bayesian networks*; • **Theory of computation** → *Reinforcement learning*; • **Software and its engineering** → **Extra-functional properties**; • **Computer systems organization** → **Self-organizing autonomic computing;**

Additional Key Words and Phrases: Non-functional requirements, decision making, uncertainty, POMDPs, self-adaptation

ACM Reference Format:
Luis Garcia, Huma Samin, and Nelly Bencomo. 2024. Decision Making for Self-Adaptation Based on Partially Observable Satisfaction of Non-Functional Requirements. *ACM Trans. Autonom. Adapt. Syst.* 19, 2, Article 11 (April 2024), 44 pages. https://doi.org/10.1145/3643889

1 INTRODUCTION

Self-adaptive and autonomous systems (SAS) are expected to apply decision-making techniques under uncertainty. As such, the decision-making technique of an SAS dictates the execution of adaptation actions according to unanticipated events [5]. Such adaptation actions impact the state of the SAS to maintain the required satisfaction levels of the **non-functional requirements (NFRs)**. Further, tradeoff analysis of the NFRs is crucial in establishing the expected balance among them [15, 29].

Authors' addresses: L. Garcia, SEA Group, Aston University, Aston Street, Birmingham, B4 7ET, UK; e-mail: garcial2@aston.ac.uk; H. Samin and N. Bencomo (Corresponding author), AIHS Group, Durham University, Upper Mountjoy Campus, Stockton Rd, Durham, DH1 3LE, UK; e-mails: huma.samin@durham.ac.uk, nelly.bencomo@durham.ac.uk.

ACM 1556-4665/2024/04-ART11
https://doi.org/10.1145/3643889

While many decision-making techniques assume that adaptation actions have invariant effects on the state of the SAS, we argue that such effects may be stochastic and can change over time according to environmental and contextual fluctuations. As an illustration, let us focus on the effects of adaptation actions on the system properties of an SAS, such as performance or reliability, which may vary over time due to changes in the environmental context. For example, the action of sending a data package through a network has an expected delivery rate with a specific effect on the reliability of the system. However, the delivery rate may fluctuate due to uncertain factors such as software or hardware failures. Further, existing decision-making techniques frequently assume full observability of the effects of adaptation actions based on the monitoring infrastructure [4, 7, 8, 20, 34–36, 53]. However, the system's state may not be fully observable in practice [23, 61, 66].

To account for the possible fluctuation of the effects of adaptation actions and partial observability of the state of the SAS, we have developed RE-STORM: Requirements Tradeoffs for self-adaptation using Partially Observable Markov Decision Processes (POMDPs) [19, 42]. We use POMDPs to (a) allow the effects of adaptation actions to be modelled using probability distributions, as opposed to having fixed effects on the system over time, and (b) treat the system's state as partially observable by a monitoring infrastructure. The system's state is represented by the levels of satisfaction associated with the NFRs. The POMDP balances the tradeoff of the conflicting NFRs over time [42, 72].

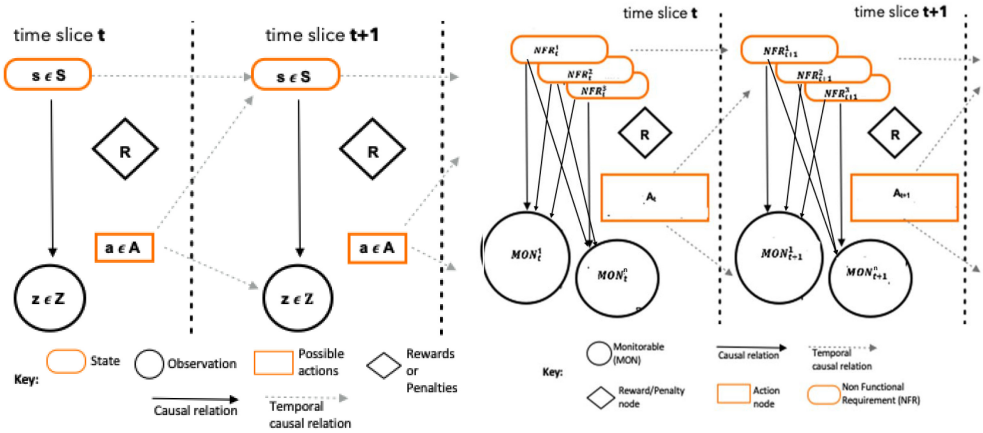
The following are the contributions of the article:

(i) *A decision-making technique for self-adaptation based on partially observable NFRs.* The system's state of the POMDP, which is not directly observable, is modelled based on the NFRs of a SAS.

(ii) *Quantification of the uncertainty associated with the effects of executing adaptation actions on the levels of satisfaction of the NFRs in the running system.* After the execution of an adaptation action, observations related to the satisfaction of NFRs are collected from the system monitoring infrastructure. Based on those observations, Bayesian inference is used to deduce the current effects on the state of the system. These effects are quantified as probabilities that represent the current satisfaction level of the NFRs of the SAS.

We have used two substantial case studies from the networking application domain [59, 64] and the **Internet of Things (IoT)** domain [30]. For each case study, different dynamic contexts have been used to show how RE-STORM supports decision-making under uncertainty while improving the level of satisfaction of the NFRs. The results also show how RE-STORM improves the tradeoff among NFRs. We have applied RE-STORM using two different POMDPs solvers; Despot [54] and Perseus [50], showing its applicability. Our results highlight how, under dynamic contexts, RE-STORM provides a quantification of the effects of executing adaptation actions on the running system, which leads to the necessary update of the utility values that will match the new context. The evaluation performed supports the conclusion that the approach delivers a statistically significant improvement in the decision-making for SAS.

This article is organized as follows: Section 2 explains the baseline concept of POMDPs. Section 3 presents our proposed approach of RE-STORM. Section 4 describes the approach using an illustrative case of Remote Mirroring. Section 5 describes the decision-making process offered by RE-STORM and how it is driven by partially observable NFRs. Section 6 present the details of the evaluations. Section 7 discusses the main findings and validation of results. In Section 8, we provide a comparison with related work. Finally, in Section 9, we conclude this article and outline future research opportunities.



(a) POMDPs for sequential decision-making (b) RE-STORM

Fig. 1. General POMDPs and RE-STORM.

2 POMDPS AND DECISION-MAKING UNDER UNCERTAINTY

In self-adaptation, uncertainty can be defined as a system state of incomplete or inconsistent knowledge such that it is not possible to know which adaptation action holds at a specific point in time [1]. Uncertainty may arise, for example, due to missing or ambiguous requirements, erroneous assumptions, unpredictable behaviour in the execution environment, or incomplete information obtained by potentially imprecise or unreliable sensors in the monitoring infrastructure [67]. Uncertainty may also affect a software system, either at the requirements, design, or execution phases of the software life cycle.

RE-STORM contributes to dealing with uncertainty during the execution phase of a system by using a Partially Observable Markov Decision Process (POMDP) to quantify the effects of the adaptation actions in the form of probabilities that model the current satisfaction level of the NFRs of the SAS.

A POMDP provides a principled approach to model sequential decision-making problems to make rational decisions in the face of uncertainty within a changing environment [73]. In RE-STORM, a POMDP keeps the up-to-date quantification of uncertainty about the effects of the adaptation actions on the running system. Figure 1 shows POMDP and RE-STORM for decision-making is SAS.

A POMDP is represented as a tuple <**S,A,R,T,O,Z,**γ>. The $\boldsymbol{S}$ represents the state space, i.e., a set of distinct states $\boldsymbol{s} \in \boldsymbol{S}$ the system could reach. $\boldsymbol{A}$ represents the space of actions. The system seeks to influence its state by executing actions from the set $\boldsymbol{A}$. The system's goal is to choose actions in such a way that desirable states $\boldsymbol{s} \in \boldsymbol{S}$ should be frequently visited. Desirable states are determined by the reward function $\mathbf{R: SxA \rightarrow R(s,a)}$, in other words, the system gets a reward $\boldsymbol{R(s,a)}$ for taking action $\boldsymbol{a}$ and arriving at the new state $\boldsymbol{s} \in \boldsymbol{S}$. A POMDP allows uncertain action effects to be modelled. This behaviour is represented by the transition function $\mathbf{T:SxAxS \rightarrow [0,1]}$, which implies that the system has a certain probability of making a transition to any state $\boldsymbol{s} \in \boldsymbol{S}$ as a result of executing an adaptation action. The stochastic nature of the action effects is described as the conditional probability function $\boldsymbol{T(s, a, s') = P(s' \mid s, a)}$ where, at each time slice, the system takes action $\boldsymbol{a} \in \boldsymbol{A}$ to move from a state $\boldsymbol{s} \in \boldsymbol{S}$ to a new state $\boldsymbol{s'} \in \boldsymbol{S}$. Furthermore, in a POMDP, states $\boldsymbol{s} \in \boldsymbol{S}$ are not directly observable. Instead, observations $\boldsymbol{z} \in \boldsymbol{Z}$ are received. This behaviour is represented

by the observation function **O:SxAxZ → [0,1]**. The conditional probability function ***O(s′, a, z)*** = ***P(z | s′, a)*** describes the system's probability of observing $\boldsymbol{z} \in \boldsymbol{Z}$ given action $\boldsymbol{a}$ was performed, and the resulting state was $\boldsymbol{s'}$, which is not directly observable. Observations $\boldsymbol{z} \in \boldsymbol{Z}$ corresponds to features of the environment directly perceptible by system sensors. We use observations $\boldsymbol{z} \in \boldsymbol{Z}$ to infer the state $\boldsymbol{s} \in \boldsymbol{S}$ as is depicted below. The $\gamma \in [0, 1)$ is the discount factor, expressing preferences for immediate rewards over future ones.

Bayesian Inference and Quantification of Uncertainty. Given that in POMDPs, the system's state $s \in S$ is not directly observable, a **belief** over possible states of the system is maintained. Let b_{t-1} be the belief at time $t-1$. If the system takes action a at time $t-1$ and receives observation z at time t, then Bayesian inference is applied to quantify the uncertainty as the new belief b_t about the state s' at time t:

$$b_t(s') = \eta \mathbf{O(s', a, z)} \sum_{s \in S} \mathbf{T\ (s, a, s')} b_{t-1}(s) \tag{1}$$

where η is a normalizing constant [54]. A **belief** is a probability distribution about the current state $s \in S$ of the system. Through Markov property, a **belief** represents the entire system state history or trajectory, in terms of its past observations and actions [9].

Furthermore, during the POMDP-based decision-making, the decision-making agent tries to find a policy π. A policy π defines the system strategy for all possible situations it can find [46]. In terms of a POMDP, a policy $a = \pi(b)$ represents a mapping that specifies the action a at the current **belief** b about the system's state $s \in S$. The goal is to maximise the expected value EV, i.e., the possible amount of reward earned under the current belief b as is shown below:

$$EV_\pi(b) = \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(b_t)) | b_0 = b \tag{2}$$

Hence, POMDPs provide reasoning and decision-making over time, using partial knowledge (i.e., beliefs) of the states $s \in S$ of a running system.

3 RE-STORM: REQUIREMENTS TRADEOFF FOR SELF-ADAPTATION USING POMDPS

This section presents the proposed approach of RE-STORM, presented in Figure 1(b), which uses POMDP to support tradeoffs of the requirements during decision-making in SAS. To achieve the satisfaction of NFRs, a system performs adaptation actions $\boldsymbol{A}$. These actions can have different effects (good or bad) on the NFRs' satisfaction levels. The NFRs cannot be labelled as fully satisfied or fully violated: the satisfaction levels of NFRs cannot be represented as an absolute value of True or False because of the lack of crispness in the nature of satisfaction of NFRs [12, 13, 21, 68]. However, the satisfaction levels can be modelled as probability distributions such as P(NFR = True). As such, an NFR is considered satisfied if it meets an acceptability threshold defined by the experts [58]. For example, in an **Internet of Things (IoT)** network, the satisfaction level of the NFR such as **Maximization of Reliability (MR)** can be specified as P(MR = True) = 0.8 or P(MR = True) = 0.3 for a given environmental context. The MR can be considered highly satisfied if the P(MR = True) >= 0.7, where 0.7 can be regarded as an acceptability threshold requirement.

In the case of RE-STORM, each state in a POMDP represents the set of combinations of satisfaction values of NFRs. As states in POMDPs are not directly observable, a belief (i.e., a probability) over each state is maintained by the POMDP. Therefore, the satisfaction levels of NFRs are specified as marginalized probability distributions P(NFRi = True) where NFRi is a member of the set of NFRs. These probabilities are used to specify whether the satisfaction level of an NFR meets the acceptability threshold.

Considering the above, we provide Definitions 1–3 as follows:

Definition 1. In RE-STORM, the state $s \in S$ of the system represents the combinations of satisfaction values (True or False) of the system's quality goals, i.e., its NFRs, which are not directly observable.

Definition 2. In RE-STORM, the stochastic effects of the execution of an adaptation action $a \in A$ on a system, represented by the conditional probability P(s' |s, a), are quantified as the **belief** about the state $s' \in S$ of the system.

Definition 3. In RE-STORM, an NFR is considered satisfied if the current belief about the satisfaction of the NFR is higher than the acceptability threshold.

When using POMDPs in RE-STORM, the reward values R(s,a) at design time reflect the initial stakeholders' preference values for executing an adaptation action based on the satisfaction of NFRs. During execution, these initial preference values are re-assessed according to new evidence collected at runtime. The reassessment can prompt new perceived utility values by stakeholders, which better fit the newly found context (i.e., the stakeholders' preferences have changed). Based on the above, we present Definition 4 as follows:

Definition 4. In RE-STORM, the reward values R(s,a) correspond to the utility value of arriving at the new state s ∈ S after executing an adaptation action a ∈ A.

A SAS based on RE-STORM uses Bayesian inference to update the **belief** about the new state s' ∈ S, as evidence arrives [i.e., new observations $z \in Z$ collected after executing adaptation actions $a \in A$]. Furthermore, based on Definition 2, the policy π, represented by the expression a = π(b), defines the adaptation action taken by a SAS based on the current belief b about the satisfaction of its NFRs. The goal is to select the adaptation action that maximises *EV*.

The definitions to represent NFRs and their evolution over time, supported by the POMDP, are presented next. Specifically, it is shown how the transition function T(s', a, s) and the observation function O(s', a, z) of the POMDP have been extended to support the modelling and treatment of partially observable NFRs.

3.1 NFRs and the POMDP Transition Function

In a POMDP, the transition function T (s, a, s') = P(s' |s, a) is a conditional probability, which represents the transition of a system from state s to state s' when action a has been executed under the current state s. Based on the above, we present the next definition:

Definition 5. The transition function T (s, a, s') = P(s' |s, a), represents the probability of a system arriving at a satisfaction state of NFRs **s'** in the next time slice, if the system takes an action **a** under the current satisfaction state **s** of its NFRs.

In accordance with Definition 1, the states in the RE-STORM are represented as combinations of satisfaction levels of NFRs. When an adaptation action is performed, the transitions of the NFRs' states occur. Therefore, for each NFR, the transition model represents the transition from the current satisfaction state of an NFR to its new state [66]. Hence, to represent the transitions between the states of the individual NFRs, the conditional independence property [75] and Bayes theorem [33] are used. The NFRs are not independent; however, the transitions of states of NFRs are independent. This allows us to factor the transition model into a product of marginal conditional distributions to represent the transitions of the satisfaction state of the individual NFRs. Given the current satisfaction state of an NFR and action, the transition to the next state with respect to each NFR is independent of the transition in the satisfaction state of other NFRs.

Considering the above, we derive a transition function as a function of the NFRs of the system:

$$\begin{aligned} T(s,a,s') &= P(NFR_{t+1}^{(1)} \dots NFR_{t+1}^{(n)} | NFR_t^{(1)} \dots NFR_t^{(n)}, a_t) \\ &= P(NFR_{t+1}^{(1)} | NFR_t^{(1)}, a_t) P(NFR_{t+1}^{(2)} | NFR_t^{(2)}, a_t) \\ &\quad \dots P(NFR_{t+1}^{(n)} | NFR_t^{(n)}, a_t) \end{aligned} \tag{3}$$

In Equation (3), $NFR_t^{(i)}$ and $NFR_{t+1}^{(i)}$ represent the non-functional requirement "i" at the time slices t and t+1 respectively, $\forall_i \in [1, n]$. Based on Bayes' theorem [71, 74], the transition function T(s, a, s') is factored as a product of conditional distributions [75].

3.2 Monitoring Variables (MON) and the POMDP Observation Function

In POMDPs, the state of the system s ∈ S is not directly observable. Instead, monitorable variables (i.e., MON variables) are used to collect observations of the system's state. The values of MON variables are represented as observations $z \in Z$ from the environment. These observations are used in Equation (1) to compute a belief about the real system's state. Based on the above, we present the following definition:

Definition 6. The observation function *O(s', a, z) = P(z|s', a), represents the probability of observation* ***z*** *collected from the environment, given an action* ***a*** *was executed and as a result the state* ***s'*** *was achieved.*

The observation function derived from Definition 6, is represented as follows:

$$\begin{aligned} O(s',a,z) = P(MON_{t+1}^1 | NFR_{t+1}^1 \dots NFR_{t+1}^n, a_t) P(MON_{t+1}^2 | NFR_{t+1}^2 \dots NFR_{t+1}^n, a_t) \dots \\ P(MON_{t+1}^l | NFR_{t+1}^n \dots NFR_{t+1}^n, a_t) \end{aligned} \tag{4}$$

In Equation (4), $MON_{t+1}^{(j)}$ represents the MON variable "j" at the time slice t+1, $\forall_j \in [1, l]$.

Next, we present the case of a Remote Data Mirroring Network to support the explanation of the details of the approach.

