SPECTRA: A Markovian Framework for Managing NFR Tradeoffs in Systems with Mixed Observability

HARGYO T.N. IGNATIUS*, University of Birmingham, UK and Universitas Multimedia Nusantara, Indonesia
HUMA SAMIN, University of Exeter, UK
RAMI BAHSOON, University of Birmingham, UK
NELLY BENCOMO, Durham University, UK

Non-Functional Requirements (NFRs) play a critical role in driving self-adaptation in software systems. In Self-Adaptive Systems (SAS), satisfying multiple NFRs simultaneously introduces significant complexity, as these requirements often conflict—improving one NFR can negatively impact others. Addressing such trade-offs becomes even more challenging due to the varying degrees of observability of NFRs, with some being fully observable and others only partially observable. Traditional approaches to SAS decision-making, such as those based on Markov Decision Processes (MDPs), often assume homogeneous observability, which limits their ability to address these challenges effectively.

We argue that treating NFRs as having mixed observability—where some are fully observable and others are partially observable—enables more effective decision-making. How can self-adaptive systems model and resolve trade-offs among NFRs with mixed observability to achieve better outcomes?

This paper introduces SPECTRA, a multi-objective decision framework based on MDPs. SPECTRA addresses trade-offs among NFRs by leveraging a multi-objective Mixed Observability Markov Decision Process (MOMDP), which models and handles the varying observability of NFRs effectively. The approach is evaluated using scenarios from MirrorNet, a realistic Remote Data Mirroring (RDM) system utilizing Software-Defined Networking (SDN). Results show that SPECTRA achieves higher utility values, faster policy planning, and more effective trade-offs compared to existing approaches.

CCS Concepts: • **Software and its engineering**; • **Computing methodologies** → *Control methods*;

Additional Key Words and Phrases: vector reward, mixed observability, Markov decision process, priorities, non-functional requirements, self-adaptive systems

ACM Reference Format:
Hargyo T.N. Ignatius, Huma Samin, Rami Bahsoon, and Nelly Bencomo. 2023. SPECTRA: A Markovian Framework for Managing NFR Tradeoffs in Systems with Mixed Observability. 1, 1 (May 2023), 34 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Self-adaptive systems (SAS) must adapt to satisfy both functional and non-functional requirements (NFRs) in dynamically changing environments, where trade-offs are often unavoidable [50]. For instance, encrypting data streams may ensure

*Corresponding author

Authors' addresses: Hargyo T.N. Ignatius, hargyo@umn.ac.id, University of Birmingham, UK and Universitas Multimedia Nusantara, Indonesia; Huma Samin, h.samin@exeter.ac.uk, University of Exeter, UK; Rami Bahsoon, r.bahsoon@cs.bham.ac.uk, University of Birmingham, UK; Nelly Bencomo, nelly@acm.org, Durham University, UK.

confidentiality but can degrade performance, while implementing a heartbeat protocol to monitor availability may reduce bandwidth efficiency with excessive usage [10]. Managing these trade-offs in SAS is inherently complex, requiring context-specific configurations and careful balancing to accommodate dynamic environmental changes [50]. Furthermore, SAS must navigate uncertainties arising from external factors and internal system intricacies [26].

Adaptation decisions in SAS should be guided by NFR priorities [52]. Without clear prioritization, resolving conflicts between competing objectives becomes challenging. Stakeholders' preferences and measurable metrics, such as response time for performance or resource consumption for efficiency, play a crucial role in evaluating NFR satisfaction [36]. Ideally, the decision-making agent requires complete information about the environment and NFR satisfaction to make informed decisions. However, full observability is not always feasible [1], necessitating estimation from incomplete data, leading to partial observability.

We argue that many SAS applications exhibit *mixed observability*, where some NFRs are fully observable while others are only partially observable. For example, an autonomous vehicle (AV) fully observes its internal states, such as speed and GPS location, but can only partially infer the intentions of other drivers and pedestrians. Here, safety becomes a partially observable NFR as it depends on limited external data. Similarly, a smart home system can monitor usage patterns for lighting and heating but cannot fully understand occupants' emotional states or preferences, making usability a partially observable NFR. These mixed observability scenarios are exemplified in the motivating example described in Section 2, concerning a MirrorNet system [32, 53].

Existing SAS solutions often model decision-making using Markov models, such as Markov Decision Processes (MDPs), which are well-suited for stochastic environments with uncertainty [7, 23]. Classical MDPs are used when NFR satisfaction is fully observable [47], while Partially Observable MDPs (POMDPs) are employed when full observability is infeasible [25, 43, 52]. However, applying existing MDP or POMDP techniques to environments with mixed observable NFRs may lead to suboptimal outcomes or inefficiencies.

Observability plays a fundamental role in determining how an SAS perceives and interacts with its environment, directly influencing the complexity of decision-making and trade-off resolution. The challenges outlined above—balancing fully and partially observable NFRs, addressing mixed observability without unnecessary complexity, and ensuring explicit representation of NFR priorities—underscore the limitations of existing approaches. Addressing these challenges requires a novel framework that can dynamically manage trade-offs, optimize decision-making under uncertainty, and accommodate diverse observability setups.

This paper makes the following novel contributions:

Firstly, **SPECTRA, a framework for Mixed Observability**: we propose a framework, SPECTRA, to address heterogeneous setups of fully, partially, and mixed observable NFRs. SPECTRA helps designers and engineers choose suitable Markov models based on NFR observability and uncertainty.

Secondly, **a MR-MOMDP Implementation**: we implement a Multi-Reward Mixed Observable Markov Decision Process (MR-MOMDP) to handle environments with mixed observability and multiple objectives. The implementation, built as a Python extension to the POMDPy framework [12], adds support for multi-objective modeling and policy generation.

Thirdly, **a comprehensive Evaluation:** the framework is evaluated in a Remote Data Mirroring (RDM) system [32, 53]. Results demonstrate SPECTRA's ability to dynamically control trade-offs in SAS with mixed observability,

[1] Observability, from control theory, refers to the extent to which the internal state of a system can be inferred from its outputs [29].

achieving superior utility and performance. To encourage adoption, we provide the implementation and artifacts for experimentation and further research.

The remainder of this paper is organized as follows: Section 2 introduces a motivating scenario. Section 3 provides an overview of Markov models. Section 4 details the MR-MOMDP concept, and Section 5 presents the SPECTRA framework. Section 6 discusses the software implementation, while Section 7 evaluates our approach. Section 8 addresses threats to validity, and Section 9 explores related work. Finally, Section 10 concludes the paper and outlines future research directions.

2 MOTIVATING SCENARIO

Consider **MirrorNet**, a hypothetical Remote Data Mirroring (RDM) system leveraging Software-Defined Networking (SDN) [34] to enable seamless network topology changes without hardware adjustments. RDM is a reliable approach for failure management, creating an identical copy of data from a primary storage system at a remote location over a network connection. It efficiently replicates and distributes data while ensuring its integrity and freedom from corruption [32]. The primary goals of the RDM **MirrorNet** are to ensure data redundancy, enable disaster recovery, and maintain business continuity.

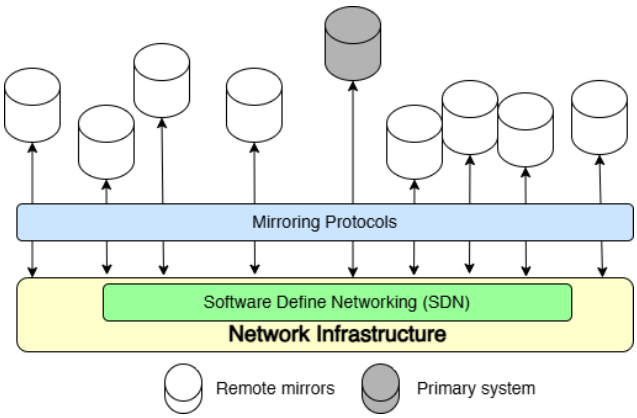
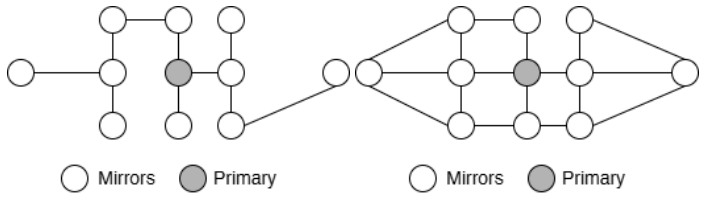


Fig. 1. MirrorNet Architecture.

As shown in Figure 1, MirrorNet comprises vital components, such as 1) the primary system, which is the primary data source that requires mirroring. This can be a storage device, a server, or even an entire data center; 2) remote systems or mirrors are locations where mirrored data is stored and are typically geographically separated from the primary system to minimize risks from regional disasters; 3) mirroring protocols that replicate data from the primary system to the remote system continuously or periodically, synchronously or asynchronously. And 4) a network infrastructure provides connections for primary and secondary systems, which can involve different types of wired and wireless networks with varying characteristics.

To establish a network of mirrors and efficiently distribute data, MirrorNet employs two dynamic topologies: Minimum Spanning Tree (MST) and Redundant Topology (RT). Figure 2 illustrates these topologies, featuring 11 nodes comprising a primary system and 10 mirrors. The MST topology (Figure 2a) minimizes the number of active links required to connect all nodes without cycles, reducing bandwidth usage and operating costs. However, MST is



(a) MST Topology (b) RT Topology

Fig. 2. Two topologies on MirrorNet

susceptible to network failure if a single link fails. In contrast, the RT topology (Figure 2b) introduces additional active links to provide alternate paths, enhancing reliability and fault tolerance. This improvement, however, comes at the cost of increased bandwidth consumption and higher operating expenses due to additional hardware and maintenance.

MirrorNet seeks to balance improved performance at lower costs (i.e., efficiency) with enhanced reliability, even in scenarios involving attacks or anomalies. Its goal is to satisfy the non-functional requirements (NFRs) of minimizing costs (MC), maximizing performance (MP), and maximizing reliability (MR), while adapting to uncertain conditions such as link failures and fluctuating bandwidth [24, 32]. To meet these NFRs, the network dynamically adapts by switching between the MST and RT topologies based on the prevailing conditions.

2.1 Mixed Observability in MirrorNet

The three NFRs that constitute MirroNet objectives have different observability. *Performance (NFR: MP)* can be fully observed based on the writing time, which represents the duration required to copy data to all mirrors, and in this aspect, both MST and RT are comparable. *Cost efficiency (NFR: MC)* can be fully observed by the total bandwidth consumption, which significantly impacts costs. On the other hand, active links and connected mirror nodes do not guarantee network *reliability (NFR: MR)* because security threats and failures can occur at all layers of RDM, involving various hardware and software components. Additionally, Intrusion Detection System (IDS) sensors have limitations in detecting and mitigating all types of attacks and failures. Nevertheless, we can partially observe reliability through probability-based metrics [46] such that higher reliability can be observed when there is a greater number of active network links and a lower severity level of attacks.

2.2 Uncertainty in MirrorNet

MirrorNet has various sources of uncertainty, some of which can be modeled using MDPs. Firstly, not all Non-Functional Requirements (NFRs) can be fully observed; only the metrics for MP and MC are completely measurable. In contrast, the satisfaction level for MR can only be estimated through a probabilistic metrics approach. This state observation uncertainties are modeled using the observation $O(s', a, o)$ detailed in Section 4 and 5.

Secondly, the switching between MST and RT topologies, or vice versa, introduces stochastic effects [24]. These effects, which influence the state representing satisfaction levels of the NFRs in MirrorNet, are shaped by inherent trade-offs that guide decision-making. While the RT topology offers superior reliability, it comes with higher costs and reduced performance. Conversely, the MST topology emphasizes improved performance and lower costs, often at the expense of system reliability. These stochastic effects are captured and analyzed using the transition function $T(s, a, s')$, as detailed in Section 4 and 5.

3 BACKGROUND

MDPs have become a popular mathematical framework for modeling and solving decision-making problems in various domains, including wireless sensor networks, robotics, and resource allocation [1]. As previously mentioned, the key advantage of MDPs is their ability to handle non-deterministic environments, where the outcome of an action is not always predictable [45]. MirrorNet entails Markov property which states that the future state of the system depends not only on the current state but also on the actions taken by the decision-maker.

In the following sections, we provide a brief overview of MDP, POMDP, and MOMDP, as well as MR-POMDP, which are the foundational concepts of this research.

3.1 MDP, POMDP, and MOMDP

The Markov Decision Process (MDP) [45] is a mathematical framework employed in decision-making, structured as a series of states, actions, and rewards.

The standard MDPs are specifically designed for scenarios where all states are fully observable. At each timestep, the agent determines its current state and selects an action, which results in a reward based on the action taken and the resulting state change. The objective of an MDP is to identify the optimal policy, which specifies the best action for each state to maximize the total expected rewards over time.