4 ILLUSTRATIVE CASE: REMOTE DATA MIRRORING

We have applied the RE-STORM approach to the two case studies associated with Remote Mirroring [37] and Internet of Things [30] domains. In this section, we use the case of **Remote Data Mirroring (RDM)** to illustrate the approach.

The RDM SAS [19] is composed of data servers and network links. It must replicate and distribute data efficiently along with providing assurance that spread data is not lost or corrupted [37]. Each link of the network has an associated operational cost and a measurable throughput, latency and loss rate that are used to determine the reliability, cost and performance of the RDM system. The goal here is to satisfy the NFRs *Minimization of Cost (MC)*, *Maximization of Performance (MP),* and *Maximization of Reliability (MR)* under uncertain environmental conditions of link failures and varying ranges of bandwidth consumption [37]. To satisfy these NFRs, the network is required to continuously take *adaptive actions* by switching between the topologies of *Minimum Spanning Tree (MST)* and *Redundant Topology (RT). An MST Topology uses the minimum possible number of network links to transfer data among different remote servers (i.e., mirrors). In contrast, an RT topology simultaneously uses numerous redundant network link paths for the transmission of information across the servers. The satisfaction levels related to the performance, reliability and operational costs of the RDM are determined according to tradeoffs which are based on the application of the topologies as follows:*

— An RT Topology offers higher levels of reliability in comparison to MST topology. However, maintaining an RT topology may be expensive in some contexts, given the additional cost of required bandwidth consumption, and as a result, the performance of the system will also be affected.
— On the other hand, MST Topology offers lower operational costs and higher levels of performance than the RT topology. However, it negatively affects reliability.

4.1 Stochastic Effects of Adaptation Actions in the RDM SAS

The stochastic effects of the topologies RT and MST over the state of the RDM SAS are determined according to the following tradeoffs that drive the decision-making: (i) RT Topology *offer higher levels of reliability* than MST topology: producing higher costs while reducing performance. Conversely, (ii) MST Topology *offer higher levels of performance* and *lower levels of cost* than RT topology: the reliability of the system can therefore be jeopardised.

These stochastic effects have been modelled by the transition function T (s, a, s') presented in Section 3.1.

4.2 Partial Observability in the System

In Figure 1(b), the NFRs NFR_t^1, NFR_t^2, and NFR_t^3 are not directly observable. Instead, observations are obtained by using monitoring variables (called *MON variables*). Three MON variables are specified in the RDM SAS: **Ranges of Bandwidth Consumption (RBC)** (i.e., RBC <x, RBC in [x,y) and RBC>=y), **Active Network Links (ANL)** (i.e., ANL<r, ANL in [r,s) and ANL >=s) and **Total Time for Writing (TTW)** (i.e., TTW<f, TTW in [f,g) and TTW >=g). TTW is a performance measure which considers the time to write each copy of data on each remote site [37]. In the RDM SAS, the pair values (x, y); (r, s); and (f, g) represent range boundaries for the MON variables RBC, ANL, and TTW, respectively. The relationships between the MON variables and the NFRs MC, MR, and MP have been modelled by the observation function O(s', a, z) presented in Section 3.2. They can be summarized as follows:

— In case of *Ranges of Bandwidth Consumption* (RBC), the *lower the monitored values* are, the *greater* the satisfaction of *Minimization of Cost* (MC) is (same relationship exists between TTW and the belief about the satisfaction of MP).
— In case of *Active Network Links* (ANL), the *higher the monitored values* are, the *greater* the satisfaction of *Maximization of Reliability* (MR) is.

Both the transition functions and observations obtained from monitoring variables are taken into account to estimate the belief about the satisfaction of the NFRs of the RDM SAS by using Bayesian inference (as presented in Section 2).

4.3 Service Level Agreements (SLAs)

To understand the **Service Level Agreements (SLAs)** in the RDM SAS, we present the definition of NFRs satisfaction. NFRs are quality goals to be satisfied in a system. Measuring the satisfaction of NFRs such as NFR_t^1, NFR_t^2, and NFR_t^3 in Figure 1(b) is challenging as it may not be possible to conclude that an NFR is fully satisfied. Instead, they can be labelled as sufficiently satisfied [21]. Probabilistic approaches have been used to model the lack of crispness about the satisfiability nature of NFRs [8, 15, 24]. We leverage the mathematical framework provided by POMDPs to model the satisfaction of NFRs using probability distributions about the current state of the system (as presented in Section 2).

Based on Definition 3, the SLAs represent the *minimum satisfaction level* proposed for each NFR to be met during the system's execution. Any value below the acceptability threshold of an NFR is

Table 1. RDM SAS - SLAs

Non Functional Requirements (NFRs)	Poor satisfaction (under the threshold)	Suitable satisfaction (over the threshold)	Service Level Agreement (SLA)
Minimization of Cost (MC)	[0.00 , **0.70**)	[**0.70** , 1.00]	P (MC=True) >= **0.7**
Maximization of Reliability (MR)	[0.00 , **0.90**)	[**0.90** , 1.00]	P (MR=True) >= **0.9**
Maximization of Performance (MP)	[0.00 , **0.75**)	[**0.75** , 1.00]	P (MP=True) >= **0.75**

Table 2. CPTs for POMDP Transition Function

(a) CPT Minimization of Cost (MC)

NFR¹: Minimization of Cost (MC)					
Action A_t	MC_t	MR_t	MP_t	$P(MC_{t+1}=T)$	$P(MC_{t+1}=F)$
a^1=MST Topology	T	T	T	0.9	0.1
a^2=MST Topology	T	T	F	0.88	0.12
a^3=MST Topology	T	F	T	0.92	0.08
a^4=MST Topology	T	F	F	0.9	0.1
a^5=MST Topology	F	T	T	0.85	0.15
a^6=MST Topology	F	T	F	0.83	0.17
a^7=MST Topology	F	F	T	0.87	0.13
a^8=MST Topology	F	F	F	0.85	0.15
a^1=RT Topology	T	T	T	0.86	0.14
a^2=RT Topology	T	T	F	0.84	0.16
a^3=RT Topology	T	F	T	0.88	0.12
a^4=RT Topology	T	F	F	0.86	0.14
a^5=RT Topology	F	T	T	0.73	0.27
a^6=RT Topology	F	T	F	0.71	0.29
a^7=RT Topology	F	F	T	0.75	0.25
a^8=RT Topology	F	F	F	0.73	0.27

(b) CPT Max. of Reliability (MR)

NFR¹: Maximization of Reliability (MR)					
Action A_t	MC_t	MR_t	MP_t	$P(MR_{t+1}=T)$	$P(MR_{t+1}=F)$
a^1=MST Topology	T	T	T	0.91	0.09
a^2=MST Topology	T	T	F	0.93	0.07
a^3=MST Topology	T	F	T	0.89	0.11
a^4=MST Topology	T	F	F	0.91	0.09
a^5=MST Topology	F	T	T	0.93	0.07
a^6=MST Topology	F	T	F	0.95	0.05
a^7=MST Topology	F	F	T	0.91	0.09
a^8=MST Topology	F	F	F	0.93	0.07
a^1=RT Topology	T	T	T	0.95	0.05
a^2=RT Topology	T	T	F	0.97	0.03
a^3=RT Topology	T	F	T	0.93	0.07
a^4=RT Topology	T	F	F	0.95	0.05
a^5=RT Topology	F	T	T	0.97	0.03
a^6=RT Topology	F	T	F	0.99	0.01
a^7=RT Topology	F	F	T	0.95	0.05
a^8=RT Topology	F	F	F	0.97	0.03

(c) CPT Max. of Performance (MP)

NFR¹: Maximization of Performance (MP)					
Action A_t	MC_t	MR_t	MP_t	$P(MP_{t+1}=T)$	$P(MP_{t+1}=F)$
a^1=MST Topology	T	T	T	0.9	0.1
a^2=MST Topology	T	T	F	0.85	0.15
a^3=MST Topology	T	F	T	0.92	0.08
a^4=MST Topology	T	F	F	0.87	0.13
a^5=MST Topology	F	T	T	0.88	0.12
a^6=MST Topology	F	T	F	0.83	0.17
a^7=MST Topology	F	F	T	0.9	0.1
a^8=MST Topology	F	F	F	0.85	0.15
a^1=RT Topology	T	T	T	0.82	0.18
a^2=RT Topology	T	T	F	0.75	0.25
a^3=RT Topology	T	F	T	0.84	0.16
a^4=RT Topology	T	F	F	0.77	0.23
a^5=RT Topology	F	T	T	0.8	0.2
a^6=RT Topology	F	T	F	0.73	0.27
a^7=RT Topology	F	F	T	0.82	0.18
a^8=RT Topology	F	F	F	0.75	0.25

considered to be in a zone of *poor satisfaction*. In contrast, any value equal to or greater than the threshold is seen in a zone of *suitable satisfaction*. The identified SLAs for the NFRs of the RDM SAS are presented in Table 1.

As an example, for the case of *Maximization of Reliability* (MR) having an SLA P(MR = True) >= **0.9** would mean that the probability of satisfying MR should be at least 0.9. In other words, for the MR to be satisfied at a particular point in time, the RDM should have at least 90% of the concurrent active links. Different SLAs have been studied to confirm the suitability of the approach. Further details about the behaviour of the RDM SAS under different SLAs can be accessed in Appendix B.

4.4 NFRs and the POMDP Transition Function

Based on Bayes' theorem [71, 74], the transition function T (s, a, s') is factored as a product of conditional distributions [75]. Hence, using Definition 5, we derived a transition function as a function of the NFRs of the system. By applying this concept to the RDM SAS, the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR), and *Maximization of Performance* (MP) are shown in the following factored transition function:

$$T(s,a,s') = P(MC_{t+1}|MC_t, MR_t, MP_t, a_t) \\ P(MR_{t+1}|MC_t, MR_t, MP_t, a_t)P(MP_{t+1}|MC_t, MR_t, MP_t, a_t) \quad (5)$$

As observed in Equation (5), MC_{t+1}, MR_{t+1} and MP_{t+1} are influenced by both, the previous action $a_t \in A$ and the previous states of MC_t, MR_t and MP_t (i.e., they are interdependent). For example, Table 2(c) shows the probability of **0.88** to transit to the new state MP_{t+1} = **T**rue when the current state of the system is MC_t = **F**alse, MR_t = **T**rue, and MP_t = **T**rue. In POMDPs, these

Table 3. CPTs for POMDP observation function and Reward values R(s,a)

(a) CPTs for RBC, ANL and TTW

OBS¹: Ranges of Bandwidth Consumption (RBC)				
Action A_t	MC_{t+1}	$P(RBC_{t+1}<x)$	$P(RBC_{t+1}$ in [x,y>)	$P(RBC_{t+1}>=y)$
a^1=MST Topology (MST)	T	0.8	0.15	0.05
a^1=MST Topology (MST)	F	0.72	0.18	0.1
a^2=Redundant Topology (RT)	T	0.78	0.16	0.06
a^2=Redundant Topology (RT)	F	0.68	0.2	0.12

OBS²: Active Network Links (ANL)				
Action A_t	MR_{t+1}	$P(ANL_{t+1}<r)$	$P(ANL_{t+1}$ in [r,s>)	$P(ANL_{t+1}>=s)$
a^1=MST Topology (MST)	T	0.06	0.16	0.78
a^1=MST Topology (MST)	F	0.12	0.2	0.68
a^2=Redundant Topology (RT)	T	0.05	0.15	0.8
a^2=Redundant Topology (RT)	F	0.1	0.18	0.72

OBS³: Total Time for Writing (TTW)				
Action A_t	MP_{t+1}	$P(TTW_{t+1}<f)$	$P(TTW_{t+1}$ in [f,g>)	$P(TTW_{t+1}>=g)$
a^1=MST Topology (MST)	T	0.83	0.13	0.04
a^1=MST Topology (MST)	F	0.67	0.23	0.1
a^2=Redundant Topology (RT)	T	0.8	0.15	0.05
a^2=Redundant Topology (RT)	F	0.63	0.25	0.12

(b) Reward values R(s,a)

Reward values R(s,a)	Action (A_t)	State (S)		
		MC_{t+1}	MR_{t+1}	MP_{t+1}
r_1=0.0919	a^1=MST	T	T	T
r_2=0.0943	a^1=MST	T	T	F
r_3=0.0896	a^1=MST	T	F	T
r_4=0.0377	a^1=MST	T	F	F
r_5=0.1014	a^1=MST	F	T	T
r_6=0.0660	**a^1=MST**	F	T	F
r_7=0.0306	a^1=MST	F	F	T
r_8=0.0023	a^1=MST	F	F	F
r_9=0.1014	a^2=RT	T	T	T
r_{10}=0.0731	a^2=RT	T	T	F
r_{11}=0.0660	a^2=RT	T	F	T
r_{12}=0.0613	a^2=RT	T	F	F
r_{13}=0.0660	a^2=RT	F	T	T
r_{14}=0.0377	**a^2=RT**	F	T	F
r_{15}=0.0542	a^2=RT	F	F	T
r_{16}=0.0259	a^2=RT	F	F	F

states are not directly observable. Instead, Bayesian inference (as presented in Section 2) is used to compute a belief about the states. The conditional probability tables (CPTs) for the transition function of the RDM SAS are shown in Tables 2(a)–2(c). These conditional probabilities are defined by the domain experts [10]. For the case studies used in this article, we have taken [30, 37, 64, 78] as the sources of the domain knowledge needed to come up with the initial probabilities for the case studies RDM [37, 64] and IoT [30, 78].

4.5 Monitoring Variables (MON) and the POMDP Observation Function

Based on Definition 6, the factored observation function for the RDM case study is presented as follows:

$$O(s', a, z) = P(RBC_{t+1}|MC_{t+1}, a_t)P(ANL_{t+1}|MR_{t+1}, a_t)P(TTW_{t+1}|MP_{t+1}, a_t) \quad (6)$$

In Equation (6), the MON variables: *Ranges of Bandwidth Consumption* (RBC), *Active Network Links* (ANL) and *Total Time for Writing* (TTW) represent indirect observations of the state of the RDM SAS, i.e., the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP), respectively. The **conditional probability tables (CPTs)** modelling the observation function of the RDM SAS are shown in Table 3(a).

4.6 Utility Value for System Stakeholders when Executing Adaptation Actions

The utility value for system stakeholders when executing the adaptation of action $a \in A$ depends on the effects that the adaptation action has on the system's state $s \in S$. Thus, different effects may represent different utility values for the system stakeholders. In this article, the system's state represents whether its NFRs are satisfied (as presented in Definitions 1 and 3). In a POMDP, the utility values correspond with the reward function R(s,a) (as presented in Definition 4).

Reward Function for the RDM SAS. A reward function R assigns a numeric value to each 2-tuple: (state, action) of the system [69], indicating its desirability level in the decision-making process. The initial stakeholders' utility values of the RDM SAS are shown in the column "Reward values R(s,a)" of Table 3(b). As an example of the initial utility values for system stakeholders, in Table 3(b), we observe that the effect of execution of an adaptation action (a = MST or a = RT) on the system's state to the new state: MC = **F**alse, MR = **T**rue and MP = **F**alse (as described in rows r_6 and r_{14}),

the stakeholders favour the topology MST (see row r_6 = 0.0660) over the topology RT (see row r_{14} = 0.0377). This suggests that, under this specific state (i.e., when MP and MC are not being satisfied), the *Minimum Spanning Tree topology* (MST) would be preferred over a *Redundant topology* (RT) which would offer more reliability. All other possible reward values R(s, a) in Table 3(b) favour the topologies MST or RT, based on the initial utility value specifications determined at design-time. It is important to note that the effects on the system's state of executing adaptation actions may change over time. Hence, the perceived utility values for system stakeholders (i.e., their preferences) can also change.

To illustrate the advantages provided by RE-STORM, Section 6.1.3 shows how different dynamic contexts have been simulated to produce changes in the effects of the execution of adaptation actions to, therefore, trigger the need for reassessment of stakeholders' utility values. Next, details on the decision-making process performed by RE-STORM are presented.

5 RE-STORM: DECISION MAKING DRIVEN BY PARTIALLY OBSERVABLE NFRS

This section presents details of the proposed approach for self-adaptation based on partially observable NFRs. The runtime behaviour of RE-STORM is based on a POMDP model within a feedback control loop (see Figure 2). The different activities of the MAPE-K loop [70], and the details of the decision-making in the RDM SAS are presented as follows:

5.1 RE-STORM: MAPE-K Loop Activities

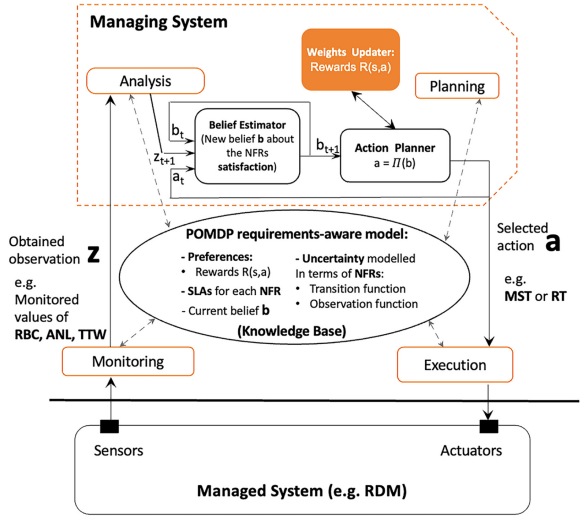
(1) **Monitoring**. In this activity, the observable data, observations z ∈ Z from the managed system, are collected by sensors (see Figure 2). Specifically, in the RDM SAS, the MON variables *Ranges of Bandwidth Consumption* (RBC), *Active Network Links* (ANL) and *Total Time for Writing* (TTW) are monitored. The observed values for each MON variable, i.e., values from the ranges (RBC <x, RBC in [x,y) and RBC>=y), (ANL<r, ANL in [r,s) and ANL >=s) and (TTW<f, TTW in [f,g) and TTW >=g), constitute the evidence to compute in the next activity the current belief about the state of the system $s \in S$.