The Partially Observable Markov Decision Process (POMDP) [55] is an extension of the MDP where the state of the system cannot be fully observed or partially observable. The agent has to rely on noisy, incomplete or partial observations to infer the current state. Partial observations are caused by sensor limitations, environmental noise and disturbance, incomplete sensing range, complexity of the state, and so on. Partial observability can be either *intrinsic* attributed to the innate nature of the NFR and the problem where observability is difficult (e.g., reliability [46], emotion, satisfaction, etc.), or *extrinsic*, where the environment (e.g., sensor inaccuracy and other technical constraints) makes it difficult to fully measure NFR metrics, despite them being actually fully observable by nature (e.g., network security [46]). Unlike MDP, which assumes full observability of states, POMDP deals with partial observability using belief states. A *belief state* is a probability distribution over all possible states in the environment. It quantifies the agent's belief about the likelihood of the system being in each of those states. This makes POMDP more challenging because it has to find an optimal policy with an uncertain actual state of the system, relying on partial observations.

The Mixed Observable Markov Decision Process (MOMDP) [41] framework generalizes POMDP by employing a hybrid state representation that combines fully observable and partially observable states. Unlike POMDP, where all states are partially observable, MOMDP distinguishes states that can be directly observed with certainty from states that can only be inferred through partial observation. This state partitioning benefits the agent by taking advantage of the certainty of fully observable states while focusing computational resources on handling the uncertainty associated with partially observable states.

The multi-reward (MR) version of MDP, POMDP, and MOMDP employs vector rewards instead of scalar rewards. The vector reward dimensions represent the number of objectives (i.e., NFRs to satisfy). Fig. 3 illustrates MR-MDP, MR-POMDP, and MR-POMDP with vector rewards for a three-objectives problem. MR-MDPs operate in fully observable settings, MR-POMDPs handle partial observability of states by incorporating an observation component into the model, and MR-MOMDPs deal with mixed observability that balances complexity by separating states into fully observable and partially observable states, all while addressing multiple reward objectives. We discuss more on MR-MOMDP in Section 4, and MR-POMDP in Section 3.2 below.

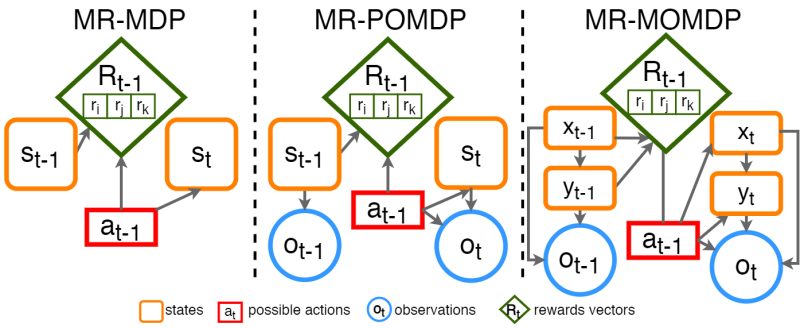


Fig. 3. Schematic diagram of MR-MDP, MR-POMDP, and MR-MOMDP model. MR-MDP is fully observable, MR-POMDP addresses partial observability by adding observation handling, and MR-MOMDP separates fully observable and partially observable states.

3.2 Multi-Reward Partially Observable Markov Decision Process (MR-POMDP)

An MR-POMDP [48, 56] model is a tuple $\{S, A, \Omega, T, O, R, \gamma\}$ where S represents the set of discrete states of the environment which are not directly observable to the agent; A denotes a finite set of all possible actions that the agent can take; Ω represents the set of discrete observations that the agent can receive; $T(s, a, s')$ is the transition function that defines state-to-state transition probabilities; $O(s', a, o)$ is observation function that generates observation outcome probabilities; $R(s, a)$ is the reward function defining the reward vector given after doing an action a in the state s, given by $R(s, a) = [r_1, \ldots, r_k]$, where k is the number of objectives; and γ is the discount factor, ranges from 0 to 1, used to prioritize immediate rewards over future rewards.

Since an agent in MR-POMDP cannot fully observe the true state, it maintains a belief state, denoted as b, a probability distribution over all possible states, representing its current knowledge and uncertainty about the environment. The belief state is updated based on the agent's actions and observations using Bayes' rule.

At each timestep, the agent selects an action a based on its belief state b to move from the current state s to the next state s'. The next state s' is given by the transition function $T(s, a, s')$ that specifies the probability distribution over the next state s' given the current state s and action a. The agent receives an immediate reward defined by reward function $R(s, a)$ for doing action a in state s. After reaching new state s', the environment generates new observations $o \in \Omega$ from the environment. The outcome of the observing o is given by $O(s', a, o) = p(o|s', a)$ for $s' \in S$ and $a \in A$. Then the agent uses these observations to update its belief state in state s' using Equation 1. The updated belief state for the next state s' is denoted as $b'(s')$.

$$b'(s') = \frac{O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)}{p(o|b, a)}, \tag{1}$$

where $p(o|b, a) = \sum_{s' \in S} O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)$ is the normalisation constant that ensures that $b'(s')$ sums to 1 over all s'.

In standard MDPs, the value function quantifies the expected cumulative reward an agent can achieve by following a particular policy starting from a given initial belief state. In contrast, MR-POMDP employs vector rewards, necessitating the use of a scalarized value function using a weight vector. An MR-POMDP policy, denoted as $\pi(b, w)$, takes the belief state $b \in B$ and weight vector w as input and returns the action $a \in A$ to be executed by the agent. The agent's goal in MR-POMDP is to find an optimal policy π^* that maximizes its scalarized value function V^* where V^* is defined in Equation 2 and satisfies the Bellman optimality equation.

$$V^*(b,w) = \max_{a \in A} \left[\sum_{s \in S} R(s,a).w.b(s) + \gamma \sum_{o \in \Omega} p(o|b,a) V^*(b',w) \right] \tag{2}$$

V^* can be approximated by iterating a number of stages where a value function V_n at stage n is parameterized by a finite set of α-matrices $\Theta_n = \{\alpha_n^i\}, i = 1, \ldots, |V_n|$ [57]. Each α-matrix $\alpha_n \in \Theta_n$ is associated with an action a. It comes with $|S| \times k$ dimension where k is the number of objectives, as in Equation 3. Each row represents a vector value for each state s, and each column corresponds to a different objective k.

$$\alpha_n = \begin{bmatrix} Obj_1 & \ldots & Obj_k \\ V(s_1)_1 & \ldots & V(s_1)_k \\ \vdots & \ddots & \vdots \\ V(s_j)_1 & \ldots & V(s_j)_k \end{bmatrix}, \tag{3}$$

where $V(s_j)_k, s \in S, k \geq 1, j = 1, \ldots, |S|$, is the value of state s_j under k-th objective.

The value function of a belief b at stage n, denoted as $V_n(b,w)$, is the inner product of belief b with the α-matrix $\alpha_n \in \Theta_n$ and weight vector w given by Equation 4:

$$V_n(b,w) = \max_{\alpha_n \in \Theta_n} b.\alpha_n.w \tag{4}$$

A point-based solver uses approximate backups to calculate the optimal α-matrix for a selected set B of sampled belief points. A point-based backup [48] is started by determining the back-projection $g_i^{a,o}$ of each next-stage value matrix $\alpha_i \in \Theta_n$ for every action a and observation o, given by Equation 5.

$$g_i^{a,o} = \sum_{s' \in S} O(a,s',o) T(s,a,s') \alpha_i(s') \tag{5}$$

Then, for each $b \in B$, the back-projection matrix $g^{a,o}$ is used to construct a new set of α-matrices, given by Equation 6 where r^a is a rewards matrix for performing action a in every state. Each new α-matrix α_{n+1}^a corresponds to an action $a \in A$, so in total, there will be $|A|$ number of new α-matrices.

$$\alpha_{n+1}^a = r^a + \gamma \sum_{o \in \Omega} \arg\max_{g^{a,o}} b.g^{a,o}.w \tag{6}$$

Finally, the new optimal α-matrix for b, denoted as $\alpha_{n+1}^{b,a}$, is the α_{n+1}^a that maximises $V_{n+1}(b,w)$ (cf. Eq. 4):

$$\alpha_{n+1}^{b,a} = \arg\max_{\alpha_{n+1}^a} b.\alpha_{n+1}^a.w \tag{7}$$

Given a belief b and weight vector w, a policy $\pi(b,w)$ at a particular stage $n+1$ is defined by choosing an action a that corresponds to the α-matrix $\alpha_{n+1}^{b,a}$.

Expanding upon the foundational concepts, in Section 4, we present our proposed approach of MR-MOMDP that deals with the mixed observability of the NFRs along with modeling of the individual priorities of NFRs to support the decision-making of SAS.

4 THE APPROACH: MULTI-REWARD MIXED OBSERVABLE MARKOV DECISION PROCESS (MR-MOMDP)

Many real-world systems contain a mix of fully and partially observable state variables. In robotics, for instance, a robot might have complete information about its battery level (fully observable) but only partial knowledge of its surrounding obstacles (partially observable). Similarly, not all NFR satisfaction levels in various SAS domains are fully observable.

Solving mixed observability problems with POMDP is not efficient due to its inability to exploit the structure inherent in mixed observability. POMDP treats all variables as partially observable, requiring the computation of a belief state (i.e., a probability distribution) over the entire state space. In contrast, MOMDP differentiates between fully and partially observable state variables. In MOMDP, the belief state is only maintained for the partially observable variables, which significantly reduces its dimensionality and simplifies computational complexity [41]. For problems with mixed observability, POMDP solvers perform redundant computations, which can increase time and memory requirements.

POMDPs often become computationally intractable for large-scale problems involving numerous state variables (i.e., the curse of dimensionality) [41]. MOMDPs, however, remain computationally manageable by focusing resources on the uncertain aspects of the system.

In the following sections, we introduce the MR-MOMDP model, which shares several attributes with the MR-POMDP model, and discuss the OLSAR solver, which requires modifications in its implementation to accommodate the specific characteristics of the MR-MOMDP model.

4.1 Model

As we mentioned earlier, MOMDP [41] is a type of POMDP where the environment includes states fully observed by the agent and partially observable states. Similar to MR-POMDP, our proposed MR-MOMDP utilizes a reward vector in which each axis component of the reward vector reflects a goal (e.g., NFR satisfaction level). The MR-MOMDP is defined as a tuple $\{X, Y, A, \Omega, b, T_x, T_y, O, R, \gamma\}$. The set X contains fully observable states $x \in X$, while the set Y contains partially observable states $y \in Y$. A complete system state is represented by a tuple (x, y), so the state space is factored as $S = X \times Y$. To transition from a start state (x, y) to an end state (x', y'), the agent takes an action a, and the value of x' is determined by the transition function $T_x(x, y, a, x') = p(x'|x, y, a)$, while the value of y' is determined by the transition function $T_y(x, y, a, x', y') = p(y'|x, y, a, x')$. The remaining components of the MR-MOMDP are the same as those in the MR-POMDP, including the set of observations Ω, the observation function O, the vector reward function R, and γ is the discount factor with a real value $\in [0, 1]$. The similarities and differences between the components of the MR-MOMDP and MR-POMDP models are illustrated in Figure 3.

By decomposing the state space into fully and partially observable states, expressing the belief space B in a factorized manner becomes feasible, reducing its high dimensionality [41]. As a result, any belief b in MR-MOMDP regarding the complete system state $s = (x, y)$ can be represented as (x, b_y), where $b_y \in B_y$ denotes the belief concerning the partially observable state y and B_y denotes the space of all beliefs on the partially observable state variable y. Each B_y is associated with the fully observable state x, denoted as $B_y(x)$. Therefore, as B is a union of $B_y(x)$ for $x \in X$, B possesses $|X| \times |Y|$ dimensions, and each $B_y(x)$ has $|Y|$ dimensions only, where $|X|$ and $|Y|$ indicate the number of fully observable and partially observable states in X and Y, respectively.

Unlike in an MR-POMDP, each fully observable state x in MR-MOMDP maintains its α-matrices set, denoted as $\Theta_y(x)$, defined over $B_y(x)$. This approach divides the belief space into smaller subspaces that are easier to manage and analyze, making it more efficient than representing the whole belief space as one complex entity [41].

As mentioned earlier in Section 3.2, an MR-POMDP policy is represented as a value function $V_n(b, w) = \max_{\alpha_n^i \in \Theta_n} b.\alpha_n^i.w$. Since the belief in MR-MOMDP is represented by the pair (x, b_y), the value function of MR-MOMDP, denoted as $V((x, b_y), w)$, is determined by a collection of α-matrices sets $\Theta = \{\Theta_y(x) | x \in X\}$, where for each x, $\Theta_y(x)$ is an α-matrices set, defined over $B_y(x)$.

$$V((x, b_y), w) = \max_{\alpha \in \Theta_y(x)} b_y.\alpha.w \tag{8}$$

To compute the value of $V((x, b_y), w)$ in equation 8, we initially utilize the value of x as an index to locate the corresponding α-matrices set $\Theta_y(x)$ and subsequently determine the maximum α-matrix α within that set using equation 9.

$$\alpha = \arg \max_{\alpha \in \Theta_y(x)} b_y.\alpha.w \tag{9}$$

4.2 Solver

A solver is an algorithm or software designed to find solutions for MDP problems, including MR-MOMDP and MR-POMDP. Its primary objective is to compute an optimal policy that maximizes the expected long-term reward while addressing uncertainties in state transitions and observations defined in the model.

We employ Optimistic Linear Support with Alpha Reuse (OLSAR) [48], a point-based solver used in MR-POMDP [48, 52], and modify it to fit the MR-MOMDP. OLSAR is a method to solve multi-objective decision problems using a series of calls of a bounded approximate single-objective POMDP solver (e.g., *OCPerseus* [48]) that leverages previously discovered α-matrices to create an initial lower bound for subsequent OCPerseus calls, enabling faster resolution of the scalarized POMDP.

The OLSAR algorithm for MR-MOMDP is largely similar to its original version for MR-POMDP. However, in MR-MOMDP, the state is divided into a fully observable component, denoted as x, and a partially observable component, denoted as y. This changes the representation of the belief state to a pair (x, b_y) consisting of the fully observable state, x, and the belief over the partially observable state, b_y. It also modifies the structure of the alpha-matrix set, transitioning from a single set of alpha-matrices, denoted as $\Theta = \{\alpha\}$, to a collection of sets, denoted as $\Theta = \{\Theta_y(x) | \Theta_y(x) = \{\alpha\}, x \in X\}$. As a result, the implementation differs, particularly in how data structures are accessed for operations involving belief states, alpha-matrices, and alpha-matrix sets, reflecting the added complexity in the implementation of OLSAR for MR-MOMDP compared to OLSAR for MR-POMDP.

Our MR-MOMDP solver consists of three layers. The term "layer" refers to the hierarchical structure of functions. The top layer is OLSAR, an outer loop responsible for calling *OCPerseus()* in the middle layer for each iteration until the convergence criteria are reached. The bottom layer is a multi-objective backup function (i.e., *backupMO()*) that implements back-projection within *OCPerseus()*.

4.2.1 *Layer 1. OLSAR for MR-MOMDP.*

OLSAR, as presented in Algorithm 1, takes initial belief b_0 as a tuple of (x_0, b_{y0}), a convergence threshold η, and maximum allowed error ϵ to produce a set of partial approximate convex coverage set (CCS) of multi-objective value

Algorithm 1: OLSAR(b_0, η, ϵ) for MR-MOMDP, adapted from [48].

Input : Initial belief $b_0 = (x_0, b_{y0})$, Convergence threshold η, Max. allowed error ϵ
Output : A set of partial approximate CCS of multi-objective value vectors C_v and its associate weights set C_w

$C_v \leftarrow \emptyset$ // A set of partial approximate CCS of multi-objective value vectors V_{b_0}
$C_w \leftarrow \emptyset$ // A set of weights associated with every value in C_v
$Q \leftarrow \emptyset$ // A queue of weights to search with their priority $\{(w, \Delta_w)\}$
Add extrema weights to Q with infinite priority
Initialize Θ_{all} // A set of α-matrices that create a minimum estimate of the value.
Initialize B // A set of sampled belief points obtained by random exploration
while $\neg Q.isEmpty() \wedge \neg timeOut()$ **do**
  $w \leftarrow Q.dequeue()$ // retrieve a weight vector
  $\Theta_r \leftarrow$ select best α-matrix $\alpha from \Theta_{all}$ for each $(x, b_y) \in B$, given w
  $\Theta_w \leftarrow$ **OCPerseus**(Θ_r, B, w, η)
  $V_{b_0} \leftarrow \max_{\alpha \in \Theta_w} b_{y0}.\alpha.w$
  $\Theta_{all} \leftarrow \Theta_{all} \cup \Theta_w$
  if $V_{b_0} \notin C_v$ **then**
    $W_{del} \leftarrow$ delete the corner weights that V_{b_0} made obsolete from Q.
    $C_v \leftarrow$ remove all values made obsolete by V_{b_0} from C_v
    $C_v \leftarrow C_v \cup V_{b_0}$
    $C_w \leftarrow C_w \cup w$
    $W_{del} \leftarrow W_{del} \cup w$
    $W_{corner} \leftarrow$ newCornerWeights(V_{b_0}, W_{del}, C_v)
    foreach $w \in W_{corner}$ **do**
      $\Delta_w \leftarrow$ calculate max. possible improvement (i.e., priority) using (w, C_v)
      if $\Delta_w > \epsilon$ **then**
        Q.add($\{(w, \Delta_w)\}$)
      end
    end
  end
end
return C_v, C_w

vectors, denoted as C_v, and its associate weights set C_w. It begins by initializing a priority queue Q and assigning extrema weights to Q with infinite priority. It also initializes a set of α-metrices Θ_{all} that define a minimum of the value and a set of sampled belief points B that can be obtained through random exploration. $\Theta_{all} = \{\Theta_y(x) | \Theta_y(x) = \{\alpha\}, x \in X\}$ consists of $|X|$ subsets where each subset corresponds to a fully observable state x.

While Q is not empty and a timeout threshold has not been reached, OLSAR does the following: it retrieves the weight vector w from Q and given w, it selects the best α-matrix α from Θ_{all} for each $(x, b_y) \in B$ and stores it into $\Theta_r = \{\Theta_y(x) | \Theta_y(x) = \{\alpha\}, x \in X\}$. It then calls $OCPerseus(\Theta_r, B, w, \eta)$ that solves MR-POMDP through scalarization using w to obtain new α-matrices set Θ_w. OLSAR then update Θ_{all} with the new Θ_w. Given initial belief $b_0 = (x_0, b_{y0})$ and Θ_w, a scalarized value of b_0, denoted as V_{b_0}, can be calculated. If V_{b_0} is not in the CCS C_v, it removes all values made obsolete by V_{b_0} from C_v, updates C_v with V_{b_0}, and stores it associated weight vector w into C_w. Then, OLSAR deletes the obsoleted corner weights from Q and stores them into W_{del}. Given V_{b_0}, W_{del}, and C_v, OLSAR calculates new corner weights and maximum possible improvements (i.e., priority) Δ_w. For each weight w in new corner weights, if Δ_w is bigger than max. allowed error ϵ, OLSAR add w and its priority Δ_w in the queue Q. These steps are repeated until Q is empty.

4.2.2 *Layer 2. OCPerseus.*

Perseus is a point-based solver, originally for POMDP, that approximates the exact solution using a set of randomized points representing the belief state. This approach allows Perseus to focus on several points of the belief space rather than dealing with a continuous representation of the entire space.

OCPerseus, described in Algorithm 2, takes a set of α-matrices Θ, a set of sampled belief points B, weight vector w, and a convergence threshold η as input to produce a new set of α-matrices Θ'. As explained in the last paragraph of Section 4.1, both Θ and Θ' consist of $|X|$ subsets of α-matrices where each subset corresponds to a fully observable state variable x.

OCPerseus performs several backup stages until it reaches the convergence threshold η. The backup stage begins by setting Θ' to an empty set and initializing B' by copying B to B'. B' is used to keep track of non-improved belief points consisting $b = (x, b_y) \in B$ whose $V_{n+1}(b, w)$ is still lower than $V_n(b, w)$ (line 11). As long as B' is not empty, *OCPerseus* randomly selects a belief point (x, b_y) from B' (line 8). Given b, w, and Θ, if possible, an improving α-matrix α is selected and added it into Θ. Then Θ' is updated with the best α-matrix from Θ. Next, all the improved belief points are deleted from B'. These steps are repeated until B' is empty.

Algorithm 2: OCPerseus(Θ, B, w, η) adapted from [48].

Input : A set of α-matrices Θ, A set of belief points $B = \{b | b = (x, b_y)\}$, weight vector w, Convergence threshold η
Output : A new set of α-matrices Θ'

1 $\Theta' \leftarrow \Theta$
2 $\Theta \leftarrow \{-\vec{\infty}\}$
3 **while** $\max_{b \in B} \left(\max_{\alpha' \in \Theta'} b.\alpha'.w - \max_{\alpha \in \Theta} b.\alpha.w \right) > \eta$ **do**
4 $\Theta \leftarrow \Theta'$
5 $\Theta' \leftarrow \emptyset$
6 $B' \leftarrow B$
7 **while** $B' \neq \emptyset$ **do**
8 $b \leftarrow$ randomly select a belief point $b = (x, b_y)$ from B'
9 $\alpha \leftarrow backupMO(\Theta, b, w)$ `// select α-matrix`
10 $\Theta' \leftarrow \Theta' \cup \{\arg \max_{\alpha' \in (\Theta \cup \alpha)} b.\alpha'.w\}$
11 $B' \leftarrow \{b | b \in B', \max_{\alpha' \in \Theta'} b.\alpha'.w < \max_{\alpha \in \Theta} b.\alpha.w\}$
12 **end**
13 **end**
14 **return** Θ'

4.2.3 *Layer 3. backupMO for MR-MOMDP.*

Since, in MR-MOMDP, each belief point $b \in B$ is a tuple (x, b_y), each observable state variable $x \in X$ maintains the back-projection matrix $g_{i,n}^{x,a,o}$ of next-stage value matrix $\alpha_n^i \in \Theta_y(x)_n$, formulated in equation 10.

$$g_{i,n}^{x,a,o} = \sum_{x' \in X} \sum_{y' \in Y} O(a, x', y', o) T_x(x, y, a, x') T_y(x, y, a, y') \alpha_i(y') \tag{10}$$

For every $b_y \in B_y(x)$, the back-projection matrix $g_{i,n}^{x,a,o}$ is utilized to create a fresh collection of α-matrices, given in Equation 11, for each $x \in X$ and for each action $a \in A$. Consequently, the total number of sets of α-matrices will equal

$|A| \times |X|$.

$$\alpha_{n+1}^{x,a} = r^{x,a} + \gamma \sum_{o \in \Omega} \arg\max_{g^{x,a,o}} b_y . g^{x,a,o} . w \tag{11}$$

The new optimal *alpha*-matrix for (x, b_y) returned by *backupMO* is given by Equation 12.

$$backupMO(\Theta_y, (x, b_y), w) = \arg\max_{\alpha_{n+1}^{x,a}} b_y . \alpha_{n+1}^{x,a} . w \tag{12}$$

5 THE SPECTRA FRAMEWORK

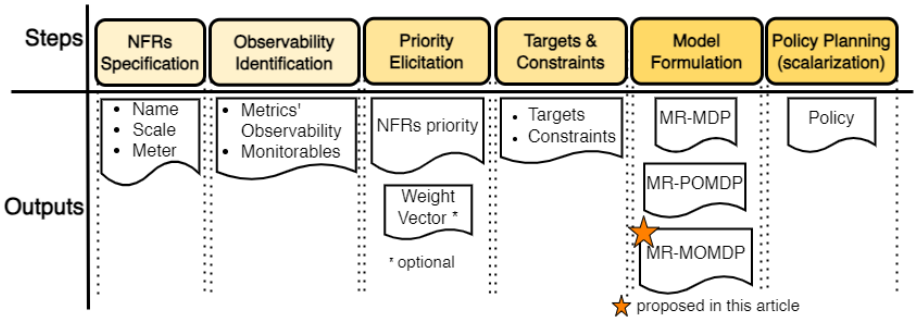


Fig. 4. The proposed SPECTRA Framework

Our proposed framework, SPECTRA, as shown in Fig. 4, aims to assist SAS in satisfying their NFRs in environments with many uncertainties. The word "SPECTRA" abbreviates specification and vector reward, which are the heart of our methodology. SPECTRA is based on Markov Decision Process models, which consider *action, state,* and *observation* uncertainty. *Action uncertainty* refers to uncertain consequences of actions; *state uncertainty* pertains to a lack of knowledge about the actual system state; and incomplete, noisy, or ambiguous observation results cause *observation uncertainty*.

This framework serves as a versatile set of steps that can be customized to suit particular domains by employing appropriate techniques. It provides a foundational structure that can be adjusted and fine-tuned according to the unique requirements and characteristics of different contexts and domains. Although the framework is presented as sequential steps/phases, it allows for flexibility in refining the design and model by revisiting previous steps or jumping further ahead. In the following sections, we describe the different phases of SPECTRA by using the hypothetical scenario of MirrorNet, described in Section 2, as an example.

5.1 Phase 1. NFRs Specification

At this stage, adopted from the SAFe framework [36], the stakeholders and designer specify NFRs by considering three attributes. Firstly, a *name* is given to each NFR, which may pertain to the system's attributes such as configurability, performance, security, and reproducibility, among others [62]. Secondly, they determine the *scale*, which identifies the object being measured and the corresponding unit of measurement. Finally, they establish the *meter*, which is the measurement method. These details are outlined in Table 1, providing an overview of the NFRs of MirrorNet and their corresponding attributes in phase 1. This approach ensures that the NFRs are well-defined and accurately measured.

Table 1. Phase 1 - NFRs Definition

Name	Scale	Meter
Minimise Cost **(MC)**	Bandwidth consumption	Average bandwidth consumption
Maximise Performance **(MP)**	Writing time	Average writing time
Maximise Reliability **(MR)**	Network reliability	Average number of healthy links

Table 2. Phase 2 - Observability Identification

NFR	Metrics	Observability	Monitorables
MC	Average bandwidth consumption	Direct	Bandwidth consumption **(BWC)**
MP	Average writing time	Direct	Writing time **(WRT)**
MR	Average number of healthy links	Indirect	Active network links **(ANL)**, Network attack and anomaly **(ATK)**

5.2 Phase 2. Observability Identification

During this stage, stakeholders and designers assess the *observability* of each metric associated with the NFRs. Observability is categorized as either direct or indirect. If the metric can be directly observed, the NFR is fully observable. On the other hand, if the metric can only be probed through various monitorable objects, it implies that the NFR is partially observable. In the case of direct or fully observable metrics, the metric itself serves as the *monitorable*. Conversely, different objects are used as monitorables to probe the desired metric for indirect or partially observable metrics.