(2) **Analysis.** Any required data transformation to enable data to be used at the Planning stage should be performed at this step. Therefore, the component labeled *Belief Estimator* in Figure 2 is responsible for updating the belief b_t about the system's state. This update is performed by using Equation (1). The result is a belief b_{t+1} about the new state of the system after the execution of the previous adaptation action a_t. This belief will be the input for the planning activity, and it is also recorded in the Knowledge Base as part of the POMDP requirements-aware model (see details in Figure 2).

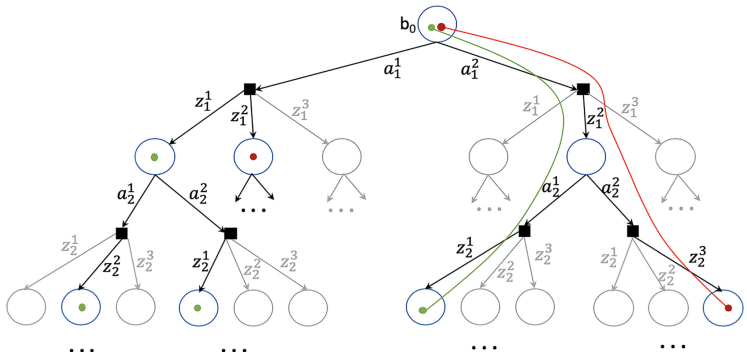
(3) **Planning.** The component labeled *Action Planner* in Figure 2 is the policy $a = \pi(b)$ responsible for generating actions, as a function of the current belief b about the satisfaction of the NFRs in the system. We use two different solvers for POMDP planning [50, 54] to choose the best adaptation action. The first solver uses online POMDP planning, a technique that interleaves planning with plan execution: at each time slice, the system searches for an optimal action $a \in A$ at the current belief b. It then executes the chosen action immediately [54].On the other hand, the second solver makes use of point-based value iteration approach for solving POMDPs [16, 56].

(4) **Execution.** Once an action has been selected, it is executed over the managed system (see Figure 2). As a result, the system reaches a new state s' with probability T (s, a, s′) = P(s′ |s, a). This state is not directly observable, yet an observation $z \in Z$ with probability O(s', a, z) = P(z|s′, a) is received. Then, the MAPE-K loop starts again.

(5) **Knowledge.** The Knowledge Base of an autonomic system holds models/abstractions that support the Monitoring, Analysis, Planning and Execution activities [2]. In the case of



(a) RE-STORM runtime architecture for decision making



(b) DESPOT Belief tree with 2 sampled trajectories represented by red and green lines.

Fig. 2. RE-STORM architecture.

RE-STORM, a POMDP model is kept in the Knowledge Base (see Figure 2). The POMDP models contains the current believes about the satisfaction of the NFRs and the stakeholders' utility values about the effects on the system of executing adaptation actions, i.e., the reward values R(s, a). It also contains the SLAs of the system and the current representation of the system's uncertainty, i.e., the transition and observation functions, both modelled in terms of the system's NFRs.

The next section presents details on the planning activity to select an adaptation action.

5.2 Online Planning Activity using POMDPs

We use the **Determinized Sparse Partially Observable Tree (DESPOT)** algorithm [54] as the planner implementation. The steps of the planning activity are shown next:

(1) *Planning Step 1: Detect NFRs Below Their Thresholds of Satisfaction.* If the current belief of an NFR has its satisfaction level labelled below its SLA, then the *Weights Updater module*

[26] (see Figure 2) is performed to update the current stakeholders' preferences according to changes in the utility values specified as reward values R(s,a).

*The **Weights Updater module** is used to reassess the stakeholders' utility values to consider assigning more importance (i.e., preference) to adaptation actions with a more positive effect on the satisfaction of NFRs below their thresholds of satisfaction.*

The latter explains how RE-STORM supports runtime reassessment and update of preferences due to changes of utility values by using the ARRoW implementation presented in [26]. Due to the modularity of the RE-STORM architecture, other techniques can be used to reassess and update preferences regarding NFRs such as [40], [41], [55], and [60].

(2) *Planning Step 2: Build a DESPOT Tree to Project Future Evolutions of the Satisfaction of NFRs.* One desirable capability of autonomic self-adaptive systems is anticipation, which is defined as being able to anticipate, to some extent, needs and behaviours to be able to manage itself in a proactive way [39]. This capability is known as *Proactive self-adaptation*, and implies predictions of how the environment will evolve in the near future [38]. RE-STORM supports decisions under the uncertainty implied by those predictions. Its implementation is based on a belief tree provided by the DESPOT algorithm [54] in order to select an adaptation action. Next, we present relevant details related to the capabilities for proactive self-adaptation used by the approach.

Future Evolutions of the System's State. The *Action Planner module* of RE-STORM (see Figure 2), considers future evolutions of the belief b about the satisfaction of the NFRs to decide the next adaptation action $a \in A$, i.e., to reason about long-term effects of immediate actions [54]. The future evolutions of the belief about the system's state are represented by the DESPOT tree shown in Figure 2(b). The algorithm builds per each time slice a sparse approximation of a standard belief tree: a DESPOT tree by using a simulation model [18]. The root node of the tree is the belief b_0 which represents the belief about the current satisfaction of the NFRs of the running system. Each edge in the tree represents an action-observation pair, i.e., (i) an adaptation action MST or RT (e.g., a_1^1 or a_1^2) and (ii) an observation $z \in Z$ (e.g., z_1^1, $z_1^2 \ldots$). Except for the root node b_0, each circular node in the tree represents a projected belief about the future state of the system, i.e., the predicted satisfaction of its NFRs. The DESPOT tree also represents the neighbourhood of the current belief b_0 about the satisfaction of the NFRs of the system.

(3) *Planning Step 3: Select the Optimal Action $a \epsilon A$.* Bellman's principle of optimality [48], used by the DESPOT POMDP solver, is shown in Equation (7). It is applied over the DESPOT tree to choose the best adaptation action.

$$V^*(b_0) = \max_{a \in A} \left\{ \sum_{s \in S} b(s)R(s,a) + \gamma \sum_{z \in Z} p(z|b,a)V^*(r(b,a,z)) \right\} \tag{7}$$

The DESPOT algorithm searches the tree with root at the current belief b_0. Specifically, the action planner module of the DESPOT algorithm (Figure 2) uses look-ahead search [49] to approximate the optimal discounted reward value $V^*(b_0)$ [52, 54, 73]. The search is guided by a lower bound $l(b_0)$ and an upper bound $\mu(b_0)$ on the approximated optimal discounted reward value $V^*(b_0)$. The explorations continue until the gap between the bounds $\mu(b_0)$ and $l(b_0)$ reaches a target level ϵ_0 or the allocated planning time T_{max}[1] runs out. Equation (7) recursively computes over the tree the maximum value of action branches and the average value of observation branches [18, 46, 54, 73]. The result is an approximately optimal policy

[1]Default values provided by the DESPOT algorithm: DESPOT tree height D_h = 90. Max planning time T_{max} = 1 second for each time slice.

based on the belief b_0 about the current system's state, i.e., the current satisfaction of its NFRs. The system then executes the optimal action of the policy $\pi(b_0)$.

The selection of the optimal action $a \in A$ made in Equation (7) is based on the current and projected beliefs about the system's state, i.e., about the satisfaction of its NFRs.

Details on how the DESPOT algorithm implements the Bellman equation can be found in Appendix D. Next, the evaluation of the proposal is described.

6 EVALUATION

This section presents the set of experiments used to evaluate RE-STORM applied to two case studies: RDM [59] and IoT [30]. Different dynamic contexts have been carefully designed to be used during the experiments. As a proof of generalization of the RE-STORM approach, we have executed experiments using two different POMDP solvers: DESPOT [54] and Perseus [50]. Both solvers use simulations for experiments. The difference is that the DESPOT has a simulation capability that is used to simulate RDM whereas, in case of Perseus, we connect it to an external simulator (RDM-Sim [59] in this article), or it can even be connected to a real system. The solvers are described in Appendixes D and E. For the purpose of reproducibility, we have provided the results of the experiments in [65].

We argue that RE-STORM allows quantifying the uncertainty of adaptation actions on the levels of satisfaction of the NFRs of the running system using probabilities (see Definition 2). For instance, after the execution of an adaptation action, RE-STORM can quantify the uncertainty about the satisfaction level of an NFR, through probabilities; e.g., the probability of Minimization of Energy Consumption may have increased due to the action performed, and the probability conveys a quantification of the uncertainty needed to support decision-making. A primary goal of the evaluation is to assess the RE-STORM's capability of quantifying this kind of uncertainty. Therefore, we state the following research question:

RQ1: *Can the uncertainty associated with the stochastic effects of executing an adaptation action be quantified?*

Once the answer to **RQ1** has been proven affirmative, a second goal of the evaluation is to assess that the capabilities of quantification of uncertainty by RE-STORM, described above, can be used to improve the decision-making and behaviour of the running system. To do so, we have explored opportunities to reassess and enhance the tradeoff among NFRs based on new knowledge and evidence acquired during runtime supported by RE-STORM's capabilities of quantification.

RE-STORM can highlight situations such as where the utility/reward associated with a given NFR has changed, which can influence changes in the preferences shown by stakeholders (e.g., in the newly identified environmental context, performance can turn out to be less crucial than reliability). We argue that based on knowledge and evidence found during execution, the tradeoff among NFRs can be improved by updating the utility value for system stakeholders (i.e., the preference of the stakeholders) with respect to executing adaptation actions. Therefore, we propose our second research question:

RQ2: *Under new and unexpected contexts observed, can the tradeoff among the NFRs be improved by updating the stakeholders' preferences based on the new knowledge found at runtime ?*

Next, we present the experiments performed using the DESPOT solver.

6.1 Evaluation using DESPOT

This subsection describes the experiments performed using DESPOT solver. Next, we provide the details of the infrastructure used and the initial setup for experiments.

6.1.1 Infrastructure Used During the Evaluation. The behaviour of the RDM SAS has been implemented with the simulation model [18] provided by the DESPOT toolkit [18, 54]. The RDM network and its behaviour are based on the case study presented in [64] and the more general specifications and expert-based knowledge shown in [31] and [37]. The RDM case study of this article comprises 25 RDM servers with 300 physical network links. For the RDM configuration, the minimum number of **Active Network Links (ANL)** expected is equal to *24* [64]. The RDM application has been simulated over 1000+ time slices (i.e., simulations of at least 1,000, 2,000, and 3,000 time slices have been performed). During each simulation, periods of dynamic perturbances have been randomly inserted.

6.1.2 Initial Setup of Experiments with the RDM SAS. Initial configurations used in the evaluation of the RDM SAS are explained next.

Uncertainty Management in RE-STORM. Two main POMDP elements deal with the uncertainty in the RDM SAS due to unpredictable environments: namely, the transition function T (s, a, s′) and the observation function O (s′, a, z). Both have been specified in Sections 4.4 and 4.5, respectively. In this work, we specifically quantify and evaluate the uncertainty related to the transition function T(s, a, s′) = P (s′ | s, a) (see Definition 5). For the case of the RDM SAS, network parameters have been initialized according to the known probability that certain network links will fail at any given point during the system's execution as described in [64] and [37].

Configuration of Dynamic Changes in the Environment. For the RDM SAS, the topologies MST and RT represent the adaptation actions. The effects on the system's state of executing these topologies may change at runtime. Simulations, considering changes on the effects of these topologies, have been implemented by randomly modifying the transition function T (s, a, s′) = P(s′ | s, a) during the simulation as presented in Section 4.4 (see Tables 2(a)–2(c)).

The probability distributions of the transition function T(s, a, s′) = P(s′ |s, a) have been randomly changed at runtime by introducing disturbances to generate dynamic changes in the environment.

The new effects are conceived to make the SAS show periods of deteriorated satisfaction of the NFRs and evaluate RE-STORM. Accordingly, the probability P(s′ = True | s, a) is randomly decreased. Different dynamic contexts envisioned to represent this behaviour are presented in Section 6.1.3. The duration of each period of deteriorated satisfaction has been randomly selected from a range between 5 and 15 time slices based on the data provided by Minwen et al. [37]. Further details about the configuration of RE-STORM and the dynamic contexts for its evaluation can be found in Appendix C.

6.1.3 Experiments. This section presents how RE-STORM performs under different dynamic contexts. First, we showcase the behaviour of the RDM under its *stable conditions*.

RDM SAS under Stable Conditions. Figure 3 shows the behaviour of the RDM SAS using the specifications presented in Sections 4.3 to 4.6. This behaviour is taken as that shown by the RDM SAS in *stable conditions*. The *stable conditions* of the system represent foreseen scenarios envisioned by

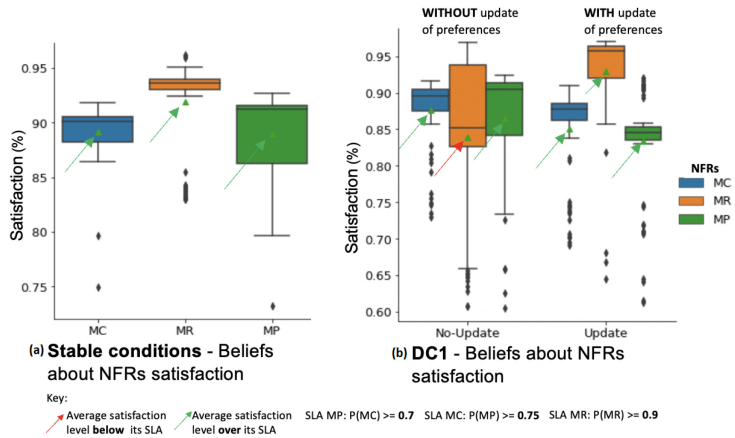
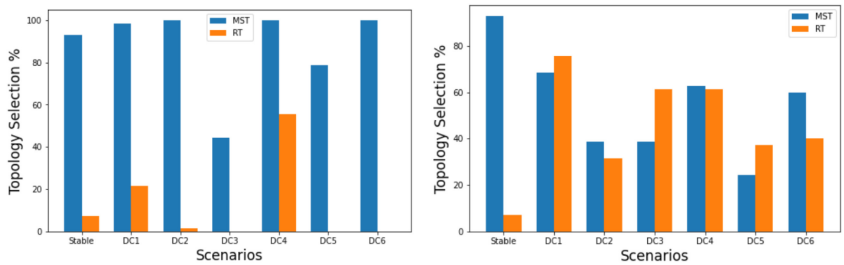


Fig. 3. RDM SAS - behaviour under different environmental conditions.



(a) Without Update of preferences (b) With Update of preferences

Fig. 4. RDM SAS - topology selection.

the experts, based on their knowledge of the context of the system. Under *stable conditions*, the following behaviour has been identified:

- The preferred configuration is to use a *Minimum Spanning Tree Topology* (MST) (See Figure 4).
- The average belief about the satisfaction of the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR), and *Maximization of Performance* (MP) complies with the SLAs of the system (see Figure 3).

The stochastic effects on the system's state of executing the adaptation actions MST and RT are quantified as beliefs about the satisfaction of the system's NFRs. Under stable conditions, these beliefs comply with the established SLAs.

RDM SAS under Dynamic Contexts. Six different dynamic contexts have been defined. Each dynamic context represents variations of the *stable conditions* of the RDM SAS to trigger the need to reassess stakeholders' utility/preference values. Next, the nature of the changes is presented.

- **Dynamic Context DC_1.** Changes in the environment during the execution of the MST topology are introduced to reduce the *reliability* of the system. These changes are implemented by altering the following conditional probability in the transition function T (s, a, s'): $P(\mathbf{MR}_{t+1} = \text{True}|\text{NFR}_t\ , \mathbf{MST}_t)$.

DC_1 Description: A period of consecutive and unexpected data packet loss during the execution of the MST Topology is generating a reduction on the reliability of the system. An MST topology is designed to connect all remote sites in an RDM SAS by the identification of a minimum spanning tree on the network of possible links among each remote site. Data packet loss may represent link failures in an RDM system, which may be caused, for example, by problems with the equipment (e.g., failures in a switch or router or power failures [37]).