In Table 2, MC and MP metrics are easily observable. Bandwidth consumption (BWC) and writing time (WRT) can be directly monitored to determine their averages. However, the number of healthy links cannot be directly observed as compromised hosts may be connected to active links and exhibit malicious behavior at the upper layer. Nonetheless, MirrorNet employs a traffic monitoring system that can detect anomalies and network attacks, enabling it to probe the number of healthy links by monitoring the active network links (ANL) and the status of network attacks and anomalies (ATK).

5.3 Phase 3. Priority Elicitation

Preference elicitation constitutes a foundational issue in creating intelligent systems that make or recommend decisions on behalf of users[4]. Priorities and preferences need to be elicited from domain experts and stakeholders, which serve as guiding factors for decision-making in adaptation, aligning with the system's objectives [64]. Various decision-making techniques (e.g., AHP [40], TOPSIS [35], POM [67], etc.) and utility theories [33, 59] can be employed in this phase.

Stakeholders may specify the tradeoff between NFRs by creating a utility function that assigns weight to each quality attribute, i.e., forming a weight vector (e.g., [0.333, 0.333, 0.333]), incorporating quality attribute scenarios, or utilizing conditional logic [10].

If a global weight vector is provided, we do not need to find the set of optimal weights in Phase 6 - Policy Planning. Policies can be generated by scalarizing and solving the model using a single-objective solver. In the case of MirrorNet, priorities are not elicited as a weight vector but rather as a conditional logic, as stated below:

> "In normal to moderately detrimental conditions, priority is given to cost efficiency and performance, with reliability being less important. However, in highly severe detrimental conditions, reliability holds the highest priority, resulting in a relaxation of cost efficiency and performance requirements."

The priorities and preferences generated in this phase determine how targets and constraints can be determined in the subsequent phase, as well as the reward function of the model.

Table 3. Phase 4. Targets in environment SL1 and SL2.

NFR	Targets
MC	Average bandwidth consumption SHALL below the threshold and AS FEW AS POSSIBLE
MP	Average writing time SHALL below the trhreshold and AS FEW AS POSSIBLE
MR	Average number of active link SHALL above the threshold and AS MANY AS POSSIBLE

Table 4. Phase 4. Constraints for normal and detrimental conditions of environment SL1.

Condition	Normal / Non-detrimental	Detrimental / Severity Level: Low-Moderate
NFR	**Constraints**	**Constraints**
MC	Total bandwidth threshold violations $\leq$ 15%	Total bandwidth threshold violations $\leq$ 15%
MP	Total writing time threshold violations $\leq$ 10%	Total writing time threshold violations $\leq$ 10%
MR	The number of active network links threshold violations $\leq$ 30%	The number of active network links threshold violations $\leq$ 35%

Table 5. Phase 4. Constraints for normal and detrimental conditions of environment SL2.

Condition	Normal / Non-detrimental	Detrimental / Severity Level: High
NFR	**Constraints**	**Constraints**
MC	Total bandwidth threshold violations $\leq$ 15%	Total bandwidth threshold violations $\leq$ 30%
MP	Total writing time threshold violations $\leq$ 10%	Total writing time threshold violations $\leq$ 10%
MR	The number of active network links threshold violations $\leq$ 30%	The number of active network links threshold violations $\leq$ 5%

5.4 Phase 4. Targets and Constraints

The next stage involves defining *target* values to achieve and *constraints* to avoid, considering the priority given in the previous stage. This step is crucial in establishing clear objectives and boundaries that determine the satisfactory level of the system.

> "If you can't measure it, you can't manage it." - Peter Drucker

To define targets and constraints, our reference values are monitorables rather than meters or metrics, as meters/metrics may not always be directly observable, while we can determine metric values through monitorables.

Additionally, we can control the values of monitorables through each action or decision we take in each state, which will impact the monitorables value.

By defining targets and constraints, stakeholders can effectively communicate their expectations and align the system with organizational goals. However, it is possible to relax the target and constraints under particular circumstances to enable SAS to keep functioning in changing and uncertain environments while still meeting critical requirements [63].

Table 3 lists the targets of MirrorNet for two different environmental scenarios: SL1 and SL2. The targets are specified using the RELAX language [63] to explicitly acknowledge and address the inherent uncertainty in SAS. The SL1 environment anticipates detrimental conditions with low to moderate severity levels, while the SL2 environment anticipates detrimental conditions with high severity levels.

Table 4 and Table 5 illustrate the constraints for the SL1 and SL2 environments in both normal and detrimental conditions. Under normal (non-detrimental) conditions, both environments share the same constraints, allowing for a higher threshold violation of active network links (ANL) to achieve better performance (MP) and cost efficiency (MC). However, in detrimental conditions, the SL1 environment relaxes its ANL threshold without sacrificing performance (MP) and efficiency (MC) because stakeholders remain confident in the reliability of the MirrorNet infrastructure under low to moderate severity levels. In contrast, the SL2 environment with high severity levels tightens its reliability requirements at the expense of cost efficiency.

5.5 Phase 5. Model Formulation

This phase is a critical part and determines the success of adaptation. Markov models rely heavily on the accuracy of their inputs. Key parameters, such as transition probabilities, reward functions, and observability functions, are crucial for determining both the model's reliability and the optimality of the resulting policy. Therefore, the Markovian framework should be applied in systems that can be accurately modeled, have well-defined states, and have reliable data, and it should be avoided in highly unpredictable or data-scarce environments.

Fig. 5 depicts MirrorNet as a control system consisting of a controller and a MirrorNet network on the plant side. To achieve the objectives, i.e., satisfice the NFRs, the controller needs to observe and analyze the plant behavior (using models and thresholds) and decide the best action to take (e.g., topology), using the available weight vector and policy, then sends the decision to the actuator. Therefore, plant behavior needs to be reflected in the model accurately.

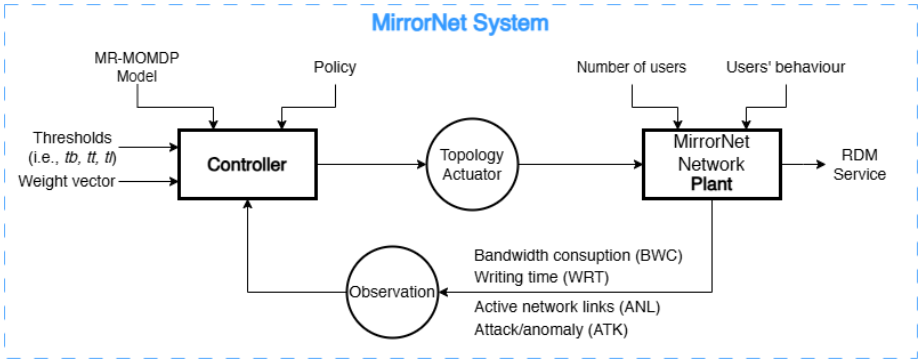


Fig. 5. The block diagram of MirrorNet as a closed loop control system.

Based on the observabilities of the NFRs, we can determine the appropriate Markovian model to utilize. If all NFRs are fully observable, using a multi-reward MDP (MR-MDP) is recommended. However, if all NFRs are only partially observable, the problem must be modeled as an MR-POMDP. In cases where the environment presents a combination of both partially and fully observable NFRs, it is advisable to model it as an MR-MOMDP.

As shown in Table 2, MirrorNet exhibits mixed observability NFRs, where MC and MP can be directly observed, but MR is only partially observed through the number of active network links (ANL) and the status of network attacks (ATK). Therefore, the suggested modeling approach is MR-MOMDP. The MirrorNet environment and NFRs can be mapped to an MR-MOMDP model shown in Fig. 6.

5.5.1 States. States at least represent combinations of satisfaction levels of the NFRs (i.e., MC, MP, and MR) but can also involve other variables that are deemed necessary to represent the states of the plant (i.e., MirrorNet network as the managed asset). In the MR-MOMDP, there are two types of states: fully observable states and partially observable states. Regarding MirrorNet, fully observable states represent combinations of satisfaction values for MC and MP, whereas partially observable states represent combinations of satisfaction values for MR. Consequently, four fully observable states $X = \{x1, x2, x3, x4\}$ and two partially observable states $Y = \{y1, y2\}$ have been identified and presented in Table 6.

5.5.2 Actions. An action represents the agent's decision or choice at each state. In the case of MirrorNet, two adaptive actions are considered for each state: selecting the most suitable topology between Minimum Spanning Tree (MST)

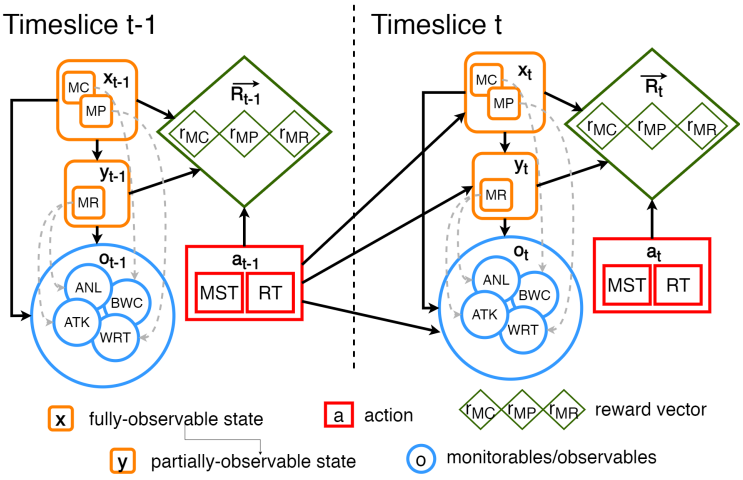


Fig. 6. MirrorNet RDM as an MR-MOMDP problem.

Table 6. Fully and partially observable states of MirrorNet.

Fully observable States (X)		
State	$NFR_1 = MC$	$NFR_2 = MP$
x1	TRUE	TRUE
x2	TRUE	FALSE
x3	FALSE	TRUE
x4	FALSE	FALSE
Partially observable States (Y)		
State	$NFR_3 = MR$	
y1	TRUE	
y2	FALSE	

or Redundant Topology (RT). Based on the current state, the agent chooses the appropriate topology to support the satisfaction of the NFRs in the RDM system.

5.5.3 Transition Functions. Any action (i.e., topology) chosen in each state will have an impact on the environment and transition the environment from the current state to a new state with some probabilities. To capture these transitions, we decompose the transition functions T_x and T_y into marginal conditional probabilities of the NFRs, specified in Table 7, as in Equation 13, 14, and 15. This information is provided by domain experts [25, 52].

$$T((x,y),a,(x',y')) = T_x(x,y,a,x')T_y(x,y,a,y') \tag{13}$$

$$T_x(x,y,a,x') = P(MC'|x,y,a)P(MP'|x,y,a) \tag{14}$$

$$T_y(x,y,a,y') = P(MR'|x,y,a) \tag{15}$$

Table 8 presents the complete transition function T_x, as defined by Equation 14 and refers to Table 7. This function specifies the transition probabilities when performing an action, either MST or RT, from the current state (x, y) to the next fully observable state $x' \in X$. For instance, executing the MST action from any current state (x, y) transitions the

Table 7. NFR satisfaction probabilities.

Action	MC_{t-1}	MP_{t-1}	MR_{t-1}	P($MC_t = T$)	P($MC_t = F$)	P($MP_t = T$)	P($MP_t = F$)	P($MR_t = T$)	P($MR_t = F$)
MST	T or F	T or F	T or F	0.9002	0.0998	0.9697	0.0303	0.3725	0.6275
RT	T or F	T or F	T or F	0.7045	0.2955	0.8788	0.1212	0.7222	0.2778

Table 8. Transition probabilities to fully observable states (Tx)

Action	x	y	$x' = x1$	$x' = x2$	$x' = x3$	$x' = x4$
MST	$\forall x \in X$	$\forall y \in Y$	0.8729	0.0273	0.0968	0.0030
RT	$\forall x \in X$	$\forall y \in Y$	0.6191	0.0854	0.2596	0.0358

x = Fully observable state, y = Partially observable state, see Table 6

Table 9. Transition probabilities to partially observable states (Ty)

Action	x	y	$y' = y1$	$y' = y2$
MST	$\forall x \in X$	$\forall y \in Y$	0.3725	0.6275
RT	$\forall x \in X$	$\forall y \in Y$	0.7222	0.2778

system to the next state $x' = x1$ with a probability of 0.8729, to the next state $x' = x2$ with a probability of 0.0273, and so on.

On the other hand, Table 9 describes transition function T_y, as defined by Equation 15 and also refers to Table 7. This function outlines the transitions from the current state (x, y) to the next partially observable state $y' \in Y$, based on the selected topology in MirrorNet. For example, performing the MST action from any current state (x, y) transitions the system to state $y' = y1$ with a probability of 0.3725 and to state $y' = y2$ with a probability of 0.6275.