- **Dynamic context DC_2.** Changes in the environment during the execution of the RT topology are introduced to increment the *cost* and reduce the system's *performance*. The changes are implemented by altering the following conditional probability in the transition function T (s, a, s′): P($\mathbf{MC}_{t+1}$ = True, $\mathbf{MP}_{t+1}$ = True|NFR_t , $\mathbf{RT}_t$).
  DC_2 Description: Unexpected data packet loss during the execution of the RT Topology generates an unusual rate of data forwarding, increasing the bandwidth consumption (i.e., cost) and reducing the system's performance. In the RDM SAS, the cost for inter-site link communication is a function of the data sent over them. Therefore, a *Redundant Topology* (RT), which involves a more significant number of inter-site network links than a *Minimum Spanning Tree Topology* (MST), is more expensive. Costs increase as the number of network links increases and a reduction in the system's performance[2] could also be expected.
- **Dynamic context DC_3.** Simultaneous occurrence of the dynamic context DC_1 and DC_2.
- **Dynamic context DC_4.** Changes in the environment during the execution of the MST topology are introduced to increment the *cost* and to reduce the *reliability* and the *performance* of the system. The changes are implemented by altering the following conditional probability in the transition function T(s, a, s′): P($\mathbf{NFRs}_{t+1}$ = True|NFR_t , $\mathbf{MST}_t$).
  DC_4 Description: It involves the behaviour presented in the dynamic context DC_1. Additionally, and different from the dynamic context DC_1, the execution of the MST topology also produces an increment in bandwidth consumption (MC) and the reduction of the system's performance (MP).
- **Dynamic context DC_5.** Changes in the environment during the execution of the RT topology are introduced to increment the *cost* and to reduce the *reliability* and the *performance* of the system. The changes are implemented by altering the following conditional probability in the transition function T (s, a, s′): P($\mathbf{NFRs}_{t+1}$ = True|NFR_t , $\mathbf{RT}_t$).
  DC_5 Description: Involves the behaviour presented in the dynamic context DC_2. Additionally, the RT topology also reduces the reliability of the system (MR).
- **Dynamic context DC_6.** Simultaneous occurrence of the dynamic context DC_4 and DC_5.
  DC_6 Description: This context represents an unusual scenario explicitly used to evaluate our approach under extremely detrimental conditions. A case like this would be usually related to a significant site failure [31, 37], where both repeated and multiple concurrent failures are expected [37] as in the previous two dynamic contexts but all at the same time. For example, a full-scale site failure may be caused by a power outage affecting all the buildings on different campuses, an earthquake or a flood affecting structures within several metropolitan areas. Under this context, the worst-case data loss [31] may occur in different sites (RDM nodes), i.e., a site can be destroyed or inoperative before the full backup of information is shipped offsite. Site failure disasters are usually modelled with a failure rate of once per year [31]. The main goal of this dynamic context is to study the behaviour of the RDM SAS using RE-STORM in situations where it may be challenging to meet the SLAs regardless of the adaptation action selected.

[2]Performance in these systems is measured as the total time to perform the write of data, which is the sum of the response times of the writes of each copy of data on each remote site [37].

6.1.4 Results of Experiments. This section presents results on the dynamic context DC_1 and an aggregated view of the results for the six (6) dynamic contexts under evaluation. Nevertheless, specific results on the dynamic contexts DC_2 to DC_6 can be consulted in Appendix A.

Dynamic context DC_1. To better explain this context, first, we showcase the behaviour of the RDM SAS before using RE-STORM to update stakeholders' preferences; then, the behaviour of the system after their update is presented.

Behaviour before updating stakeholders' preferences due to changes of utility values. The following findings are presented:

- Without the update of preferences, the preferred configuration is to use a *Minimum Spanning Tree Topology* (MST) (see Figure 4(a)).
- Without the update of preferences, the execution of the MST topology reduces the system's reliability compared to its *stable conditions*. While the average belief about the satisfaction of the NFRs *Minimization of Cost* (MC) and *Maximization of Performance* (MP) agree with their SLAs, the belief about the satisfaction of *Maximization of Reliability* (MR) is below its SLA (see Figure 3(b): Beliefs about NFRs satisfaction *without* update of preferences).

The current stakeholders' preferences (see Section 4.6), suitable for the *stable conditions* of the system identified in Section 6.1.3, are not suitable for the dynamic context DC_1. Instead, they favour the MST topology, which reduces the system's reliability. Therefore, reassessment and update of stakeholders' preferences are needed under *the new and unexpected context* detected at runtime. Let us explore the results of this:

*Behaviour after Updating Stakeholders' Preferences due to Changes of Utility Values.*The behaviour presented below has benefited from the update of stakeholders' preferences (Planning Step 1 of RE-STORM in Section 5.2). Next, the main findings are highlighted:

- By updating the stakeholders' preferences, the preferred configuration is *Redundant Topology* (RT) (see Figure 4(b)).
- The average belief about the satisfaction of *Maximization of Reliability* (MR) improves as a result of the better-informed decision-making provided by RESTORM. It is also observed that the trade-off among NFRs slightly reduces the belief about the satisfaction of *Minimization of Cost* (MC) and *Maximization of Performance* (MP) in comparison to the *stable conditions* of the system (See Figure 3(b): Beliefs about NFRs satisfaction *with* update of preferences). Nevertheless, MC and MP are still over their satisfaction thresholds.

The behaviour above corresponds to Planning Step 1 of the online planning activity of RE-STORM (see Section 5.2). In this step, more importance is given to an adaptation action (i.e., RT), which has a more positive effect on the system's reliability (MR), which was below its threshold of satisfaction.

In the dynamic context DC_1, the update of stakeholders' preferences, i.e., reward values R(s, a) in a POMDP, contributes to improving the satisfaction of the reliability in the RDM SAS. As a result, all the NFRs meet their SLAs.

Aggregated view of results. Figure 5 synthesizes the NFRs satisfaction under the dynamic contexts DC_1 to DC_6.

It can be observed when the decision-making process is applied under *new detected contexts* that were not foreseen in advance and RE-STORM was not used to update the stakeholders' preferences (see Figure 5: Beliefs about NFRs satisfaction *without* update of preferences), the beliefs about the

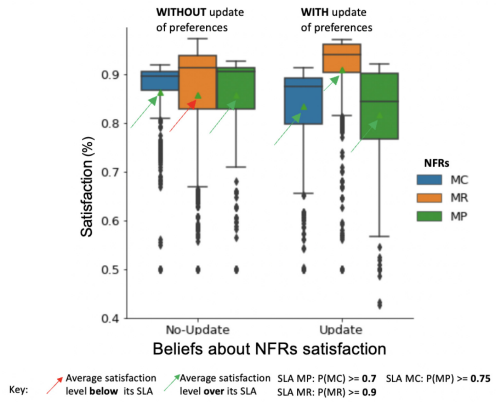


Fig. 5. RDM SAS - consolidated view under dynamic context DC_1 to DC_6.

satisfaction of cost (MC) and performance (MP) met their SLAs. Still, the belief about the reliability (MR) satisfaction was below its required SLA. Under this scenario, the preferred adaptation action is the MST topology (see Figure 4: Selected topology *without* update of preferences), even though the RT topology offers higher levels of reliability based on the tradeoffs specified in Section 4.1. This behaviour is due to initial stakeholders' preferences introduced in Section 4.6, which were only suitable for the RDM SAS under *stable conditions* (see Section **??**). On the other hand, when the stakeholders' preferences are updated by RE-STORM according to the newly detected contexts, the RT topology is more frequently selected. As a result, the belief about the satisfaction of MR is improved and taken to a value that meets its SLA. As a tradeoff, a reduction in the beliefs about the satisfaction of cost (MC) and performance (MP) is also observed but still continues to meet their SLAs (see Figure 5: Beliefs about NFRs satisfaction *with* update of preferences).

The reassessment and update of preferences allowed the RDM SAS to select more suitable adaptation actions to improve the general performance and tradeoffs among its NFRs.

Hence, RE-STORM has proved to provide the infrastructure to support the reassessment and update of the stakeholders' preferences, i.e., reward values R(s, a) in a POMDP when dynamic contexts not previously foreseen are found during the system's execution. Therefore, RQ2 is answered in the affirmative.

Next, we present the experiments performed using Perseus solver.

6.2 Evaluation using Perseus

In this section, we present experiments using the Perseus POMDP solver. Details regarding the implementation of RE-STORM with the Perseus algorithm can be consulted in Appendix E. This time, we evaluate the application of RE-STORM when it is connected to the simulating environments RDMSim [59] and DELTA-IoT [30]. This allows us to test both the validity of the approach when it interacts with real environments and to present RE-STORM's applicability in different domains. Due to the limitation of space, the experiments shown do not cover the update of preferences, as the experiments related to RQ2 have already been approached in the previous section. However, these results have been reported in [60].

6.2.1 RDMSim Experiments. In this section, we describe the experimental setup and results of RE-STORM (using Perseus) for the RDMSim network.

Experimental Setup: The tool of RDMSim presents a stimulating environment for the remote data mirroring network [31, 37]. The simulator is designed according to the infrastructure presented in [64] and is equivalent to the case study presented in Section 4. Similar to the experiments presented in previous sections, we have considered the three NFRs of *Maximization of Reliability* (MR), *Maximization of Performance* (MP) and *Minimization of Cost* (MC). In the case of RDMSim, Operational Cost is measured in terms of Bandwidth Consumption, Performance is measured in terms of Time to Write Data and reliability is measured using number of active network links. For the purpose of satisfaction of NFRs, the network is required to take adaptation actions by switching between the topologies of MST and RT. The simulation tool of RDMSim provides us with the implementation of the different dynamic contexts for the RDM network similar to the ones presented in Section 4.3. According to the specifications of RDMSim, the following threshold requirements for the satisfaction of NFRs are considered:

R1: Bandwidth consumption should be less than or equal to 40 per cent of total bandwidth consumption to satisfy MC.

R2: Number of active links should be greater than or equal to 35 per cent of total links to satisfy MR.

R3: Time to write data should be less than or equal to 45 per cent of total writing time to satisfy MP.

Experiment Results: We have executed experiments using all the dynamic contexts DC1 to DC6 for the RDMSim network for 1,000 simulation time slices. These experiments correspond to those presented in Section 6.1.4 and Appendix A by using RE-STORM without updating stakeholders' preferences. Experimental results show that MC and MP meet the threshold requirements having the average satisfaction level below the threshold for satisfaction as presented in Figure 6. For MC, the satisfaction threshold has a total bandwidth consumption below 3,700 Gbps, whereas the satisfaction threshold for MP has a total writing time below 2,700 milliseconds. In the case of MR, the average satisfaction level lies above the threshold, i.e., 105 active links, in several dynamic contexts. The exception lies in the results for DC1, DC3 and DC6 where the average satisfaction level is below the threshold. The reason behind it is the uncertain contextual situations like link failures that occur at runtime under these dynamic contexts. Figure 7 shows the behaviour in terms of topology selection for RDMSim network as a result of the adaptations offered by RE-STORM. Let's consider the case of DC4. The uncertain situation of link failures during the execution of MST topology affects the satisfaction of MR and the NFRs MC and MP are also affected due to synchronous mirroring.[3] In such a situation, RE-STORM increases the usage of RT topology to support the satisfaction of MR along with maintaining satisfaction levels of MC and MP.

In summary, the results achieved using the Perseus implementation of RE-STORM for the case of the RDMSim network show comparable results to the DESPOT implementation of the RE-STORM approach without updating preferences, which leads to similar conclusions. Moreover, the topology selection behaviour in all the cases using the Perseus implementation for RE-STORM is also similar to that of DESPOT without an update of preferences as presented in Section 4 and Appendix A.

6.2.2 DELTA-IoT Experiments. In this section, we describe the experimental setup for the case of DELTA-IoT network and experimental evaluations of RE-STORM (using Perseus) for the case.

Experimental Setup: The simulating environment of DELTA-IoT presents a multi-hop IoT network for a smart campus. The network consists of 15 motes based on LoRa (Long-Range)

[3]The implementation of RDMSim considers the synchronous mirroring protocol during communication.

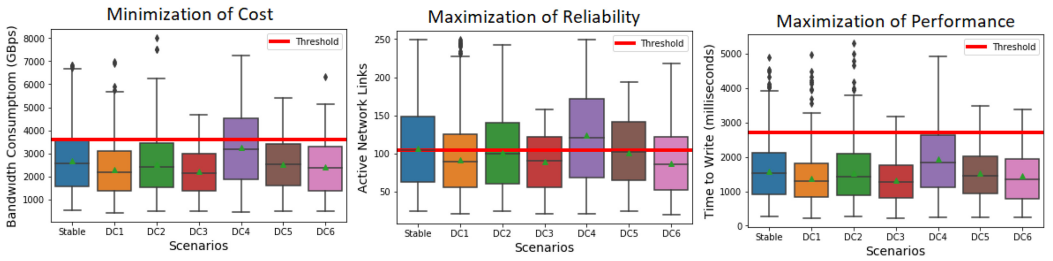


Fig. 6. RE-STORM using perseus: Satisfaction of NFRs under different dynamic contexts of RDMSim.

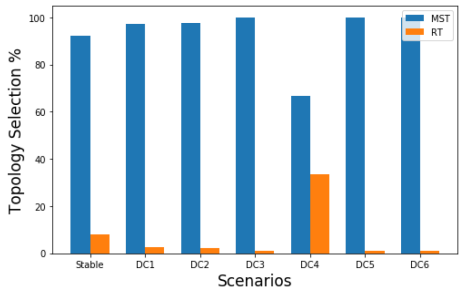


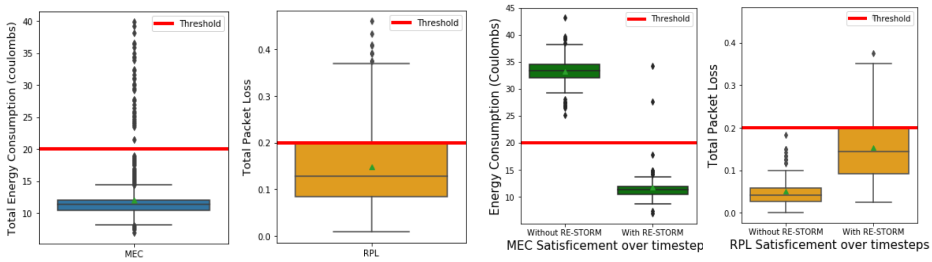
Fig. 7. RE-STORM using perseus: Satisfaction of NFRs under different dynamic contexts of RDMSim.

radio communication. The motes comprise RFID sensors, passive infrared sensors and temperature sensors that are installed across different buildings of the KU Leuven campus for the purpose of providing access to the laboratories, monitoring the occupancy status and sensing the temperature. The functional goal for the communication of motes is to relay information to the central gateway at the central monitoring facility. The IoT network is required to survive for a longer period of time with minimal battery usage and achieve communication reliability. Hence, for this purpose, the NFRs considered in the case of the DELTA-IoT network are the Minimization of Energy Consumption (MEC) and Reduction of Packet Loss (RPL) under uncertain environmental conditions of communication interference on the links and dynamic traffic load. In order to achieve the required satisfaction levels for the NFRs, tuning the network link settings such as the transmission power, communication range and distribution factor for the links using different adaptation strategies is needed. The threshold requirements for the satisfaction of NFRs according to the specifications of the DELTA-IoT network are as follows:

R1: Total energy consumption for the network should be less than or equal to 20 coulombs in order to satisfy MEC.

R2: Total packet loss for the network should be less than or equal to 20 per cent in order to satisfy RPL.

For the purpose of experimental evaluations, we have performed adaptations using RE-STORM for 100 simulation time slices of the DELTA-IoT network. In the case of the DELTA-IoT network, one simulation time slice is equal to 15 minutes of network activity. During each simulation time slice, local adaptation decisions for each mote are made using the Perseus implementation for RE-STORM. Based on the monitored link interference measured in the form of Signal to Noise Ratio for each mote, the decision of increasing transmission power (ITP) or decreasing transmission power (DTP) on the links is taken to support the satisfaction of MEC and RPL. Next, we present the experimental results for the case of the DELTA-IoT network.



(a) RE-STORM using Perseus: Satisfaction of NFRs for DELTA-IoT network

(b) Satisfaction of NFRs for DELTA-IoT network without using RE-STORM

Fig. 8. DELTA-IoT network satisfaction of NFRs.

Experiment Results: First, we describe the results of the adaptations offered by RE-STORM at each mote level during each simulation time slice. As the network comprises of 14 motes and 1 gateway, so adaptive decisions for the 14 motes are taken individually at the end of each simulation time slice. Figure 8(a) shows the satisfaction level of NFRs MEC and RPL maintained by RE-STORM under the dynamic environmental changes caused by the link interference on the network links associated with the motes. The results show that the average satisfaction level for both the NFRs is below the required threshold. The average satisfaction for MEC is 11.9758 coulombs, which is less than 20 coulombs, whereas, for RPL, the average satisfaction is 0.147, which is below the required threshold of 0.20 as shown in Figure 8(a). Hence, the required satisfaction threshold requirements are met.

Furthermore, we have also compared the satisfaction level of NFRs at the end of each simulation time slice with and without the application of RE-STORM as shown in Figure 8(b). Considering the case of MEC, the average satisfaction level of MEC is above the required threshold when the DELTA-IoT network operates without the application of RE-STORM. The adaptations offered by RE-STORM maintain the required threshold constraint of 20 coulombs by achieving the average satisfaction below the threshold as shown in Figure 8(b). Furthermore, for RPL, the application of RE-STORM shows an increase in packet loss by offering a tradeoff for the satisfaction of MEC. However, there is an increase in the packet loss, RE-STORM maintains the required satisfaction level by keeping the average satisfaction level of RPL below the satisfaction threshold of 0.20. Hence, RE-STORM proves to show better results in terms of maintaining the requirements of the system when compared to the DELTA-IoT system working without the adaptation offered by RE-STORM.

7 DISCUSSION

This section discusses the evaluation of the results presented and the threats to the validity of this study.

7.1 Evaluation of Results and Implications

To validate our findings, we have run several statistical tests to the data of the experiments. These tests supported the evaluation of the null Hypotheses described next.

Let's recall from Definition 2 that RE-STORM allows for the quantification of the uncertainty of adaptation actions on the levels of satisfaction of the NFRs by using probabilities (i.e., these probabilities represent the beliefs about the levels of satisfaction of the NFRs). Further, Definition 2 also states that these probabilities are conditional to the stochastic effects of the adaptation actions executed over the system. Therefore, we define the following null hypothesis:

Table 4. DC_1 - Logistic Regression Models

Non-functional requirement (NFR)	Overall Percentage (%) Predicted for actual state	B	S.E.	Wald	df	Sig.
MC	**85.100**	5.872	1.365	18.498	1.000	**0.000**
MP	**84.300**	8.559	1.168	51.368	1.000	**0.000**
MR	**92.800**	8.370	1.418	36.422	1.000	**0.000**

$H_{0,1}$: *"The probabilities (i.e., beliefs) about the actual state of the system's NFR don't represent the real observations collected."*

Based on this Hypothesis, with the experiments, we assessed if the belief probabilities represent the actual observations. If the belief probabilities don't account for the actual observations the null hypothesis would be rejected.

For the second Hypothesis, let's recall from Section 6, RE-STORM can highlight situations such as those when the utility/reward associated with a given NFR has changed. These new utilities can affect the preferences shown by stakeholders (e.g., if some new environmental context is identified, performance can be found to be less crucial than reliability, and as such, the end user needs to be informed). Based on the new knowledge found by RE-STORM, the tradeoff among NFRs may be improved by updating the preferences of the stakeholders with respect to adaptation actions. Therefore, we define our second null hypothesis:

$H_{0,2}$: *"There is no difference between updating and not updating stakeholders' preferences to improve the trade-off among the system's NFRs."*

With the experiments performed, we assessed if the update of preferences improves the tradeoff among the system's NFRs. The eventual rejection of the null hypotheses $\mathbf{H}_{0,1}$ and $\mathbf{H}_{0,2}$ will support the answers provided to the research questions **RQ1** and **RQ2** respectively.

Hypothesis $H_{0,1}$ evaluation. In POMDPs, the system's state s ∈ S is not directly observable. Instead, the decision-making is driven by the belief about these states. Let us recall from Definition 2 that RE-STORM allows quantifying the uncertainty of adaptation actions on the levels of satisfaction of the NFRs of the running system using belief probabilities. Given that we have used the DESPOT toolkit [54] to simulate the dynamic contexts in our experiments, we had access to the satisfaction state of NFRs based on the SLAs, i.e., $s \in S$. Therefore, a logistic regression analysis could determine how well the beliefs represent the satisfaction state of NFRs, i.e., the state of the system. ***Statistical significance.*** The results related to the dynamic context DC_1 are presented in Table 4. We found in all cases that our belief representation about the satisfaction of the NFRs in the RDM SAS was statistically significant. Specifically, for *Minimization of Cost* (MC), the logistic regression analysis shows that our belief representation had an 85.1% of success rate to predict the actual system's state (p-value = **0.000**, Wald statistic = **18.498**). In the case of *Maximization of Reliability* (MR), the success rate for predicting the real system's state was of 92.8% (p-value = **0.000**, Wald statistic = **36.422**); finally, for *Maximization of Performance* (MP), we obtained a success rate of 84.3% (p-value = **0.000**, Wald statistic = **51.368**).

Equivalent results have also been obtained for our probabilistic representations in the dynamic context DC_2 to DC_6 (see Appendix A). The use of logistic regression analysis allowed us to demonstrate how the beliefs about the system's state **s'** in the transition function T (s, a, **s'**) = P(**s'** |s, a) are statistically significant. The new state **s'** represents the stochastic effects on the system of

executing the adaptation action a ∈ A (see Definition 2). Therefore, we have been able to quantify the uncertainty related to the state **s'** as its belief by using RE-STORM.

Furthermore, we have also performed logistic regression to evaluate the results for implementation of RE-STORM using Perseus solver when used with the simulating environments of RDMSim and DELTA-IoT. In these cases, we have both the belief probabilities and observations coming from the simulators. Hence, using logistic regression, we further investigate when both the belief probability and observations are considered and to what extent they represent the actual state of the systems' NFRs. We have used the statistical measures of precision and recall [62] to evaluate the results. Let's first consider the case of RDMSim. Under DC1, the logistic regression model shows an accuracy score of 0.971 for MC with a precision and recall of 0.977 and 0.989, respectively. For MR and MP, the accuracy score for classification is 0.986 and 0.965, having a precision of 0.968 and 0.966, respectively. The recall for both MR and MP is 1.0. The logistic regression model has shown similar results for other dynamic context scenarios. The data set and the results for all the contexts are reported in [65]. Moreover, for the case of DELTA-IoT, the logistic regression model has been applied in a similar way to study the relationship of actual NFRs' state based on the beliefs and observations. The model shows a classification accuracy score of 0.999 for MEC with a precision and recall of 0.999 and 1.0, respectively. For RPL, the accuracy score is 0.954, having a precision and recall of 0.943 and 1.0, respectively.

The results and the behaviour aforementioned enable us to reject the null hypothesis $\boldsymbol{H}_{0,1}$*. Therefore, it answers the RQ1.*

Hypothesis $\mathbf{H}_{0,2}$ evaluation: With regards to the null hypothesis $\mathbf{H}_{0,2}$, we evaluated the behaviour of the RDM SAS under two specific scenarios: *without update* and *with the update* of stakeholders' preferences about the effects of executing adaptation actions on the system's state. RE-STORM has quantified these stochastic effects as probability distributions, i.e., beliefs about the satisfaction of the NFRs in the RDM SAS.

Because these beliefs are continuous variables, we have been able to run a two-sample comparison test to determine if there is a statistically significant difference between the means of the beliefs when the update of stakeholders' preferences is performed.

Statistical significance: The results related to the dynamic context DC_1 are depicted in Table 5. In all cases, we found a statistically significant difference between the means of the satisfaction of the NFRs while updating or not updating stakeholders' preferences. For the case of *Maximization of Reliability* (MR), Table 5(a) shows that the update of stakeholders' preferences effectively contributes towards a higher belief about the satisfaction of MR (mean = **0.93**, SD = 0.058) compared to the belief when stakeholders' preferences are not updated (mean = **0.86**, SD = 0.038). These results are statistically significant (Table 5(b): t = **19.708**, p-value = **.000**). In Table 5(a), it is observed that the update of stakeholders' preferences produces a lower belief about the satisfaction of *Minimization of Cost* (MC) (mean = **0.86**, SD = 0.056) compared to the belief when stakeholders' preferences are not updated (mean = **0.88**, SD = 0.041). The results are significant (Table 5(b): t = **12.129**, p-value = **.000**). Finally, Table 5(a) shows that the update of stakeholders' preferences also produces a lower belief about the satisfaction of *Maximization of Performance* (MP) (mean = **0.83**, SD = 0.066) compared to the belief when stakeholders' preferences are not updated (mean = **0.88**, SD = 0.055). The results are also significant (Table 5(b): t = **17.771**, p-value = **.000**). In general, during the experiments under different dynamic contexts, it was observed that when initial stakeholders' preferences are not updated, the beliefs about the satisfaction of NFRs can be drastically compromised, e.g., dynamic contexts DC_1 (Figure 3(b)), DC_3 (Figure 11) and DC_6 (Figure 13). These beliefs can go even below their required thresholds due to the unsuitability of the initial stakeholders' preferences.

Table 5. DC_1 - NFRs Statistics and Independent Samples t-tests for Equality of Means

(a) DC_1 - NFRs Statistics

		N	Mean	Std. Deviation	Std. Error Mean
MC	Not update of preferences	1000	**0.8836**	0.0411	0.0013
	Update of preferences	1000	**0.8567**	0.0567	0.0018
MR	Not update of preferences	1000	**0.8674**	0.0879	0.0028
	Update of preferences	1000	**0.9327**	0.0571	0.0018
MP	Not update of preferences	1000	**0.8802**	0.0554	0.0018
	Update of preferences	1000	**0.8317**	0.0662	0.0021

(b) DC_1 - Independent Samples t-tests

		t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference
MC	Equal variances assumed	12.129	1998	0.000	0.0269	0.0022
	Equal variances not assumed	12.129	1822.795	**0.000**	0.0269	0.0022
MR	Equal variances assumed	-19.708	1998	0.000	-0.0653	0.0033
	Equal variances not assumed	-19.708	1713.600	**0.000**	-0.0653	0.0033
MP	Equal variances assumed	17.771	1998	0.000	0.0485	0.0027
	Equal variances not assumed	17.771	1938.621	**0.000**	0.0485	0.0027

In contrast, when the stakeholders' preferences are updated, the results show that the decision-making process improves the NFR with the lowest belief about its satisfaction, taking it eventually to a suitable zone.[4] As a tradeoff, a slight reduction in the beliefs about the satisfaction of the other NFRs involved can also be observed. Moreover, the aggregated results presented in Section 6.1.4 also show that the reassessment and update of preferences and the decision-making by RE-STORM can improve the general performance of the RDM SAS.

The results and findings reported above enabled us to reject the null hypothesis $\boldsymbol{H}_{0,2}$ *and answers* ***RQ2****.*

Next the Research questions will be discussed based on the results and the insights gained from the experiments.

7.2 Research Questions Revisited

In this section, we highlight our contributions by answering the research questions of this article. Two research questions identified in this work were presented in Section 6. Answers to these questions are depicted as follows:

RQ1: *Can the uncertainty associated with the stochastic effects of executing an adaptation action be quantified?*

We have shown how the analysis activity of RE-STORM, which is part of the approach's runtime behaviour embedded in a MAPE-K loop, allows for the quantification of uncertainty of the effects of adaptation actions. The system's state is not directly observable; however, the beliefs about the effects on the system's state of executing an adaptation action are updated based on collected observations. Specifically, Equation (1), applies the use of Bayesian Inference (see Section 2), to quantify the uncertainty of the current system's state based on probability distributions. In Section 7.1 and based on the case studies, it has been shown that the belief representation of the system's state is statistically significant in predicting the system's state.

[4]Unusual high SLAs (see examples in Appendix B) or extreme hostile environments (see dynamic context DC_6 in Appendix A) could prevent an NFR from being taken to a suitable zone of satisfaction.

*Therefore, by answering the **RQ1**, it has been shown the first contribution: a decision-making technique for self-adaptation based on partially observable NFRs.*

***RQ2:** Under new and unexpected contexts observed, can the tradeoff among the NFRs be improved by updating the stakeholders' preferences based on the new knowledge found at runtime ?*

In Section 6, we have shown that the tradeoffs among NFRs can be improved by reassessing and updating stakeholders' preferences about the effects of executing adaptation actions on the system's state. To this effect, we have used the *Weights Updater module* presented as part of the Planning step 1 of RE-STORM (see Section 5.2). The module shows how the original stakeholders' preferences inhibited better adaptation choices that would allow achieving higher satisfaction of the NFRs regarding their SLAs and that the reassessment of the preference values is therefore needed. The results evaluated in Section 7.1 have shown that when the original stakeholders' preferences that do not match the current runtime context are updated with better-fitting values, the tradeoff and the decision-making in a SAS are improved accordingly. Under complex and extremely hostile environments (e.g., dynamic context DC_6) or unusually high SLAs (see Appendix B), the tradeoff performed by the approach may not necessarily guarantee the fulfilment of its SLA. However, better results were observed in syster with the original values.

*Our main aim in approaching research question **RQ2** was to explore further opportunities to reassess and improve the tradeoff among NFRs due to new knowledge acquired during runtime supported by RE-STORM. Specifically, we focused on the reassessment and update of stakeholders' preferences to improve the tradeoffs among the NFRs in a system.*

7.3 Threats to Validity

In this section, the main threats that might have an impact on the validity of the results of this work are presented.

- ***Internal validity:*** Internal validity refers to the degree of confidence that relationships being tested are not influenced by other factors and whether the evidence supports our claims [11, 63]. Accordingly, there is a potential risk when we try to determine whether the update of stakeholders' preferences affects the beliefs about satisfaction of NFRs, the beliefs may also be affected by other causes (e.g., different environmental conditions under the same evaluation). We have mitigated this threat by performing experiments with randomized dynamic contexts (see Section **??**), under which the case studies have been evaluated. Specifically, we use the same configuration (i.e., the same randomized scenario) for RDM when RE-STORM was/was not applied to update stakeholders' preferences during the system's execution.
  Another internal threat to validity is related to the extent to which the presented approach performs in an actual environmental setup. In this article, we have used a case study approach based on a simulator. The experimental results are based on the environmental factors presented by a simulated environment, not an actual physical network. However, the simulation artefacts that are selected for the case studies [30, 59] provide simulations that are closer to the real settings. Both RDMSim [64] and DELTA-IoT [30] are public software artefacts of the research community and have been used by different research teams, which adds confidence to the results.
- ***External validity:*** This aspect of validity is concerned with studying the extent to which it is possible to generalize the findings [76]. We support the generalizability of the approach by applying it to two case studies from different domains and executing experiments using two different POMDP solvers. The case studies [30, 59] that we have selected provide simulations

Table 6. Comparison of RE-STORM to other Approaches

Decision-Making Approaches	Scalability Issues	Long-term effect	Representation of Partially-Observable NFRs	Preferences Specification	Update of Preferences at runtime
Garcia-Galan et al. [28]	X	X	X	✓	X
Song et al. [55]	X	X	X	✓	✓
Letier et al. [36]	✓--	X	X	✓	X
Elahi et al. [7]	✓--	X	X	✓	X
Liaskos et al. [34]	✓--	X	X	✓	X
Peng et al. [40]	✓--	X	X	✓	✓
Sousa et al. [51]	✓--	X	X	✓	✓
Filleri et al. [25]	✓--	✓	X	X	X
Bencomo et al. [14]	X	✓	X	✓	X
Camara et al. [3]	✓--	✓	X	✓ -	X
Bowers et al. [20]	✓--	✓	X	✓	X
Moreno et al. [38]	✓--	✓	X	✓-	X
RE-STORM	✓-	✓	✓	✓	✓

X: Approach does not fulfill the criterion ✓ **-** **OR** ✓ **--** : Approach partially fulfills the criterion at different levels ✓ : Approach fulfills the criterion.

that are closer to the real environmental settings. Accordingly, we believe that our work is achievable in real settings of other domains similar to those in [30] and [59] . From the point of view of scalability, we also support the generalizability of the approach. RE-STORM uses DESPOT and Perseus, algorithms that overcome the scalability issues related to the "curse of history"and the "curse of dimensionality" [54, 56] in POMDPs. Given the current state-of-the-art, RE-STORM is a novel solution to represent the evolution of the beliefs about the satisfaction of partially observable NFRs and their tradeoff. However, the current implementation of the RDM SAS using DESPOT is based on a simulated environment provided by the algorithm [54]. Nevertheless, the simulation is based on real data [37]. In RE-STORM, the states are defined in terms of combinations of satisfaction levels of NFRs. If the number of NFRs is two, we would have four states for POMDP; for three NFRs, it would be eight, and so on. Therefore, in practice, it should target applications with not many NFRs. Therefore, while using our approach, SAS's design experts must limit their reasoning to critical NFRs to drive self-adaptation, which, according to [47], includes a considerable number of applications.

In the next section, we contrast our contribution against related work.

8 COMPARISON WITH RELATED WORK

There is a variety of approaches for decision-making under uncertainty and driven by their NFRs [7, 17, 27, 34, 36, 51, 55, 57]. Table 6 shows a summary of them. In this section, a comparison against RE-STORM is presented. We compare RE-STORM with other approaches based on the following criteria: *(1) Scalability issues*, *(2)"Long-term effects" in decision-making under uncertainty*, *(3) Representation of Partially Observable NFRs*, *(4) Stakeholders' preferences specification at design-time*, and *(5) Runtime reassessment and update of stakeholders' preferences.*

The first column, *Scalability issues,* in Table 6 is mainly related to the curse of history and the curse of dimensionality, which have been explained in Section 7.3. Garcia-Galan et al. [28] and Song et al. [55] fail to deal with such scalability issues; while the approaches [4, 7, 20, 25, 34, 36, 38, 40, 51] partially deal with these scalability problems. A limitation of RE-STORM relates to the curse of dimensionality, i.e., the fact that the states are defined in terms of combinations of satisfaction levels of NFRs, which can grow exponentially. Therefore, when using RE-STORM design experts should restrict reasoning to critical NFRs. Nevertheless, it constitutes a substantial number

of applications [47]. RE-STORM has been applied using two different POMDP implementations (DESPOT [54] and Persus [77]), which allows it to overcome the curse of "History" (i.e., in the case of RE-STORM beliefs do not grow exponentially along with the planning horizon). For instance, RE-STORM can take a maximum of 1 second to take a decision if this is required by the stakeholders. RE-STORM is considered one step ahead with respect to the other approaches towards fulfilling the scalability criterion and represented by ✓- in Table 6.

The second column, *Long-term effect*, allows us to classify the evaluated approaches in two main categories, (i) those that use reactive control decision-making [7, 27, 34, 36, 51, 55] and thus their decision-making process relies on techniques that focus on short-term effects, and (ii) those that take into consideration the long term effects of their immediate actions [4, 15, 20, 25, 38] while using sequential decision-making approaches such as Markov Decision Processes (MDPs) and Partially Observable MDPs. In the case of RE-STORM, it supports decision-making based on Bellman's principle of optimality, which considers the current state of the system while projecting future evolutions of the satisfaction of the NFRs [see Section 5.2 (Planning step 2)]. This capacity enables RE-STORM to overcome the well-known problem present in reactive approaches: i.e., choosing attractive short-term actions with potential undesirable long-term consequences [38].

The third column, *Representation of Partially Observable NFRs,* shows how well the approaches deal with full observability of the current state of the NFRs. Whether they use a general goal model [7, 27, 34, 36, 51, 55], or an implementation of an MDP [4, 20, 25, 38], or Dynamic Decision Networks [14, 15, 57]. Unlike RE-STORM these approaches do not model the uncertainty related to the satisfiability of the NFRs in a system, as they assume that the state of the NFRs is fully observable at every time step. As we argue in this article, this assumption often does not hold in reality. Instead, using RE-STORM, a belief over the state of NFRs is maintained based on observations and actions. Such beliefs (probability distributions) represent the quantification of uncertainty related to the real system's state and are used to drive the decision-making process presented in Section 5.2. Some approaches make use of Bayesian Artificial Intelligence to tackle the uncertainty in the model selection process for SAS [22, 45] however, they don't provide modelling of partial observability for NFRs.