The overall transition function, which defines the movement from the current state (x, y) to the next state (x', y'), is represented as a joint probability of T_x and T_y, as given in Equation 13.

In our transition probabilities, we utilize four decimal places. While longer decimal representations can provide more precise probabilities, the appropriate number of decimal places ultimately depends on the context and the judgment of the designers and domain experts. For many real-world applications, a few decimal places are often sufficient. However, in highly sensitive or critical systems, additional decimal places may be necessary to ensure the required level of accuracy. Although four decimal places may not offer significantly more meaningful information than two in our specific scenario, it does enhance accuracy without substantially increasing the computational load of the system.

5.5.4 Rewards. In the decision-making process of SAS, it is crucial to consider the satisfaction priorities of individual NFRs. SPECTRA uses vector rewards to model these priorities and considers higher reward value as higher priority/desirability when making adaptation decisions. The reward value may also represent the utility of a particular objective-action pair.

A vector reward gives different values to each objective, as opposed to a scalar reward, which gives a single value to each action. This approach provides a more precise understanding of how a particular action can impact multiple NFRs. By simultaneously evaluating several factors, a vector reward helps to make more informed decisions by capturing the relative importance and trade-offs between objectives. It enables decision-makers to optimize choices based on multiple competing objectives and more accurately represents the decision-making process. The use of reward vectors adds complexity but offers transparency to capture the different impacts of adaptive decisions on individual NFRs in terms of their satisfaction.

Table 10. Reward vector values for NFRs in MirrorNet.

State	Action	Reward vector values		
		r_{MC}	r_{MP}	r_{MR}
x1:y1	MST	110	95	95
x1:y2	MST	60	60	10
x2:y1	MST	95	40	80
x2:y2	MST	65	10	10
x3:y1	MST	40	80	80
x3:y2	MST	10	60	10
x3:y1	MST	35	35	65
x3:y2	MST	3	3	3
x1:y1	RT	80	95	90
x1:y2	RT	85	83	20
x2:y1	RT	60	10	60
x2:y2	RT	80	18	20
x3:y1	RT	10	66	68
x3:y2	RT	40	75	40
x3:y1	RT	11	10	26
x3:y2	RT	10	10	10

The vector reward values are acquired from domain experts and should accurately reflect the knowledge and information available to them [52]. Since every possible action in each state must have a defined reward, when the state space is large, designers need more effort to determine the reward. The provision of vector rewards in a consistent and precise manner is challenging, especially in a large state space, but it is still possible to do so with caution in an adequate timeframe.

MirrorNet aims to satisfice three NFRs, and to achieve this, each state is linked to a three-dimensional reward vector function specified as $R(s, a) = [r_{MC}, r_{MP}, r_{MR}]$. Each element of the reward vector indicates the priority of the corresponding NFR in each state compared to the other NFRs along their respective axes. The values of the reward vector are based on the elicited priorities obtained during Phase 3 - Priority Elicitation and might follow utility theories [33]. During this stage, we establish local weights as state-specific reward vectors, while the global weight vector, generated by OLSAR-Perseus in Phase 6 or determined beforehand in Phase 3, reflects the stakeholders' overall preferences.

The complete reward values are presented in Table 10, serving as the reward function in our model. Each entry of Table 10 indicates the preference over the execution of an action according to the value of NFRs in each state. We can observe that MST is preferable when all NFRs are satisfied or when MC is unsatisfied, whereas RT is preferable when MR is most likely unsatisfied. This reward function captures the cumulative impact of the NFRs on the overall reward in MirrorNet.

5.5.5 Observations. Observations represent combinations of monitorable values used to determine the current state. For fully observable states, the monitorables directly determine the state. However, for partially observable states, the monitorables are used to probe the state value based on probabilistic values obtained from observations validated by domain experts.

MirrorNet incorporates four monitorables (i.e., BWC, WRT, ANL, and ATK) associated with each NFR, as outlined in Phase 2 of the MR-MOMDP and listed in Table 2. The observation probabilities for these monitorables in MirrorNet are presented in Table 11. BWC, WRT, and ANL can take on two possible values: violating or not violating the threshold, while ATK indicates the boolean status of anomalies or attacks in the network. Threshold values for BWC, WRT, and

Table 11. Observation probabilities of MirrorNet.

MON1 Bandwidth consumption (BWC)				
	P(MC=T|BWC)		P(MC=F|BWC)	
Action	BWC $\leq tb$	BWC $> tb$	BWC $\leq tb$	BWC $> tb$
MST	1	0	0	1
RT	1	0	0	1
MON2 Writing time (WRT)				
	P(MP=T|WRT)		P(MP=F|WRT)	
Action	WRT $\leq tt$	WRT $> tt$	WRT $\leq tt$	WRT $> tt$
MST	1	0	0	1
RT	1	0	0	1
MON3 Active network links (ANL)				
	P(MR=T|ANL)		P(MR=F|ANL)	
Action	ANL $< tl$	ANL $\geq tl$	ANL $< tl$	ANL $\geq tl$
MST	0.3	0.7	0.7	0.3
RT	0.15	0.85	0.85	0.15

MON4 Anomaly/attack status (ATK)								
	P(MR=T|ATK)				P(MR=F|ATK)			
Env:	SL1		SL2		SL1		SL2	
Action	ATK=True	ATK=False	ATK=True	ATK=False	ATK=True	ATK=False	ATK=True	ATK=False
MST	0.4	0.6	0.15	0.85	0.6	0.4	0.85	0.15
RT	0.4	0.6	0.15	0.85	0.6	0.4	0.85	0.15

Table 12. Selected optimal weight vectors.

Framework	Environment	
	Weight vector for SL1	Weight vector for SL2
MR-MOMDP	[0.8441, 0. , 0.1559]	[6.7026e-01, 3.5220e-06, 3.2974e-01]
MR-POMDP	[0.8387, 0. , 0.1613]	[6.2697e-01, 1.7295e-09, 3.7303e-01]

ANL (i.e., tb, tt, tl) are given in Table 13. With the inclusion of these four monitorables and two possible values for each, our MR-MOMDP model for MirrorNet incorporates a total of sixteen observations $o \in \Omega \therefore |\Omega| = 16$.

5.6 Phase 6. Policy Planning

Given the model, our next step is to solve the problem and obtain an optimal set of policies. The policy planning can be conducted offline or online using a specific solver. If the global weight vector can be determined in Phase 3, the policy can be generated using an appropriate single-objective solver through scalarization.

This article employs OLSAR and Perseus for offline planning, assuming that the global weight vector is not determined in Phase 3. As shown in Fig. 7, OLSAR provides us with a set of optimal weight vectors and their corresponding vector value and policy. Various methods, such as knee point detection [5], can be utilized to select a values vector that aligns with the preferences. However, this study defines the minimum CCS values as the threshold to choose an optimal weight vector and its corresponding policy for each value point that exceeds the threshold. The chosen weight vector and policy are among the inputs the controller uses (i.e., managing system) as in Fig. 5. Table 12 shows the optimal weight vectors chosen for different frameworks and environments during our evaluation.

6 SOFTWARE IMPLEMENTATION

We implemented MR-MOMDP in a Python framework called XPOMDPy, which stands for Extended POMDPy. For the purpose of reproducibility, the framework is available online at [30]. XPOMDPy extends the current POMDPy framework [12] by providing the following additional features:

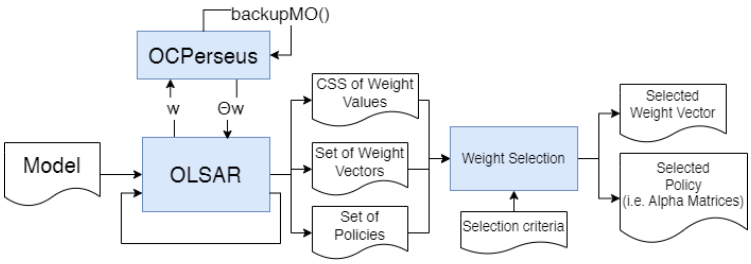


Fig. 7. Policy planning with OLSAR & OCPerseus.

(1) **Perseus solver for single-objective POMDP and MOMDP, as well as OLSAR-Perseus for multi-objective MR-POMDP and MR-MOMDP.** Our implementation of Perseus is a Python translation of Java implementation by [60]. We also use OLS implementation by [3] in our OLSAR implementation.
(2) **Model parsing capabilities from POMDP and PomdpX file formats.** This feature simplifies the process of solving and generating new model objects by easily reading and interpreting the model specifications.
(3) **Policy writer that supports the PolicyX file format for both single-objective (i.e., alpha vector) and multi-objective (i.e., alpha matrix) policies.** This enables storing and retrieving policies in a structured format, facilitating their use in decision-making processes.

7 EVALUATIONS

We evaluate the feasibility of our proposed SPECTRA and MR-MOMDP frameworks by comparing the policy planning of MirrorNet using MR-MOMDP with baseline approaches. Specifically, we aim to address the following research questions:

(RQ1): How can SPECTRA better optimize the planning process and produce better policy?
(RQ2): To what extent does SPECTRA produce better policies and effectively satisfy NFRs with tradeoffs aligning with stakeholders' preferences?

Through this evaluation, we aim to assess the effectiveness of our proposed SPECTRA framework and determine the superiority of MR-MOMDP over the baseline approaches in terms of planning time and policy quality for MirrorNet.

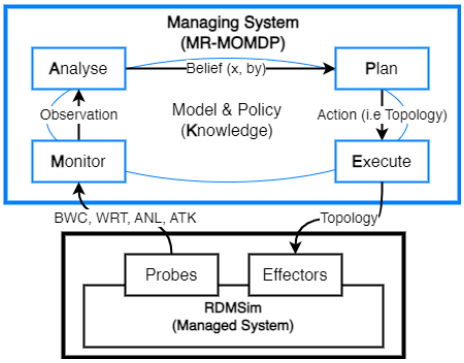
7.1 Experiment Setup

7.1.1 Environment Model and Scenario. To analyze the effectiveness of the policies, each approach is tested in two scenarios: the *Non-detrimental Scenario*, where all topologies operate normally without any disruptions, and the *Detrimental Scenario*, where system disruptions occur. To introduce variability in our experiments, we consider two environmental models representing the Low-Moderate Severity Level (SL1) and High Severity Level (SL2) in the Detrimental Scenario. These environments are modeled using different observation functions in MON4, as shown in Table 11.

7.1.2 Simulation. Our experiments were conducted using a modified Python version of RDMSim [53] - a discrete event simulator of a Remote Data Mirroring environment. As illustrated in Figure 8, we use the MAPE-K architecture, which consists of two layers: the managing system and the managed system. We incorporated RDMSim as the managed system at the bottom layer. To ensure reproducibility, we made modifications to the original RDMSim, including

the implementation of a random seed function. The modified RDMSim can be accessed at [30]. We conducted our simulations and experiments on a laptop equipped with an Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz and 8 GB RAM, running Windows 10 Pro.

Fig. 8. MAPE-K Architecture of the simulation using RDMSim and MR-MOMDP.



The managing system at the upper layer consists of a decision-making agent with MAPE-K components [49]. It employs MR-MOMDP or other baseline approaches in the Analysis & Planning component. The managing system is responsible for providing the optimal policy for the managed system. This policy can be generated offline at design time or online at runtime. In this article, we use OLSAR Perseus to generate offline policies, which are stored in the architecture's Knowledge component. Online policy planning can use solvers such as DESPOT [66], POMCP [54], etc., which are currently beyond the scope of this article.

Runtime adaptation works as follows: At each time step the managing system makes observations of the managed system (using four monitorables: BWC, WRT, ANL, and ATK). Based on the observations made by the Monitoring component, the Analysis component updates its belief state. The Planning component uses the updated belief state to select the appropriate topology based on the policy available in the Knowledge component. Given an updated belief state, weight vector, and policy (containing alpha matrices, see Figure 7), the Planning component selects the action (i.e., topology) associated with the alpha matrix that provides the most optimal value using Equation 9. The topology chosen is then activated in the managed system by the Execute component.

We configured RDMSim by adjusting the parameters in *Config.py* and *MAPEKLoop.py* to simulate MirrorNet, which consists of 25 mirrors with 300 physical links used for data transfer between these mirrors, with the settings specified in Table 13.

7.2 Evaluation Metrics

We utilize the following metrics to analyze the experimental results:

(1) Planning time: the execution time required to generate a CCS of weight vectors and a set of policies.
(2) Expected total reward: the accumulated expected reward obtained by executing an action in each step, in which the belief state determines the value for each step.
(3) Size of CCS solutions: the number of non-dominated weight vectors successfully found.
(4) Topology proportions: the proportion of selecting MST topology relative to RT topology in a simulation round.
(5) Target satisfaction: measures how closely the execution results meet the target.

Table 13. RDMSim configuration

Configuration item	Value
Random seed	0
Time steps	200
Mirrors	25
MST Active Links %	[8, 58]
RT Active Links %	[30, 65]
Single bandwidth consumption	[20, 30]
Single writing time	[10, 20]
BWC max. threshold tb	4050
WRT max. threshold tt	3000
ANL min. threshold tl	90
Discount factor γ	0.95

(6) Constraints violation: identifies whether the execution results violate the given constraints.

7.3 Baselines

7.3.1 MR-POMDP. For a fair comparison, we selected the implementation of MR-POMDP that also utilizes OLSAR-Perseus [48, 52] to investigate the effects of modeling multi-objective mixed observability problems using MR-MOMDP (a specific model for mixed observability) and MR-POMDP designed for multi-objective partial observability. In this evaluation, we employ a discount factor of $\gamma = 0.95$ to generate policies for both MR-POMDP and MR-MOMDP.

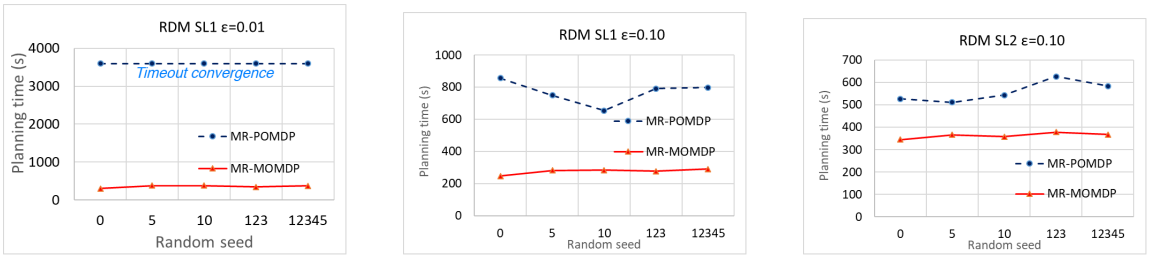
7.3.2 Rule-Based Adaptation. . RDMSim [53] is equipped with a default rule-based adaptation mechanism that selects the MST topology if there is a violation of the BWC threshold and WRT threshold, and chooses the RT topology if the ANL threshold is violated. We use this as an additional baseline to compare various adaptation strategies in satisficing NFRs.

7.4 (RQ1) Optimal Policy Planning

The SPECTRA method enhances the management of tradeoffs in uncertain situations for SAS. This is achieved through three key features: 1) considering diverse observability to determine the most suitable MDP model, 2) priority elicitation to guide policy decision-making during runtime, and 3) representing the priorities of each NFR through vector reward to facilitate well-informed decision-making.

Through SPECTRA, we can model the decision-making of MirrorNet using MR-MOMDP, which can be compared with baselines. As previously explained, we implemented MR-MOMDP with OLSAR and Perseus to formulate an optimal policy configuration for MirrorNet, using a convergence threshold of $\eta = 0.01$, a maximum allowed error of $\epsilon = 0.01$, and a convergence timeout of 3600 seconds. The convergence threshold is used to determine when the policy computation has converged to an acceptable solution. A policy is considered good enough when the improvement in the policy falls below the convergence threshold. The maximum allowed error defines how much deviation from the optimal policy is acceptable. By controlling the maximum allowed error, we can trade off precision against runtime. Convergence timeout is used to prevent excessively long computations. The designers and domain experts choose these values to generate an acceptable policy while ensuring computational feasibility and practical applicability.

We compared the planning time required by MR-MOMDP and MR-POMDP in two environments with different values of ϵ. In the default setting, as shown in Fig. 9a, with $\epsilon = 0.01$, MR-POMDP is intractable and always reaches the timeout convergence, while MR-MOMDP completes within the range of 300-372 seconds. When the maximum allowed error is relaxed to $\epsilon = 0.10$, MR-MOMDP still finishes faster than MR-POMDP, taking 247-291 seconds in the



(a) Env. model=SL1, $\epsilon = 0.01$ (b) Env. model=SL1, $\epsilon = 0.10$ (c) Env. model=SL2, $\epsilon = 0.10$

Fig. 9. Planning time of MR-POMDP and MR-MOMDP in different settings

SL1 environment (Fig. 9b) and 345-379 seconds in the SL2 environment (Fig. 9c), while MR-POMDP requires 654-856 seconds in the SL1 environment and 511-627 seconds in the SL2 environment. It can also be observed that different random seeds yield different planning times.

We observed that the backup time required by MR-MOMDP is longer than MR-POMDP due to the additional loop caused by the decomposition of states into fully and partially observable states. This decomposition increases the number of alpha-matrices and computational operations needed in MR-MOMDP. However, we suspect that the smaller belief space maintained by MR-MOMDP reduces the number of corner weights that need to be tested individually. On the other hand, MR-POMDP with a larger belief space has a higher likelihood of finding more corner weights, which leads to a longer convergence time for MR-POMDP.

Fig. 10 shows the size of CCS found by MR-POMDP and MR-MOMDP using $\epsilon = 0.01$, where MR-POMDP yields 174 points in the SL1 environment and 164 points in the SL2 environment, while MR-MOMDP only yields 18 points and 34 points, respectively. However, when the maximum allowable error is relaxed to $\epsilon = 0.10$, we can reduce the size of the CCS in MR-POMDP to approximately half, as shown in Table 14.

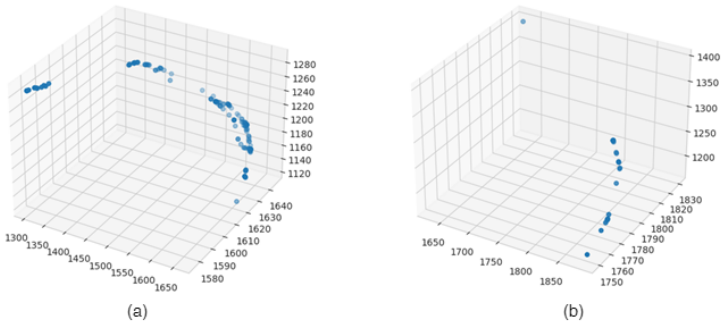


Fig. 10. Convex Coverage Set (CCS) of (a) MR-POMDP (164 points) and (b) MR-MOMDP (34 points) under SL2 context.

Table 14. CCS size of MR-MOMDP and MR-POMDP in different environments and settings

CCS Size					
Part 1. Seed=0, Epsilon=0.01					
		Env: SL1		**Env: SL2**	
MR-MOMDP		18		34	
MR-POMDP		174		164	
Part 2. Env: SL1, Epsilon=0.10					
Seed	**0**	**5**	**10**	**123**	**12345**
MR-MOMDP	8	8	10	10	10
MR-POMDP	93	87	81	89	98
Part 3. Env: SL2, Epsilon=0.10					
Seed	**0**	**5**	**10**	**123**	**12345**
MR-MOMDP	12	12	12	12	13
MR-POMDP	74	70	77	85	78

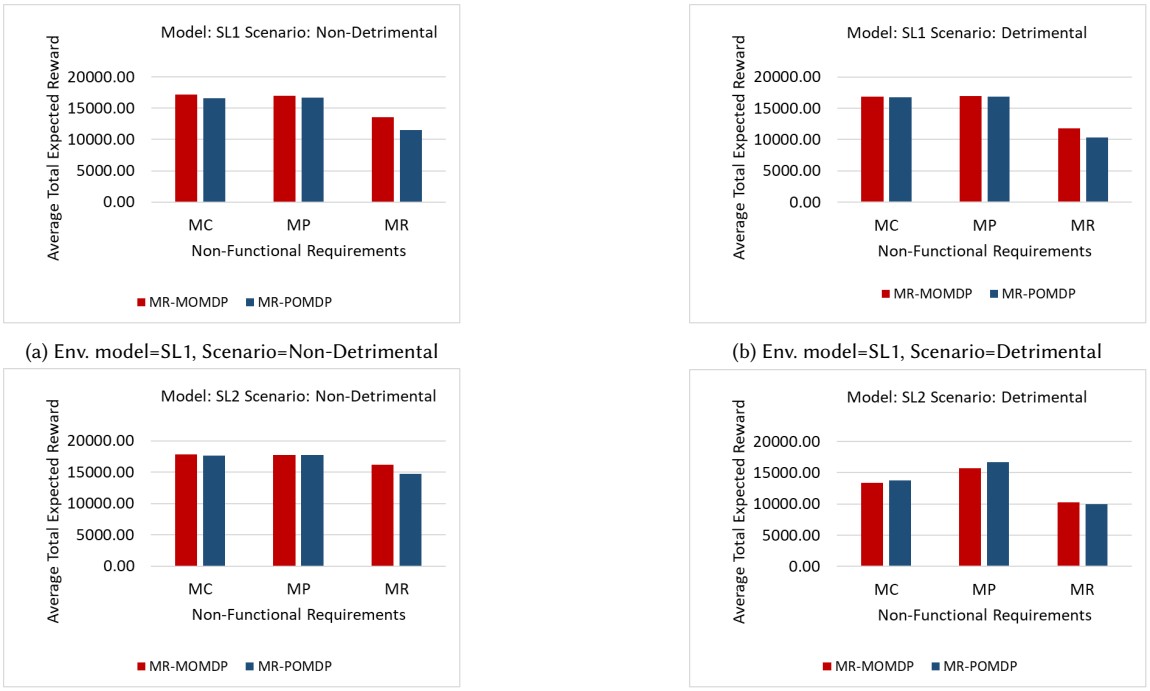
As we need to decide on a single final weight vector along with its corresponding set of policies, having a larger CCS size provides more alternatives. However, it also requires more time and effort to determine the choice, especially if the selection is made manually by a human decision-maker. Nevertheless, having a reliable autonomous component to select the best final weight vector will benefit the planning process.

7.5 (RQ2) Policy Quality and NFR Satisfaction

We answer RQ2 by evaluating the policies produced by MR-MOMDP and MR-POMDP using metrics: expected total reward, the proportion of topology used, target satisfactions, and constraint violations, which we discuss below.

7.5.1 ***Expected Total Reward.*** The expected total reward represents the utility value of a system because the agent's goal is to maximize the expected total reward through a sequence of actions while accounting for the uncertainty of the system's state. Therefore, a set of policies that yields a higher expected total reward can be considered a better set of policies.

We compared the average expected total reward of MR-MOMDP and MR-POMDP for each NFR in SL1 and SL2 environments under both non-detrimental and detrimental conditions, using different random seeds mentioned in Table 14 and Fig. 9.



(a) Env. model=SL1, Scenario=Non-Detrimental

(b) Env. model=SL1, Scenario=Detrimental

(c) Env. model=SL2, Scenario=Non-Detrimental

(d) Env. model=SL2, Scenario=Detrimental

Fig. 11. Expected total reward of MR-MOMDP and MR-POMDP in different environment models.

Fig. 11 shows that under non-detrimental conditions, all NFRs had higher average expected total rewards in MR-MOMDP and MR-POMDP compared to detrimental conditions. This outcome is not surprising since NFRs are easier to satisfice in normal conditions than during anomalies. In non-detrimental conditions, SL2 provided higher rewards than SL1, while in detrimental conditions, SL1 performed better than SL2. The reason behind this is the probability values in the observation function, where SL2 has higher probability values of P(MR=T|ATK=F) and lower values of P(MR=T|ATK=T).

Overall, MR-MOMDP yielded higher average values for all NFRs compared to MR-POMDP. However, in SL2-Detrimental, MR-POMDP outperformed MR-MOMDP in MC and MP, as MR-MOMDP sacrificed both NFRs for higher MR, which is a preferred outcome.

To ensure the reward difference was not due to the weight vector choice, we evaluated the solutions using the policy evaluator tool provided by APPL [41]. We compared the scalarized versions of MR-MOMDP and MR-POMDP with a weight vector of [0.333, 0.333, 0.333], solved using SARSOP [41]. Fig. 12 shows that the scalarized MR-POMDP (SARSOP-MOMDP) consistently yielded higher expected rewards than SARSOP-POMDP across various configurations. This is because MOMDP reduces uncertainty through a smaller belief space, resulting in higher belief values that impact optimal rewards.

7.5.2 ***Topology Proportion.*** As highlighted in Section 5, the utilization of the MST topology offers the advantage of lower operational expenses, although it may not provide the same level of reliability as the RT topology. The MST

(a) Env. model=SL1 (b) Env. model=SL2

Fig. 12. Expected Total Reward of Scalarised MR-MOMDP and MR-POMDP using SARSOP

topology can be a suitable option when the risk of disruptions is low, such as in non-detrimental conditions. However, it may not be ideal in detrimental situations where reliability is crucial.

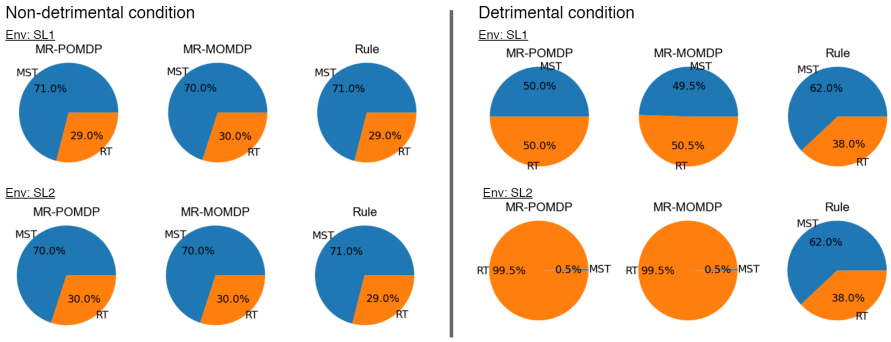


Fig. 13. Proportions of topology under different conditions.