In the fourth column, *Preferences Specification*, it is observed that most of the approaches present an explicit specification of preferences. The exception is [25], which is oblivious to the specification of preferences. The other approaches range from the specification of preferences provided by system stakeholders [17, 51, 55] to preferences determined by using a simulator, such as the case in [20]. Some authors use weights to specify the stakeholders' preferences [3, 38]. Specifically, authors in [38], as in our case, use the rewards based on utility functions to specify the preferences included in the SLAs; however, they have done it in an ad hoc manner for a specific pair of preferences. The latter is the reason why both [3, 38] were qualified to be in the process of fulfilling the criterion *Preferences Specification.* In RE-STORM, the initial preferences provided by system stakeholders are encoded by using the requirements-aware runtime model using POMDPs. During runtime, these preferences can be reassessed and updated if needed, as explained in the next paragraph.

Finally, the fifth column, *Update of Preferences,* shows that most approaches do not update preferences at runtime, except for the approaches in [40], [51], and [55] and RE-STORM. Moreover, the update of preferences in [51] and [55] is not autonomous. Different from the above, we have shown how initial assumptions at design-time (i.e., stakeholders' preferences) can be updated during the system's execution to explore opportunities to improve the tradeoffs among NFRs according to new contexts, which may not have been foreseen. Specifically, we support runtime preferences reassessment and update by using the *Weights Updater module* [26] presented in Section 5.2. Furthermore,

we also have a complementary approach to support autonomous tuning of NFRs priorities to keep SLAs compliant in an autonomous system [60, 61] using Multi-Reward POMDPs.

9 CONCLUDING REMARKS AND NEXT STEPS

In this article, we have presented the RE-STORM approach to account for the quantification of uncertainty of the possible fluctuations of the effects of adaptation actions based on partial observability of the state of an SAS using the model of a POMDP. The state of the SAS is represented by the levels of satisfaction associated with the NFRs. The POMDP that supports RE-STORM balances the required tradeoff of the conflicting NFRs over time. Instead of assuming fixed effects of the adaptation actions on the system over time, which is common in traditional approaches, RE-STORM models the effects of the adaptation actions over the state of the SAS by using probability distributions that change depending on the observations captured during runtime.

Possible future steps include:

Runtime Learning of Stakeholders' Preferences: Our work has shown the potential of runtime updates of stakeholders' preferences to improve the tradeoffs among NFRs and, therefore, the decision-making in SAS. We have used the implementation in [26], but different approaches can be used given the modular architecture of RE-STORM. We believe that a feasible next step is using Bayesian and machine learning approaches to compute optimal or approximately optimal preferences that fit new environmental contexts detected at runtime. At the present stage, we have obtained initial results on a new approach for autonomous tuning of NFRs' priorities [60, 61] to maintain the SLAs in an SAS based on data gathered at runtime.

Further Ways to Quantify Uncertainty. A future version of RE-STORM will extend its current system's state with soft requirements (e.g., human values [43, 44]) to quantify uncertainty as beliefs about the effects on them of executing adaptation actions. We envision the modelling and continuous monitoring of soft requirements to collect runtime insights regarding potential violations of stakeholders' values and other soft requirements. Collected insights in this context may suggest improving design features in a software product or represent socially oriented implications for systems at a higher level.

APPENDICES

A APPENDIX: ADDITIONAL DYNAMIC CONTEXTS

This appendix shows details on the behaviour of the RDM SAS under the dynamic contexts DC_2 and DC_4, DC_3, DC_5, and DC_6.

A.1 Dynamic Contexts DC_2 and DC_4

Both contexts represent situations where despite dynamic changes at the environment, the current RDM configuration and preferences still keep the beliefs about the satisfaction of the NFRs over their thresholds (see Figures 9 and 10: Behaviour under dynamic context DC_2 and DC_4 *without* update of preferences).

Nevertheless, the reward values R(s,a) were updated in DC_2 and DC_4, when in specific time slices during the system's execution, the beliefs about the satisfaction of the NFRs were below their thresholds. The final result was an improvement on the reliability of the system (see Figures 9 and 10: Behaviour under dynamic context DC_2 and DC_4 *with* update of preferences). As a tradeoff effect, a reduction on the beliefs about the satisfaction of cost and performance has been observed, but still, they were able to meet their SLAs.

Statistical Significance. The logistic regression analysis shows that for *Minimization of Cost* (MC) (see Table 7), our belief representation had an 85.2% of success rate to predict the actual system's

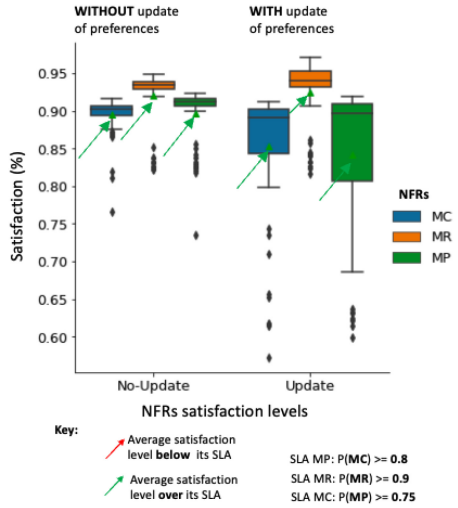


Fig. 9. RDM SAS - behaviour under dynamic context DC_2.

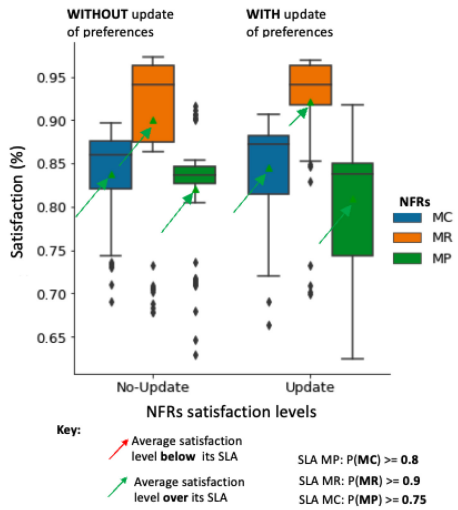


Fig. 10. RDM SAS - behaviour under dynamic context DC_4.

state (p-value = **0.000**, Wald statistic = **40.346**). In the case of *Maximization of Reliability* (MR), the success rate for predicting the real system's state was of 93.3% (p-value = **0.000**, Wald statistic = **37.709**) and for *Maximization of Performance* (MP), a success rate of 82.8% was obtained (p-value = **0.000**, Wald statistic = **56.831**).

For the dynamic context DC_4, it was observed that for *Minimization of Cost* (MC) (see Table 8), the predictive capacity of the real system's state by the belief representation had a success rate of 84.2% (p-value = **0.000**, Wald statistic = **24.440**). In the case of *Maximization of Reliability* (MR), the success rate for predicting the real system's state was of 92.5% (p-value = **0.000**, Wald statistic = **29.000**) and for *Maximization of Performance* (MP), a success rate of 79.9% was obtained (p-value = **0.000**, Wald statistic = **46.826**).

Table 7. DC_2 - Logistic Regression Models

Non-functional requirement (NFR)	Overall Percentage (%) Predicted for actual state	B	S.E.	Wald	df	Sig.
MC	**85.200**	5.859	0.922	40.346	1	**0.000**
MP	**82.800**	6.422	0.852	56.831	1	**0.000**
MR	**93.300**	15.282	2.489	37.709	1	**0.000**

Table 8. DC_4 - Logistic Regression Models

Non-functional requirement (NFR)	Overall Percentage (%) Predicted for actual state	B	S.E.	Wald	df	Sig.
MC	**84.200**	6.294	1.273	24.440	1	**0.000**
MP	**79.900**	7.145	1.044	46.826	1	**0.000**
MR	**92.500**	6.718	1.248	29.000	1	**0.000**

A.2 Dynamic Context DC_3

In this context, the effects on the system's state of the new environment produced an important reduction on the reliability of the system. The reliability was under its satisfaction threshold (see Figure 11: behaviour under dynamic context DC_3 *without* update of preferences). Despite the new detected context, the preferred topology continued to be MST: 100% (see Figure 4: selected topology *without* update of preferences).

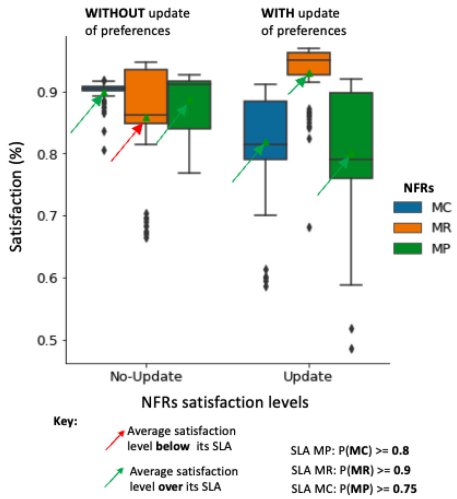


Fig. 11. RDM SAS - behaviour under dynamic context DC_3.

After the reassessment of stakeholders' preferences performed by the *Weights Updater module* (Planning step 1 of RE-STORM), the preferences were updated at runtime, and the reliability of the system was taken by the decision-making process to levels where its average satisfaction addressed the required SLA (see Figure 11: behaviour under dynamic context DC_3 *with* update of preferences). A slight reduction on the performance and cost was also observed due to the tradeoffs among

NFRs. However, this reduction does not imply any risk to continue meeting the SLAs of cost and performance: both NFRs were over their thresholds.

Statistical Significance. Results of the logistic regression analysis show that for *Minimization of Cost* (MC) (see Table 9), the predictive capacity of the real system's state by the belief representation had a success rate of 81.0% (p-value = **0.000**, Wald statistic = **37.959**). In the case of *Maximization of Reliability* (MR), the success rate for predicting the real system's state was of 91.8% (p-value = **0.000**, Wald statistic = **37.959**); finally, for *Maximization of Performance* (MP), we obtained a success rate of 81.5% (p-value = **0.000**, Wald statistic = **50.125**).

Table 9. DC_3 - Logistic Regression Models

Non-functional requirement (NFR)	Overall Percentage (%) Predicted for actual state	B	S.E.	Wald	df	Sig.
MC	**81.000**	4.970	0.807	37.959	1	**0.000**
MP	**81.500**	5.439	0.768	50.125	1	**0.000**
MR	**91.800**	8.046	1.229	42.847	1	**0.000**

A.3 Dynamic Context DC_5

The dynamic context DC_5 represents environments where the current impact of the RT topology over the beliefs about the satisfaction of the NFRs is less favourable, i.e., the probability $P(\mathbf{NFRs}_{t+1} = \text{True} \mid \text{NFR}_t, \mathbf{RT}_t)$ has been reduced with respect to the *stable conditions* of the RDM SAS. Under *stable conditions*, the RT topology has a perceived positive impact on the reliability of the system, and a less favourable impact on the cost and performance.

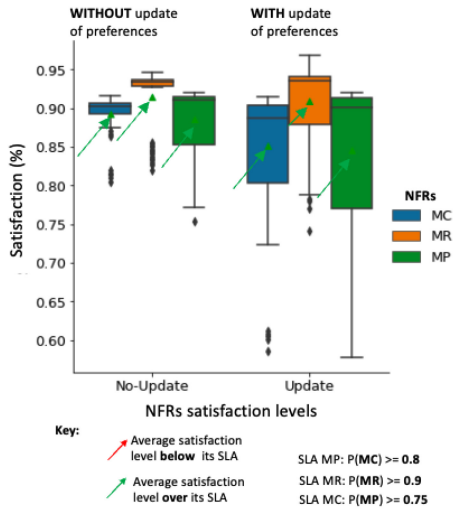


Fig. 12. RDM SAS - behaviour under dynamic context DC_5.

However, in DC_5, the positive impact on the reliability of the system has been reduced, while the negative impacts on the cost and performance have been incremented to take into account the changes introduced by this dynamic context. One immediate consequence of this context upon the behaviour of the system is the exclusive use of the MST topology: 100% (see Figure 4: selected

topology *without* update of preferences). Later, when the update of reward values R(s,a) is performed and the RT topology is used again, it can be seen that it is used 37.1% (see Figure 4: selected topology *with* update of preferences). However, there is no increment on the beliefs about the satisfaction of reliability (MR) (see Figure 12: behaviour under dynamic context DC_5 *with* update of preferences), because as we stated above, the positive impact on the reliability of the system has been reduced. The final result under this context is a tradeoff among NFRs, with a slight reduction on the average satisfaction of reliability, and also a reduction in cost and performance. Note, however, that their average satisfaction still meet their SLAs (see Figure 12: behaviour under dynamic context DC_5 *with* update of preferences).

Table 10. DC_5 - Logistic Regression Models

Non-functional requirement (NFR)	Overall Percentage (%) Predicted for actual state	B	S.E.	Wald	df	Sig.
MC	**83.800**	6.740	0.874	59.469	1	**0.000**
MP	**83.100**	5.423	0.812	44.623	1	**0.000**
MR	**89.500**	10.937	1.799	36.969	1	**0.000**

Statistical Significance. Results for this context show that for *Minimization of Cost* (MC) (see Table 10), the predictive capacity of the real system's state by the belief representation had a success rate of 83.8% (p-value = **0.000**, Wald statistic = **59.469**). In the case of *Maximization of Reliability* (MR), the success rate for predicting the real system's state was of 89.5% (p-value = **0.000**, Wald statistic = **36.969**) and for *Maximization of Performance* (MP), we obtained a success rate of 83.1% (p-value = **0.000**, Wald statistic = **44.623**).

A.4 Dynamic Context DC_6

Finally, in the dynamic context DC_6, the most hostile environment designed for this evaluation was studied. DC_6 reflects the negative impacts of the dynamic contexts DC_4 and DC_5 simultaneously. Under DC_6, when applying RE-STORM with no reassesment of stakeholders' preferences, it was found that regardless the adaptation action selected by the RDM SAS, the environment showed a tradeoff behaviour with a tendency to increment the cost (MC) while reducing the reliability (MR) and performance (MP). The satisfaction of all the NFRs in DC_6 were lower than those shown in the experiments with the *stable conditions*. When repeating the experiments of DC_6 but using RE-STORM with the update of the reward values R(s,a), it was found that the satisfaction of the reliability of the system was incremented. However, the increment was not enough to meet its SLA. Additionally, as a result of the tradeoffs to give priority to reliability, the satisfaction of the performance was reduced to a level below its threshold (see Figure 13: behaviour under dynamic context DC_6 *with* update of preferences). In the cases of the beliefs about the satisfaction of MR and MP, they are not in compliance with their SLAs, due to the current extremely hostile dynamic context (see Figure 13: behaviour under dynamic context DC_6 *with* update of preferences).
Statistical Significance. The analysis has shown in this context that for *Minimization of Cost* (MC) (see Table 11), the predictive capacity of the real system's state by the belief representation had a success rate of 74.9% (p-value = **0.000**, Wald statistic = **37.721**). In the case of *Maximization of Reliability* (MR), the success rate for predicting the real system's state was of 80.5% (p-value = **0.000**, Wald statistic = **85.105**) and for *Maximization of Performance* (MP), a success rate of 73.2% was obtained (p-value = **0.000**, Wald statistic = **44.232**).

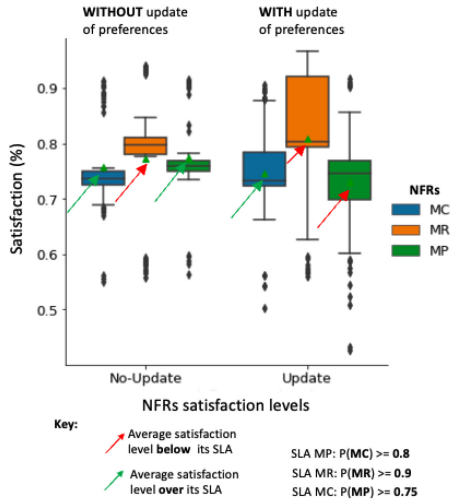


Fig. 13. RDM SAS - behaviour under dynamic context DC_6.

In the dynamic context DC_6, the update of stakeholders' preferences contribute to improving the belief about the satisfaction of the reliability of the system. However, it can not be taken to a level to meet its SLA.

Table 11. DC_6 - Logistic Regression Models

Non-functional requirement (NFR)	Overall Percentage (%) Predicted for actual state	B	S.E.	Wald	df	Sig.
MC	**74.900**	4.484	0.730	37.721	1	**0.000**
MP	**73.200**	4.477	0.673	44.232	1	**0.000**
MR	**80.500**	6.863	0.744	85.105	1	**0.000**

Under extreme hostile contexts, e.g., dynamic context DC_6 or contexts with virtually unreachable SLAs (see examples in Appendix B), a better tradeoff of the NFRs is still performed by RESTORM. To determine how suitable is the approach under those specific contexts, would depend on the stakeholders' preferences. For example, if the reliability of the system is considered to be the most critical NFR, it can be seen as valid to tolerate low satisfaction of other NFRs even if the improvement on reliability is too small that has not reached its SLA.

B APPENDIX: RDM SAS UNDER DIFFERENT SERVICE LEVEL AGREEMENTS

In this appendix, we evaluate the behaviour of the RDM SAS using different Service Level Agreements (SLAs) as those presented in Section 4.3.