In Fig. 13, the usage proportions of the MST and RT topologies are depicted for the SL1 and SL2 environments, respectively. In non-detrimental conditions, MR-MOMDP, MR-POMDP, and the Rule-based adaptation demonstrate similar proportions in terms of utilizing the two topologies. However, in detrimental conditions, there are notable differences.

The rule-based adaptation only marginally increases the allocation of RT by approximately 9-11% in detrimental conditions. On the other hand, both MR-MOMDP and MR-POMDP significantly increase the portion of RT. In the SL1 environment, they allocate RT by approximately 20%. However, in the more stringent SL2 environment, MR-MOMDP and MR-POMDP dramatically increase the allocation of RT by about 70%. This shift towards RT usage is driven by the need to adhere to stricter MR constraints in SL2-Detrimental compared to SL1-Detrimental. However, it is important to note that this higher reliance on RT in SL2-Detrimental leads to increased operational costs compared to SL1-Detrimental.

7.5.3 ***Target Satisfaction.*** Based on the specified targets in Table 3, the objective is to minimize the average values of WRT (MP) and BWC (MC) while maximizing the average value of ANL (MR). Fig. 14 illustrates that under normal

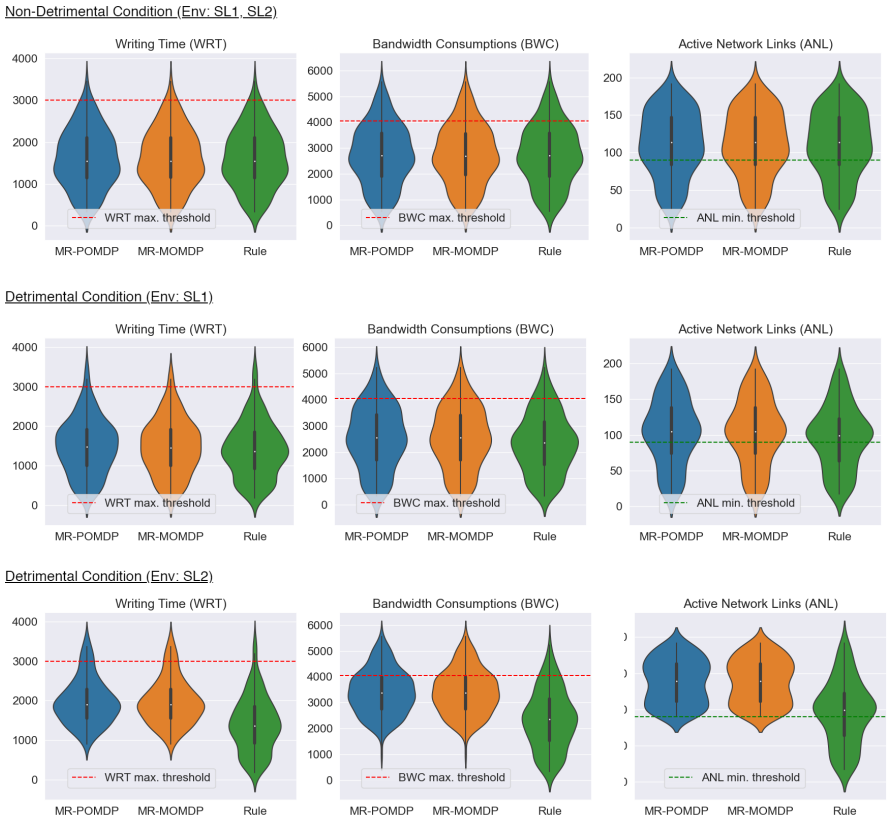


Fig. 14. Target satisfaction under various conditions.

or non-detrimental conditions, MR-MOMDP, MR-POMDP, and the Rule-based adaptation perform equally well in achieving all the targets.

However, in detrimental conditions, a notable difference arises. The Rule-based adaptation yields a lower WRT and BWC average than MR-MOMDP and MR-POMDP. This discrepancy can be attributed to the rule-based adaptation relying more heavily on the MST topology than MR-MOMDP and MR-POMDP. Consequently, when detrimental conditions occur, disruptions in the MST topology reduce the number of active links, resulting in decreased values of both BWC and WRT.

On the contrary, MR-MOMDP and MR-POMDP adapt their strategy by increasing the allocation of the RT topology to maintain system reliability. This strategic adjustment allows them to equally satisfy all the targets while outperforming the Rule-based adaptation in satisficing the ANL objective.

To summarize, under detrimental conditions, the Rule-based adaptation exhibits lower values of WRT and BWC due to its reliance on the MST topology, which is more susceptible to disruptions. In contrast, MR-MOMDP and MR-POMDP respond by prioritizing the RT topology to enhance system reliability, resulting in better performance across all targets and excelling in satisficing the MR requirements.

7.5.4 ***Constraint Violation***. Regarding constraint violation, as shown in Fig. 15, none of the approaches, including MR-POMDP, MR-MOMDP, and rule-based adaptation, violate any constraints under non-detrimental conditions. However, when subjected to detrimental conditions, the rule-based adaptation violates the MR constraints in SL1 and SL2, indicating its inability to handle detrimental conditions effectively. On the other hand, both MR-POMDP and MR-MOMDP still manage to avoid violating any constraints.

Additionally, during the evaluation, we specifically observed the number of threshold violations between MR-MOMDP and MR-POMDP across different random seeds. The aim was to assess the performance of the two approaches in maintaining the predefined thresholds for the NFRs. Notably, our findings indicate that the number of violations is comparable between MR-MOMDP and MR-POMDP. This implies that the two approaches exhibit similar capabilities in effectively maintaining the set thresholds for the NFRs.

Based on these findings, we can conclude that the default RDMSim rule-based adaptation cannot effectively satisfy the NFRs in this experiment. In contrast, MR-POMDP and MR-MOMDP exhibit better performance by successfully adhering to the MR constraints, even under detrimental conditions. However, one can improve the rule-based adaptation by adding more rules to address different conditions, which is often impractical.

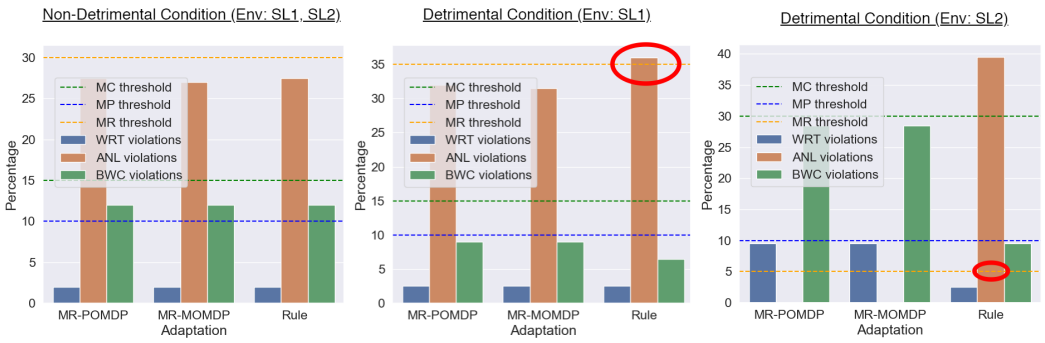


Fig. 15. Constraint violation under various conditions.

8 THREATS TO VALIDITY

8.1 Threats to external validity

Threats to external validity are associated with the generalisability of results and the replicability of experiments. Our testing was conducted using RDMSim simulator [53], which has also been used in [51, 52]. *We present two models and different environmental scenario conditions with the initial configuration provided by domain experts.*

Our implementation of Markov models relies on several key assumptions, including constant transition probabilities and discrete state representations. Specifically, the models assume that the probabilities of transitioning between states remain fixed over time. However, some systems may exhibit non-stationary behavior, where transition probabilities evolve due to factors such as system wear, user behavior, or external influences. Additionally, the use of discrete state representations may be inadequate for environments that require a continuous range of state values. In more complex and dynamic settings, extending the models to accommodate these factors is essential to ensure accurate and reliable results.

Another issue is the computational cost of both MR-MOMDP and MR-POMDP, which can become intractable in their worst-case scenarios when the number of states used is too large, even though MR-MOMDP requires a shorter computation time than MR-POMDP. Our approach adopts [25, 52], which uses states as representations of combinations of two values for the satisfaction levels of NFRs (i.e., True and False). For example, 2 NFRs will result in 4 states, 3 NFRs will yield 8 states, and so on. Therefore, limiting the number of NFRs used to drive the self-adaptation is recommended when implementing our proposal. To strive for replicability, we provided a detailed experiment setup and configuration used in the simulator. Our implementation code, model, and simulator are also available online for the public [30].

8.2 Threats to internal validity

Threats to internal validity can be associated with randomness and settings in the implementation. We observe that the randomness of the environment affects the test results. Changes and differences in values for each component in the model will result in different policies.

To minimize potential biases arising from specific implementation or environmental factors, we conducted experiments under different environment/constraint configurations. We also used multiple random seeds during our experiments to ensure consistent results while capturing diversity in possible outcomes. This approach helps to reduce the impact of random fluctuations and strengthens the overall robustness of our findings.

We evaluated MR-MOMDP against MR-POMDP using the same solver and similar weight vectors. Additionally, we also applied scalarization techniques using identical weight vectors for a fair comparison between the two models. This uniformity in experimental setup ensures that differences in performance can be attributed to the models themselves rather than the solvers or other parameterizations.

8.3 Threats to construct validity

Threats to construct validity are related to the metrics' suitability, which is crucial to accurately assess the models concerning the research objectives. To anticipate this, we utilized various metrics in the evaluation to address our research questions. The metrics employed include planning time - which measures the computational efficiency shown by the time required to generate policies, expected total reward - which evaluates the overall effectiveness of the policies by quantifying their ability to maximize system utility, CCS size, target satisfaction - which reflects the models' ability to achieve specified objectives, and constraint violation - that quantified the adherence of policies to system requirements.

9 RELATED WORK

This section presents relevant literature that aligns closely with our proposal. Our focus is solely on those works that employ the Markov model within the realm of decision-making for self-adaptation. To emphasize the unique aspects of our approach, we have divided this section into two parts. A comparison of our proposal with some of the most relevant previous work is summarized in Table 15.

9.1 Observability

Introduced by the Reinforcement Learning (RL) and the robotics community, MDPs and POMDPs have been applied to model sequential decision-making processes in various domains of SAS, including SOA-based systems [26, 47, 61, 65], cloud computing [38, 39, 42], mission planning [27, 28], and security [6, 37]. While one may assume that the impact of the adaptation action is deterministic [6], much research based on Markov processes assumes that the effects of adaptation on NFRs are uncertain.

Table 15. Comparison of our proposal with existing approaches

Approach	# of NFRs	State space represents NFR satisfaction information?	Observability			Reward
			Full	Partial	Mixed	
Hybrid planning [42]	Single	No	✓			Scalar
CSSC-MDP [47]	Multiple	Yes	✓			Scalar
ADAM [26]	Multiple	Yes	✓			Scalar
MDP-LSP with HTN [65]	Multiple	Yes	✓			Scalar
Wang et.al. [61]	Single	Yes		✓		Scalar
RE-STORM-ARRoW [43]	Multiple	Yes		✓		Scalar
MR-POMDP++ [52]	Multiple	Yes		✓		Vector
Filieri et.al. [15]	Multiple	No	✓			Scalar
Moreno et.al. [19]	Multiple	No	✓			Scalar
SPECTRA (our proposal)	Multiple	Yes	✓	✓	✓	Vector

The studies conducted by [25, 26, 42, 43, 47, 52, 61, 65] share common objectives with our work in the realm of enhancing Quality of Service (QoS) and ensuring the satisfaction of NFRs. The works by [25, 26, 43, 47, 52, 61, 65] utilize state space to represent information related to the satisfaction level of NFRs. However, only RE-STORM [25] and Pri-AwaRE [51, 52] specifically demonstrate how NFRs, which only can be partially observed in the context, are mapped and represented by the states of the POMDP model.

In other studies using MDP [2, 6, 26–28, 37–39, 42, 47, 65], the agent works in an environment where it could fully observe the current state. However, in POMDP-based systems [25, 43, 52, 61], the agent can only partially observe the current state of the environment. Adopting POMDP in SAS is less common than MDP for various reasons. Besides additional computational complexity, not all domains or applications require modeling and reasoning under partial observability. Most of the existing studies considering full observability of the satisfaction state of NFRs either use a general goal model [11, 16–18, 20, 21] or an implementation of approaches such as Markov Decision Process (MDP), Discrete-Time Markov Chains (DTMCs) [15, 19], Monte Carlo Tree Simulations [9] or Dynamic Decision Networks [13, 14, 22]. The limitation of these approaches, considering full observability, is that they do not model the uncertainty related to the satisfiability of the NFRs in a system.

It is important to note that all studies mentioned above only focus on one type of observability (i.e., homogeneous observability), either fully observable or partially observable states.