B.1 SLAs: Less Strict Scenario

If the SLAs, different from the ones established in Section 4.3, were less strict, for example:

- by keeping the SLA related to the cost of the system as P(MC = True) >= **0.7**
- by reducing the required reliabiliy from P(MR = True) >= **0.90** to P(MR = True) >= **0.8** and
- by reducing the performance from P(MP = True) >= **0.75** to P(MP = True) >= **0.6**

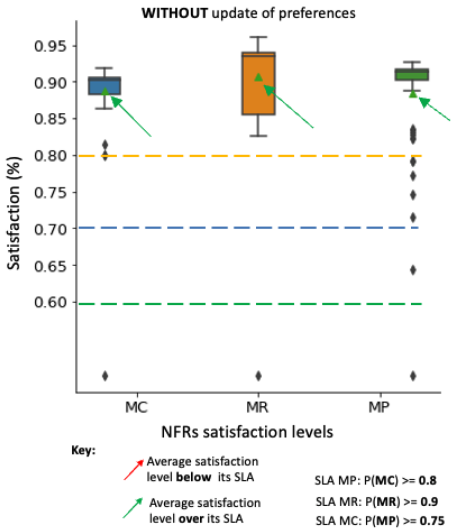


Fig. 14. Beliefs about NFRs satisfaction - Less strict scenario.

Under this scenario (see Figure 14), the satisfaction of the NFRs would always meet their SLAs as expected. Thus, there is no need to explore opportunities to improve the tradeoffs among NFRs by updating stakeholders' preferences. The *Weights Updater module* of RE-STORM would not be called.

B.2 SLAs: Stricter Scenario

In the following two examples, the behaviour of the RDM SAS under a stricter scenario is described.

(1) **SLAs example 01:** In this example, the required threshold of satisfaction for the cost, reliability and performance of the system has been incremented. The new established SLAs are: P(MC = True) >= **0.80**, P(MR = True) >= **0.95** and P(MP = True) >= **0.85**.
Beliefs about NFRs satisfaction - It is observed in Figure 15: Beliefs about NFRs satisfaction *without* update of preferences, that the reliability of the system has the lowest satisfaction in relation to its SLA. Therefore, the stakeholders' preferences will be updated by the *Weights Updater module* accordingly to this context. The decision-making provided by RE-STORM, taking into account the new preferences, will improve the reliability but with a reduction on the cost and the performance of the system due to the tradeoff performed. Figure 15: Beliefs about NFRs satisfaction *with* update of preferences, shows the new satisfaction of the NFRs after the update of the preferences. The average satisfaction for the NFRs shown in Figure 15 after the update is MC = **0.8608**, MR = **0.9373** and MP = **0.8354**. Although, it is possible to observe in the figure above that the satisfaction of reliability is *during some scarce time slices* (upper whisker) over its threshold, on average, it was not possible to meet its SLA after the update of stakeholders' preferences.

RE-STORM improves the NFR with the lowest satisfaction concerning its SLA. However, unusual high SLAs may prevent taking the NFR to a suitable zone of satisfaction.

(2) **SLAs example 02:** Different from the previous example, an even higher threshold of satisfaction for the reliability of the system is required in this example. The new established SLAs

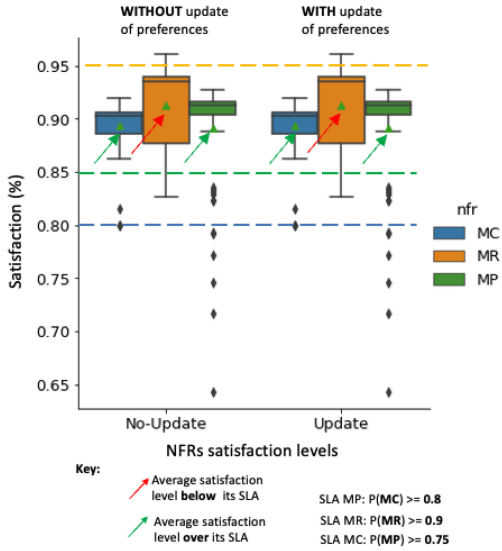


Fig. 15. Beliefs about NFRs satisfaction - SLAs example 01.

are: P(MC = True) >= **0.80**, P(MR = True) >= **0.99** and P(MP = True) >= **0.85**. Figure 16: Beliefs about NFRs satisfaction *without* update of preferences, shows the behaviour of the RDM SAS before the update of the stakeholders' preferences.

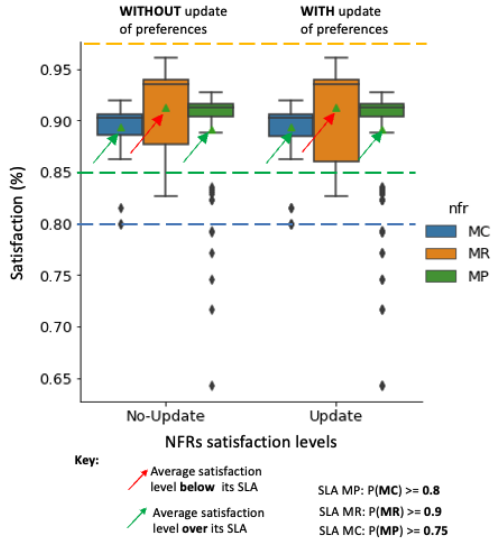


Fig. 16. Beliefs about NFRs satisfaction - SLAs example 02.

In Figure 16: Beliefs about the NFRs satisfaction *without* update of preferences, it is observed that the belief about the satisfaction of the reliability is always below its SLA. Therefore, the stakeholders' preferences are updated by the *Weights Updater module* accordingly to this context. Then, Figure 16: Beliefs about the NFRs satisfaction *with* update of preferences,

shows the new beliefs about satisfaction of the NFRs. In this case, regardless of the increment of the average satisfaction of the reliability of the system in comparison to Example 01 (from **0.9373** to **0.9418**), its satisfaction is constantly under its SLA, as is shown in Figure 16.

Even if not always possible to address the SLAs in a system under extreme conditions, RE-STORM is able to perform the tradeoff of the NFRs independently of the SLAs in use.

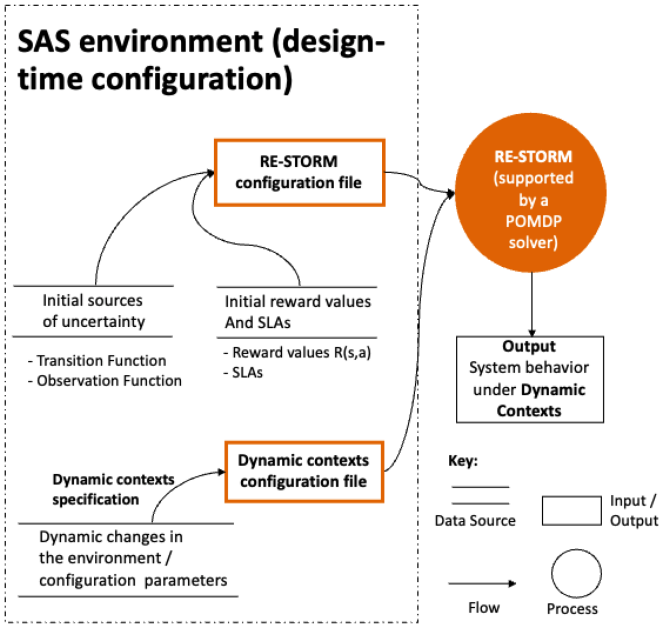


Fig. 17. Initial setup of RE-STORM: inputs and output.

C APPENDIX: SETUP OF RE-STORM

This appendix presents details on the initial configuration of RE-STORM to implement its runtime behaviour (see Figure 17). The initial representation of the *SAS environment* should be specified at design-time to be used during the system's execution. The configuration files of this environment are briefly described as follows:

C.1 RE-STORM Configuration File

The main parameters of the RE-STORM configuration file are presented as follows:

- **Transition Function.** It is a set of real values between [0,1]. They represent the probability distributions of the transition function T (s, a, s') = P(s' |s, a) in a POMDP. Details on their specification in terms of the NFRs of a SAS have been presented in Section 3.1. The specific values assigned to this parameter for the RDM SAS have been presented in the Tables 2(a)–2(c).
- **Observation Function.** It is a set of real values between [0,1]. They represent the probability distributions of the observation function O(s', a, z) = P(z|s', a) in a POMDP. Details on their specification in terms of the MON variables of a SAS have been presented in Section 3.2. The specific values assigned to this parameter for the RDM SAS have been presented in Table 3(a).

— **Reward Values R(s,a).** It is a set of real values between [0,1]. They represent the initial preferences of the system's stakeholders, i.e., the obtained reward $R(s, a)$ after taking action $a \in A$ at time t, to arrive to the new state $s \in S$ at time $t + 1$. Details on their specification have been presented in Section 4.6. The values assigned to this parameter for the RDM SAS have been presented in Table 3(b).
— **Thresholds for the Levels of Satisfaction of NFRs.** It is a set of real values between [0,1]. The thresholds of satisfaction represent the *Service Level Agreements* (SLAs) to be monitored during the system's execution. RE-STORM uses them to trigger the need to update reward values R(s,a) in a POMDP. The default values used for the NFRs *Minimization of Cost* (MC), *Maximization of Reliability* (MR) and *Maximization of Performance* (MP) are **[0.7, 0.9, 0.75]**. Details on their specification have been presented in Section 4.3. Other SLAs have also been evaluated and reported in Appendix B.

Next, the configuration file to enable the behaviour of the dynamic contexts DC_1 to DC_6 is described.

C.2 Dynamic Contexts Configuration File

This configuration file supports further simulation of the SAS under different dynamic environments. Its main parameters are:

— **Dynamic Context DC_i to be Activated.** It is an integer value (between 0 and 5) which represent the dynamic context DC_i to be activated.
— **Noise Factor.** It is a real value between [0,1] which represent the probability of activation of the selected dynamic context DC_i. The default value used during this evaluation is 0.5.
— **Deviation Range of the Selected Dynamic Context DC_i.** It is a range of values [lowerBound, upperBound], from which a *real number* is randomly selected to decrease specific probability values (see Tables 2(a)–2(c)), accordingly to the dynamic context selected. For example, for the range **[0.1, 0.15]** and the dynamic context DC_1: P($\mathbf{MR}_{t+1}$ = True|NFR_t , $\mathbf{MST}_t$), a random value between **10**% and **15**% will be selected to reduce the current positive impact of the topology MST_t over the realiability of the system (MR_{t+1} = True).
— **Length of the Selected Dynamic Context DC_i.** It is a range of values [lowerBound, upperBound], from which an *integer number* is randomly selected to specify the number of time slices that the dynamic context is performed. The default range of values used is **[5, 15]** .
— **Flag to Update Reward Values R(s,a).** It is an integer value (0 or 1) to determine if the *Weights Updater module* of RE-STORM is performed when an NFR is detected below its threshold of satisfaction (i.e., below its SLA). For example, the behaviour for the dynamic context DC_1 reported as *without update of preferences* has been obtained by using the **flag value 0**. Conversely, the behaviour reported as *with update of preferences* has been obtained with the **flag value 1**.

D APPENDIX: DETAILS ON THE RE-STORM IMPLEMENTATION SUPPORTED BY THE DESPOT ALGORITHM

In this appendix, a further explanation of the planning activity of RE-STORM, represented by Algorithms 1, 2 and 3, is presented. Additional details on the implementation of the online POMDP solver tool can be accessed at the DESPOT toolkit repository [6].

Algorithm 1, provides a high-level view of the process to build, search and choose an action from a DESPOT tree [54] (see Figure 2(b) for a DESPOT tree example). It constructs and searches

ALGORITHM 1: Algorithm for building and searching a DESPOT tree D in each time slice

1: Parameter(s):
2: - Initial belief b_0 about the current satisfaction level of the NFRs
3: Runtime execution:
4: -Use the belief b_0 to create the root node of a new DESPOT tree D
5: -Initialize upper and lower bounds: $\mu(b_0)$ and $l(b_0)$
6: -Initialize $\epsilon(b_0) \leftarrow \mu(b_0)$ - $l(b_0)$
7: **while** $\epsilon(b_0) > \epsilon_0$ and running time is less than T_{max} **do**
8: b ← Explore(D,b_0) ▷ (Explore a promising path)
9: Backup(D,b) ▷ (Backup on bounds at each node b)
10: **end while**
11: **Return:** DESPOT tree with an approximated optimal policy $\pi^*(b_0)$

a DESPOT tree increasingly. Initially, it contains only a root node with belief b_0 about the current satisfaction of the NFRs in a system. The tree also contains the initial upper and lower bounds associated to the belief b_0 (lines 4–5). The algorithm performs explorations using Algorithm 2, to expand the DESPOT tree and to reduce the gap $\epsilon(b_0)$ between the bounds $\mu(b_0)$ and $l(b_0)$ at the root node b_0. Each exploration aims at choosing and expanding a promising leaf node (line 8) and adds its child nodes into the tree. For each new child, initial bounds $\mu(b)$ and $l(b)$ are computed. The process continues until the current leaf node is not heuristically promising [54]. Then, the algorithm traces the path back to the root and performs backup using Algorithm 3, i.e., the upper and lower bound values $\mu(b)$ and $l(b)$ are updated at each tree node along the way to the root node (line 9). The updated values at the root node b_0, are used to compute a new $\epsilon(b_0)$ factor. The explorations continue until the gap between the bounds, i.e., $\epsilon(b_0) = \mu(b_0)$ - $l(b_0)$, reaches the target level ϵ_0 (where $\epsilon_0 >= 0$) or the planning time T_{max} finish (line 7). Then, the system executes the first action of the policy $\pi^*(b_0)$, i.e., the action branch a^* with the highest upper bound $\mu(b_0, a)$.

ALGORITHM 2: Algorithm to expand the branches of a DESPOT tree D

1: Parameter(s):
2: -A DESPOT tree D and the current belief b
3: Runtime execution:
4: -$D_h \leftarrow$ DESPOT tree height
5: -Δ(b) ← current height of the belief b
6: **while** Δ(b) <= D_h and E(b) > 0 **do**
7: **if** b is a leaf node in D **then**
8: Expand b one level deeper. Insert each new child b' of b into D,
9: and initialize $\mu(b')$ and $l(b')$
10: $a^* \leftarrow \arg\max_{a \in A} \mu(b, a)$
11: $z^* \leftarrow \arg\max_{z \in Z_{b,a^*}} E(\tau(b, a^*, z))$
12: b ← $\tau(b, a^*, z^*)$
13: **end if**
14: **end while**
15: **Return:** An expanded DESPOT tree

In Algorithm 2, the exploration to expand the DESPOT tree starts at the root node b_0. At each node b along the exploration path, the best action branch a^* (i.e., the topologies MST or RT in the

RDM SAS) is selected, according to the upper bound $\mu(b, a)$ (line 10). Afterwards, the observation branch z that leads to a child node $b' = \tau(b, a^*, z)$ maximizing the excess uncertainty $E(b')$ [32, 54], is selected (line 11). The excess uncertainty $E(b')$ measures the difference between the current gap at b' and the expected gap at b' if the target gap $\epsilon(b_0)$ at b_0 is satisfied. Each node in the tree is created by a simulation model provided by the DESPOT algorithm [54] as part of a strategy to compute an approximated optimal policy [54]. Initial upper and lower bounds $\mu(b)$ and $l(b)$ are created at each tree node. These bounds are influenced by the current reward values R(s,a) of the system. As stated in Definition 1, the reward values correspond with stakeholders' utility values associated with the state of the system (NFRs in RE-STORM) and adaptation actions. Therefore, update of preferences due to changes of utility values in the Planning step 1 (See Section 5.2), can determine new bound values and thus a different adaptation action selected by the policy $\pi(b_0)$. The exploration at a node b is terminated under the following conditions (line 6). First, Δ(b) <= D_h, i.e., the maximum tree height is exceeded, and second, $E(b) > 0$, indicating that when the expected gap at b is reached, further exploration from b onwards may be unproductive. When the exploration terminates, Algorithm 3 is performed. In Algorithm 3, the path back to the root node is traced and backup is performed using the Bellman's principle of optimality to recompute the bounds $\mu(b)$ and $l(b)$ at each node b of the tree [54].

ALGORITHM 3: Algorithm to perform backup on the bounds of each node b using Bellman's principle

```
1: Parameter(s):
2: -A DESPOT tree D and the current belief b
3: Runtime execution:
4: for each node x on the path from b to the root D do
5:     Perform backup on μ(x) and l(x)
6: end for
```

E APPENDIX: DETAILS ON THE RE-STORM IMPLEMENTATION SUPPORTED BY THE PERSUES ALGORITHM

In this appendix, an explanation of the planning activity of RE-STORM, using an alternative POMDP solver is presented. Additional details on the implementation of this solver can be accessed at [77].

We have used Perseus algorithm as an alternative for planning. Perseus is based on the **Point-Based Value Iteration (PBVI)** framework [50]. The algorithm is divided into two main parts.

(1) Random Exploration of the belief space to collect the belief values for NFRs.

(2) Update of the Value function using Bellman backup [56].

In point-based methods, the Value function V is represented in the form of α-vectors presented as:

$$V = \alpha = \begin{bmatrix} V(s_1) \\ V(s_2) \\ \dots \\ V(s_n) \end{bmatrix} \tag{8}$$

Hence, the Value over belief is represented as:

$$V_b = \alpha.b \tag{9}$$

ALGORITHM 4: Perseus: Random Exploration

Initialize $b \leftarrow b_0$
repeat
  select $a \in A$ randomly
  $o = performAction(a)$
  $b^{a,o} \leftarrow beliefUpdate(b, a, o)$
  Add $b^{a,o}$ to B
  $b \leftarrow b^{a,o}$
until $|B| \neq n$
return B

ALGORITHM 5: Perseus: Value Function Update

Initialize $V \leftarrow V_0 \leftarrow \alpha_{min}$
repeat
  $B' \leftarrow B$
  $V' \leftarrow \emptyset$

  repeat
    Randomly pick $b \in B'$
    $\alpha_{new} \leftarrow PointBasedBackup(b, V)$

    if $\alpha.b \geq V_b$ **then**
      $B' \leftarrow b \in B' : \alpha.b < V_b$
      $\alpha_b \leftarrow \alpha_{new}$

    else
      $B' \leftarrow B' - b$
      $\alpha_b \leftarrow argmax_{\alpha \in V} \alpha.b$

    end if
    $V' \leftarrow V' \cup \alpha_b$

  until $B' = \emptyset$
until V converges
return V, $B \leftarrow \emptyset$

During the first step of random exploration, action is randomly selected and executed in offline mode to generate an observation. Consequently, a new belief sample point is computed, based on the selected action a and observation o, using the belief update procedure as follows [56]:

$$b^{a,o}(s') = Pr(s'|b, a, o) = \frac{Pr(s', b, a, o)}{Pr(b, a, o)} \tag{10}$$

$$= \frac{Pr(o|s', b, a)Pr(s'|b, a)Pr(b, a)}{Pr(o|b, a)Pr(b, a)} \tag{11}$$

$$= \frac{O(s', a, o) \sum_{s \in S} Pr(s'|b, a, s)Pr(s|b, a)}{Pr(o|b, a)} \tag{12}$$

where

$$Pr(o|b,a) = \sum_{s \in S} b(s) \sum_{s' \in S} T(s,a,s')O(s',a,o) \tag{13}$$

Once the belief sample set is collected, the initial set of α-vectors are computed as:

$$R_{min} = minR(s,a) \tag{14}$$

$$\alpha_{min} = R_{min}/1-\gamma \tag{15}$$

$$V_0 = \{\alpha_{min}\} \tag{16}$$

Next, the value function update is performed iteratively using the point-based Bellman backups to compute the optimal α-vector. For each iteration, a set B' = B and a new Value function V' =$\emptyset$ are initialized. A belief point, randomly selected from B', is used for the computation of the new α-vector by applying the point-based Bellman backup. As an example, for k+1 iteration, the backup can be calculated as follows:

$$backup(b,V) = \arg\max_{\alpha_{k+1}^{b,a}} b.\alpha_{k+1}^{b,a} \tag{17}$$

where

$$\alpha_{k+1} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{g^{a,o}} b.g^{a,o} \tag{18}$$

$$g_i^{a,o} = \sum_{s' \in S} O(s',a,o)T(s,a,s')\alpha_i(s') \tag{19}$$

The $g^{a,o}$ are the back-projections for all actions a and observations o of each next stage α-vector i.e., $\alpha_i \in \alpha_k$.

Once the new α-vector is computed, the optimal Value function is computed as follows:

If $\alpha_{new}.b > V_b$, then α_{new} is added to V' and all other belief sample points $b' \in B'$ that show an improvement in value α_{new} , i.e., $\alpha_{new}.b' > V_{b'}$ are removed from B'. Hence, they are not backed up at the current iteration. In contrast, if $\alpha_{new}.b < V_b$, then $\alpha_{old} = argmax_{\alpha \in V}\alpha.b$ is added to V'. The iteration ends when B'= $\emptyset$ and V is set to V'.

The process is repeatedly performed till the convergence of V occurs. The V is considered to be converged when the difference between V_b (value over belief) from the current iteration and the previous iteration is greater than the the precision parameter η [50]. Both parts of the Perseus algorithm as presented in [56] are shown in Algorithms 4 and 5, respectively.

ACKNOWLEDGEMENT

This work has been part funded by Leverhulme Trust Research Fellowship(Grant No.RF-2019-548/9) and the EPSRC Project Twenty20Insight(Grant No. EP/T017627/1).

REFERENCES

[1] A. Jensen A. Ramirez, and B. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive system. *SEAMS,* June 2012, pp. 99–108. (2012).

[2] Javier Camara, David Garlan, Won Gu Kang, Wenxin Peng, and Bradlet Schmerl. 2017. Uncertainty in self-adaptive systems: Categories, management, and perspectives. Institute for Software Research. Carnegie Mellon University (2017).

[3] Javier Cámara, Antónia Lopes, David Garlan, and Bradley Schmerl. 2016. Adaptation impact and environment models for architecture-based self-adaptive systems. *Science of Computer Programming* 127 (2016), 50–75.

[4] Javier Cámara, Wenxin Peng, David Garlan, and Bradley Schmerl. 2018. Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Science of Computer Programming* 167 (2018), 51–69.

[5] Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Giovanna Di Marzo Serugendo, Schahram Dustdar, Anthony Finkelstein, Cristina Gacek, Kurt Geihs, Vincenzo Grassi, Gabor Karsai, Holger M. Kienle, Jeff Kramer, Marin Litoiu, Sam Malek, Raffaela

Mirandola, Hausi A. Müller, Sooyong Park, Mary Shaw, Matthias Tichy, Massimo Tivoli, Danny Weyns, and Jon Whittle. 2009. Chapter: Software Engineering for Self-Adaptive Systems: A Research Roadmap, *Software Engineering for Self-Adaptive Systems*. Springer-Verlag, Berlin, , 1–26.

[6] DESPOT-Repo. 2018. (2018). https://github.com/AdaCompNUS/despot

[7] Golnaz Elahi and Eric Yu. 2011. Requirements trade-offs analysis in the absence of quantitative measures: A heuristic method. In *SAC (SAC '11)*. ACM, New York.

[8] Naeem Esfahani and Sam Malek. 2012. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems 2 (SEfSAS 2)*. Springer-Verlag.

[9] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artif. Intell.* 101, 1–2 (1998).

[10] Anthony O'Hagan, Caitlin E. Buck, Alireza Daneshkhah, J. Richard Eiser, Paul H. Garthwaite, David J. Jenkinson, Jeremy E. Oakley, and Tim Rakow. 2006. Uncertain judgements: Eliciting experts' probabilities. (2006).

[11] Apostolos Ampatzoglou, Stamatia Bibi, Paris Avgeriou, Marijn Verbeek, and Alexander Chatzigeorgiou. 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and Software Technology* 106 (2019).

[12] L. Baresi and C. Ghezzi. 2010. The disappearing boundary between development-time and run-time. In *FSE/SDP Workshop on Future of Software Engineering Research*.

[13] Luciano Baresi, Liliana Pasquale, and Paola Spoletini. 2010. Fuzzy goals for requirements-driven adaptation. In *RE 2010*.

[14] Nelly Bencomo, Amel Belaggoun, and Valérie Issarny. 2013. Bayesian artificial intelligence for tackling uncertainty in self-adaptive systems: The case of dynamic decision networks. In *RAISE 2013*.

[15] Nelly Bencomo, Amel Belaggoun, and Valérie Issarny. 2013. Dynamic decision networks to support decision-making for self-adaptive systems. In *(SEAMS)*.

[16] Haoyu Bai, David Hsu, and Wee Sun Lee. 2013. Planning how to learn. In *International Conference on Robotics and Automation 2013*.

[17] Nelly Bencomo and Amel Belaggoun. 2013. Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks. In *REFSQ*.

[18] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. 2015. Intention-aware online POMDP planning for autonomous driving in a crowd. In *International Conference on Robotics & Automation* (2015).

[19] Nelly Bencomo and Luis H. Garcia Paucar. 2019. RaM: Causally-connected requirements-aware models using Bayesian inference. In *IEEE/ACM MODELS*. Munich, Germany (2019).

[20] Kate M. Bowers, Erik M. Fredericks, Reihaneh H. Hariri, and Betty H. C. Cheng. 2020. Providentia: Using search-based heuristics to optimize satisficement and competing concerns between functional and non-functional objectives in self-adaptive systems. *SoSyM* 162 (2020), 110497.

[21] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. 1999. *Non-Functional Requirements in Software Engineering*. Springer.

[22] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2022. Taming model uncertainty in self-adaptive systems using bayesian model averaging. In *SEAMS*.

[23] Edwin K. P. Chong, Christopher M. Kreucher, and Alfred O. Hero III. 2009. Partially observable Markov decision process approximations for adaptive sensing. *Discrete Event Dynamic Systems* 19 (09 2009), 377–422.

[24] Antonio Filieri, Carlo Ghezzi, and Giordano Tamburrelli. 2012. A formal approach to adaptive software: Continuous assurance of non-functional requirements. *Formal Aspects of Computing* (2012).

[25] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. 2016. Supporting self-adaptation via Quantitative - Verification and sensitivity analysis at run time. *TSE* (2016).

[26] Luis H. Garcia Paucar, Nelly Bencomo, and Kevin Kam Fung Yuen. 2019. ARRoW: Automatic runtime reappraisal of weights for self-adaptation. *ACM/SIGAPP 2019* (2019).

[27] Jesús García-Galán, Liliana Pasquale, Pablo Trinidad, and Antonio Ruiz-Cortés. 2014. User-centric adaptation of multi-tenant services: Preference-based analysis for service reconfiguration. In *SEAMS 2014*.

[28] Jesús García-Galán, Liliana Pasquale, George Grispos, and Bashar Nuseibeh. 2016. Towards adaptive compliance. (2016).

[29] Heather J. Goldsby, Pete Sawyer, Nelly Bencomo, Betty H. C. Cheng, and Danny Hughes. 2008. Goal-based modeling of dynamically adaptive system requirements. In *ECBS Conf.*

[30] M. Usman Iftikhar, Gowri Sankar Ramachandran, Pablo Bollansee, Danny Weyns, and Danny Hughes. 2017. Deltaiot: A self-adaptive Internet of Things exemplar. In *SEAMS*. 76–82.

[31] Kimberly Keeton, Cipriano Santos, Dirk Beyer, Jeffrey Chase, and John Wilkes. 2004. Designing for disasters. *USENIX Conference on File and Storage Technologies, Berkeley* (2004).

[32] Hanna Kurniawati, David Hsu, and Wee Sun Lee. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Robotics: Science and Systems* (2008).
[33] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.
[34] Sotirios Liaskos, Sheila A. McIlraith, Shirin Sohrabi, and John Mylopoulos. 2011. Representing and reasoning about preferences in requirements engineering. *Requir. Eng.* 16, 3 (Sept. 2011), 23 pages. https://doi.org/10.1007/s00766-011-0129-9
[35] Rogério de Lemos, Holger Giese, Hausi A. Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M. Villegas, Thomas Vogel, Danny Weyns, Luciano Baresi, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, Ron Desmarais, Schahram Dustdar, Gregor Engels, Kurt Geihs, Karl M. Göschka, Alessandra Gorla, Vincenzo Grassi, Paola Inverardi, Gabor Karsai, Jeff Kramer, Antónia Lopes, Jeff Magee, Sam Malek, Serge Mankovskii, Raffaela Mirandola, John Mylopoulos, Oscar Nierstrasz, Mauro Pezzè, Christian Prehofer, Wilhelm Schäfer, Rick Schlichting, Dennis B. Smith, João Pedro Sousa, Ladan Tahvildari, Kenny Wong, and Jochen Wuttke. 2011. Software engineering for self-adaptive systems: A second research roadmap. In *SE for Self Adaptive Systems*. Schloss Dagstuhl.
[36] Emmanuel Letier, David Stefan, and Earl T. Barr. 2014. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of ICSE* (Hyderabad, India) *(ICSE 2014)*. 12 pages.
[37] Minwen Ji, Alistair Veitch, and John Wilkes. 2003. Seneca: Remote mirroring done write. In *USENIX 2003*.
[38] Gabriel A. Moreno, Javier Camara, David Garlan, and Bradley Schmerl. 2016. Efficient decision-making under uncertainty for proactive self-adaptation. In *IEEE ICAC*. 147–156.
[39] Manish Parashara and Salim Hariri. 2005. Autonomic computing: An overview. *UPP 2004* (2005.).
[40] Peng Xin, Chen Bihuan, Yu Yijun, and Zhao Wenyun. 2010. Self-tuning of software systems through goal-based Feedback loop control. In *RE Conference*.
[41] Luis H. Garcia-Paucar, Nelly Bencomo, and Kevin Kam Fung Yuen. 2017. Juggling preferences in a world of uncertainty. *RE NEXT 2017*, Lisbon, Portugal (2017).
[42] Luis H. Garcia-Paucar and Nelly Bencomo. 2019. Knowledge base K models to support trade-offs for self-adaptation using Markov processes. *SASO* (2019).
[43] Luis H. Garcia-Paucar, Nelly Bencomo, and Alistair Sutcliffe. 2021. Towards technology acceptance: A Bayesian network of soft requirements, The case of the NHS COVID-19 test and trace app. *AIRE'21* (2021).
[44] Luis H. Garcia-Paucar, Nelly Bencomo, Alistair Sutcliffe, and Pete Sawyer. 2022. A Bayesian network-based model to understand the role of soft requirements in technology acceptance: The case of the NHS COVID-19 test and trace app in England and Wales. In *SAC 2022* (2022).
[45] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation. In *5th ACM/SPEC International Conference on Performance Engineering*. 3–14.
[46] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. 2008. Online planning algorithms for POMDPs. *Journal of AI Research* 32, 1 (2008), 663–704.
[47] Diederik Marijn Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of AI Research* 48 (2013), 67–113.
[48] Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence - a Modern Approach: The Intelligent Agent Book*.
[49] Trey Smith and Reid Simmons. 2004.. Heuristic search value iteration for POMDPs. In *Conference on Uncertainty in AI*, (2004), 520–527.
[50] Matthijs T. J. Spaan and Nikos Vlassis. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of AI Research* 24 (2005).
[51] João Pedro Sousa, Rajesh Krishna Balan, Vahe Poladian, David Garlan, and Mahadev Satyanarayanan. 2008. User guidance of resource-adaptive systems. In *Conference on Software and Data Technologies*.
[52] David Silver and Joel Veness. 2010. Monte-Carlo planning in large POMDPs. In *Advances in NIPS*.
[53] Pete Sawyer, Nelly Bencomo, Jon Whittle, Emmanuel Letier, and Anthony Finkelstein. 2010. Requirements-aware systems: A research agenda for RE for self-adaptive systems. In *RE*.
[54] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. 2013. DESPOT: Online POMDP planning with regularization. *Advances in NIPS* (2013).
[55] Hui Song, Stephen Barrett, Aidan Clarke, and Siobhán Clarke. 2013. Self-adaptation with end-user preference: Using run-time models and constraint solving. In *MODELS*.
[56] Guy Shani, Joelle Pineau, and Robert Kaplow. 2013. A survey of point-based POMDP solvers. *AAMAS* 27, 1 (2013), 1–51.
[57] Ahmed Abdo Ali Saeed and Seok-Won Lee. 2018. Non-functional requirements trade-off in self-adaptive systems. In *RESACS 2018*. IEEE, 9–15.
[58] Alistair Sutcliffe, Pete Sawyer, Gemma Stringer, Samuel Couth, Laura J. E. Brown, Ann Gledson, Christopher Bull, Paul Rayson, John Keane, Xiao-jun Zeng, and Iracema Leroi. 2020. Known and unknown requirements in healthcare. *Requirements Engineering* 25, 1 (2020), 1–20.

[59] Huma Samin, Luis H. Garcia-Paucar, Nelly Bencomo, Cesar M. Carranza Hurtado, and Erik M. Fredericks. 2021. RDM-Sim: An exemplar for evaluation and comparison of decision-making techniques for self-adaptation. In *SEAMS 2021*.
[60] Huma Samin, Luis H. Garcia-Paucar, Nelly Bencomo, and Peter Sawyer. 2021. Towards priority-awareness in autonomous intelligent systems. In *ACM SAC Conference* (2021).
[61] Huma Samin, Nelly Bencomo, and Peter Sawyer. 2022. Decision-making under uncertainty: Be aware of your priorities. *SoSyM* (2022), 1–30.
[62] Luis Torgo and Rita Ribeiro. 2009. Precision and recall for regression. In *International Conference on Discovery Science*. Springer, 332–346.
[63] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers (2000).
[64] Erik M. Fredericks. 2015. *Mitigating Uncertainty at Design Time and run time to address assurance for dynamically adaptive systems*. Michigan State University. PhD Thesis. (2015).
[65] Luis Garcia Paucar. 2022. RESTORM Data Set. https://github.com/researchre/RESTORM.git
[66] Luis Garcia-Paucar and Nelly Bencomo. 2018. Re-STORM: Runtime non-functional requirements trade-off supported by partially observable Markov decision processes. *SEAMS* (2018).
[67] Holger Giese, Nelly Bencomo, Liliana Pasquale, and Andres J. Ramirez. 2014. Living with uncertainty in the age of runtime models. In *Models@run.time*. LNCS, Vol. 8378. Springer.
[68] Martin Glinz. 2007. On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE'07)*. IEEE, 21–26.
[69] E. J. Horvitz, J. S. Breese, and M. Henrion. 1998. Decision theory in expert systems and artificial intelligence. *Int. Journal of Approximate Reasoning* 2, (1998), 247–302.
[70] J. O. Kephart and D. M. Chess. 2003. The vision of autonomic computing. *Computer* 36, 1 (Jan 2003), 41–50.
[71] Daphne Koller and N. Friendman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.
[72] Hanna Kurniawati. 2022. Partially observable Markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems* 5, 1 (2022), 253–277.
[73] H. Kurniawati and V. Yadav. 2013. An online POMDP solver for uncertainty planning in dynamic environment. In *International Symposium on Robotics Research* (2013).
[74] Judeal Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann (1988).
[75] Pascal Poupart. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. University of Toronto Toronto, Canada.
[76] Per Runeson and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering. SP 131, vol 14* (2008).
[77] Erwin Walraven. 2017. SolvePOMDP. https://www.erwinwalraven.nl/solvepomdp/
[78] Danny Weyns. 2019. Software engineering of self-adaptive systems. *Handbook of Software Engineering* (2019), 399–443.

Received 23 March 2022; revised 15 August 2023; accepted 22 November 2023