MOMDP [41] considers mixed observability and solves it by representing separately the fully and partially observable components of an environment's state (i.e., robot) and derives a compact lower-dimensional representation of its belief space, which can be combined with any point based algorithm to compute approximate POMDP solutions, resulting in significant improvements in performance.

Our proposal, SPECTRA, builds upon RE-STORM [25] and Pri-AwaRE [51] by incorporating a state space model to represent the satisfaction level of NFRs. We recognize the significance of using MDP and POMDP in environments with uniform observability. However, unlike existing approaches, our work takes a step further by incorporating MOMDP to handle decision-making in mixed observability environments, an under-explored area in Software Engineering and Requirements Engineering in particular.

9.2 Reward Function

Existing approaches generally use single-objective approaches with scalar rewards in their models [2, 6, 8, 9, 15, 19, 25, 27, 28, 37–39, 42, 43, 47, 61, 65]. Although not explicitly stated, works by [42, 65] rely on the reward function to meet NFRs instead of using NFR satisfaction level as a determinant of the environment's current state. Although their model does not explicitly provide information on the satisfaction level of NFRs, the scalar reward function utilized in their MDP model strives to optimize the system's overall utility by prioritizing the essential quality constraints.

While scalar rewards are generally static, ARRoW [44] introduces dynamic rewards representing the preferences/weights over NFRs updated at runtime by monitoring the current levels of satisfaction of NFRs (i.e., states), the satisfaction threshold for each NFR, and comparing it to a set of satisfaction ranges. The weights associated with the NFRs are mapped onto a decision-making specification using a quantitative requirement prioritization scheme based on the P-CNP method. The resulting priorities are then aggregated into a single scalar reward value representing the overall importance of each NFR.

One issue with the scalar approach is that it hides important information about the significance of meeting each NFR in various states of the environment [52]. Scalar rewards also obscure the effects of decisions on adapting to specific NFRs. MR-POMDP++ [52] tackles the problem of scalar reward by suggesting an approach based on MR-POMDP [56]. This method employs a reward vector to represent the priorities of objectives (i.e., NFRs) by indicating how desirable they are in terms of satisfaction, given a particular state of the environment. The use of reward vectors adds complexity but offers transparency to capture the different impacts of adaptive decisions on the satisfaction of each individual NFR.

Echoing the importance of explicitly defining the significance of each NFR through vector rewards, our work goes beyond MR-POMDP++ by considering the mixed observability of using MOMDP and contributing to a multi-reward version of MOMDP (i.e., MR-MOMDP, described in Section 4) which can be applied in various domains. The SPECTRA we propose covers the multi-reward/vector reward MDP, MR-POMDP, and MR-MOMDP and serves as a guideline for designing priority-aware decision-making models of SAS to satisfy NFRs tradeoffs.

10 CONCLUSION AND FUTURE WORK

We have proposed SPECTRA, a framework that can be used as a guide in designing decision-making tradeoffs for satisficing multiple NFRs in SAS using various types of Markov models according to the observability properties of the NFRs involved. Through the MirrorNet scenario, we have shown that MR-MOMDP is more suitable for contexts with NFRs showing mixed observability. The evaluation shows that MR-MOMDP can shorten planning time, produce higher expected values total rewards, and better satisfy NFRs. This findings encourage further development and exploration in other domains with mixed observability.

The current implementation of SPECTRA requires enhancement to make it more effective in real-world applications. Several methods have the potential to work with OLSAR for offline planning (e.g., SARSOP [41]) as well as online planning (e.g., DESPOT [66]). Further exploration is also needed for fine-tuning the selection of weight vectors from CCS generated by OLS, as our current implementation still relies on human judgment for this task. Additionally, we will explore the SPECTRA framework's application in satisfying NFRs in different socio-technical contexts, including Human-in-The-Loop Cyber-Physical Systems, where humans can play different roles [31]. Some NFRs in socio-technical systems are 'soft-requirements' [58], linked to soft goals involving partially known human attributes like values, beliefs, motivations, and personalities, making their satisfaction challenging to measure [58].

ACKNOWLEDGMENTS

Ignatius' work was supported by the Indonesia Endowment Fund for Education (LPDP).

REFERENCES

[1] Mohammad Abu Alsheikh, Dinh Thai Hoang, Dusit Niyato, Hwee-Pink Tan, and Shaowei Lin. 2015. Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys Tutorials* 17, 3 (2015), 1239–1267. https://doi.org/10.1109/COMST.2015.2420686

[2] Marco Abundo, Valeria Cardellini, and Francesco Lo Presti. 2011. An MDP-based admission control for a QoS-aware service-oriented system. In *2011 IEEE Nineteenth IEEE International Workshop on Quality of Service*. 1–3. https://doi.org/10.1109/IWQOS.2011.5931324

[3] Lucas Nunes Alegre, Ana Bazzan, and Bruno C Da Silva. 2022. Optimistic linear support and successor features as a basis for optimal policy transfer. In *International Conference on Machine Learning*. PMLR, 394–413.

[4] Craig Boutilier. 2002. A POMDP formulation of preference elicitation problems. In *AAAI/IAAI*. Edmonton, AB, 239–246.

[5] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. 2004. Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature-PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings 8*. Springer, 722–731.

[6] 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 (Dec. 2018), 51–69. https://doi.org/10.1016/j.scico.2018.07.002

[7] Radu Calinescu, Lars Grunske, Marta Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. 2011. Dynamic QoS Management and Optimization in Service-Based Systems. *IEEE Transactions on Software Engineering* 37, 3 (2011), 387–409. https://doi.org/10.1109/TSE.2010.92

[8] 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.

[9] Rodrigo Saar de Moraes and Simin Nadjm-Tehrani. 2024. NetGAP: A graph grammar approach for concept design of networked platforms with extra-functional requirements. *Engineering Applications of Artificial Intelligence* 133 (2024), 108089.

[10] Tobias Dürschmid, Eunsuk Kang, and David Garlan. 2019. Trade-off-oriented development: making quality attribute trade-offs first-class. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 109–112.

[11] 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, NY, USA.

[12] P Emami, AJ Hamlet, and C Crane. 2015. Pomdpy: An extensible framework for implementing pomdps in python. (2015).

[13] Bencomo et.al. 2013. Bayesian artificial intelligence for tackling uncertainty in self-adaptive systems: The case of dynamic decision networks. In *RAISE 2013*.

[14] Bencomo et.al. 2013. Dynamic Decision Networks to Support Decision-making for Self-adaptive Systems. In *(SEAMS)*.

[15] Filieri et.al. 2016. Supporting Self-Adaptation via QuantitativeVerification and Sensitivity Analysis at Run Time. *TSE* (2016).

[16] García-Galán et.al. 2014. User-centric Adaptation of Multi-tenant Services: Preference-based Analysis for Service Reconfiguration. In *SEAMS 2014*.

[17] Liaskos et.al. 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

[18] Letier et.al. 2014. Uncertainty, Risk, and Information Value in Software Requirements and Architecture. In *Proceedings of ICSE* (Hyderabad, India) *(ICSE 2014)*. 12 pages.

[19] Moreno et.al. 2016. Efficient Decision-Making under Uncertainty for Proactive Self-Adaptation. In *IEEE ICAC*. 147–156.

[20] Sousa et.al. 2008. User guidance of resource-adaptive systems. In *Conference on Software and Data Technologies*.

[21] Song et.al. 2013. Self-Adaptation with End-User Preference: Using Run-Time Models and Constraint Solving. In *MODELS*. USA.

[22] Saeed et.al. 2018. Non-functional requirements trade-off in self-adaptive systems. In *RESACS 2018*. IEEE, 9–15.

[23] Antonio Filieri, Martina Maggio, Konstantinos Angelopoulos, Nicolás D'ippolito, Ilias Gerostathopoulos, Andreas Berndt Hempel, Henry Hoffmann, Pooyan Jamshidi, Evangelia Kalyvianaki, Cristian Klein, Filip Krikava, Sasa Misailovic, Alessandro V. Papadopoulos, Suprio Ray, Amir M. Sharifloo, Stepan Shevtsov, Mateusz Ujma, and Thomas Vogel. 2017. Control Strategies for Self-Adaptive Software Systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 11, 4 (Feb. 2017), 1–31. https://doi.org/10.1145/3024188

[24] Luis Garcia, Huma Samin, and Nelly Bencomo. 2024. Decision Making for Self-Adaptation Based on Partially Observable Satisfaction of Non-Functional Requirements. *ACM Transactions on Autonomous and Adaptive Systems* 19, 2 (April 2024), 1–44. https://doi.org/10.1145/3643889

[25] Luis Hernan Garcia Paucar and Nelly Bencomo. 2018. RE-STORM: Mapping the Decision-Making Problem and Non-functional Requirements Trade-Off to Partially Observable Markov Decision Processes. In *2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 19–25.

[26] Carlo Ghezzi, Leandro Sales Pinto, Paola Spoletini, and Giordano Tamburrelli. 2013. Managing non-functional uncertainty via model-driven adaptivity. In *2013 35th International Conference on Software Engineering (ICSE)*. 33–42. https://doi.org/10.1109/ICSE.2013.6606549

[27] Robert P. Goldman, David J. Musliner, and Kurt D. Krebsbach. 2003. Managing Online Self-adaptation in Real-Time Environments. In *Self-Adaptive Software: Applications*. Springer, Berlin, Germany, 6–23. https://doi.org/10.1007/3-540-36554-0_2

[28] Mohand Hamadouche, Catherine Dezan, and Kalinka R. L. J. C. Branco. 2020. Reward Tuning for self-adaptive Policy in MDP based Distributed Decision-Making to ensure a Safe Mission Planning. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*

Workshops (DSN-W). 78–85. https://doi.org/10.1109/DSN-W50199.2020.00025

[29] IBM. 2023. What is observability? https://www.ibm.com/topics/observability [Online; accessed 17. Sep. 2023].

[30] Hargyo T.N. Ignatius. 2023. XPOMDPy. https://github.com/hxi903/xpomdpy

[31] Hargyo T. N. Ignatius and Rami Bahsoon. 2021. A Conceptual Reference Model for Human as a Service Provider in Cyber Physical Systems. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 1–10. https://doi.org/10.1109/SEAMS51251.2021.00012

[32] Minwen Ji, Alistair Veitch, and John Wilkes. 2003. Seneca: Remote Mirroring Done Write. In *2003 USENIX Annual Technical Conference (USENIX ATC 03)*. USENIX Association, San Antonio, TX. https://www.usenix.org/conference/2003-usenix-annual-technical-conference/seneca-remote-mirroring-done-write

[33] Edi Karni and David Schmeidler. 1991. Utility theory with uncertainty. *Handbook of mathematical economics* 4 (1991), 1763–1831.

[34] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2015. Software-Defined Networking: A Comprehensive Survey. *Proc. IEEE* 103, 1 (2015), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

[35] Nupul Kukreja. 2013. Decision Theoretic Requirements Prioritization: A Two-Step Approach for Sliding towards Value Realization. In *Proceedings of the 2013 International Conference on Software Engineering* (San Francisco, CA, USA) *(ICSE '13)*. IEEE Press, 1465–1467.

[36] Dean Leffingwell and Scaled Agile Team. 2023. Nonfunctional Requirements - Scaled Agile Framework. https://scaledagileframework.com/nonfunctional-requirements [Online; accessed 22. Jun. 2023].

[37] Junwei Liang, Maode Ma, and Xu Tan. 2021. Gadqn-ids: A novel self-adaptive ids for vanets based on bayesian game theory and deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* 23, 8 (2021), 12724–12737.

[38] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2015. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*. 1–12.

[39] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2018. Flexible and efficient decision-making for proactive latency-aware self-adaptation. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 13, 1 (2018), 1–36.

[40] Ivana Ognjanovic, Dragan Gašević, Ebrahim Bagheri, and Mohsen Asadi. 2011. Conditional Preferences in Software Stakeholders' Judgments. In *Proceedings of the 2011 ACM Symposium on Applied Computing* (TaiChung, Taiwan) *(SAC '11)*. Association for Computing Machinery, New York, NY, USA, 683–690. https://doi.org/10.1145/1982185.1982335

[41] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. 2010. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research* 29, 8 (2010), 1053–1068.

[42] Ashutosh Pandey, Gabriel A Moreno, Javier Cámara, and David Garlan. 2016. Hybrid planning for decision making in self-adaptive systems. In *2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. IEEE, 130–139.

[43] Luis Hernán García Paucar and Nelly Bencomo. 2019. Knowledge base K models to support trade-offs for self-adaptation using Markov processes. In *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. IEEE, 11–16.

[44] Luis H. Garcia Paucar, Nelly Bencomo, and Kevin Kam Fung Yuen. 2019. ARRoW: Automatic runtime reappraisal of weights for self-adaptation. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (Limassol, Cyprus) *(SAC '19)*. Association for Computing Machinery, New York, NY, USA, 1584–1591. https://doi.org/10.1145/3297280.3299743

[45] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[46] Alex Ramos, Marcella Lazar, Raimir Holanda Filho, and Joel J. P. C. Rodrigues. 2017. Model-Based Quantitative Network Security Metrics: A Survey. *IEEE Communications Surveys Tutorials* 19, 4 (2017), 2704–2734. https://doi.org/10.1109/COMST.2017.2745505

[47] Lifang Ren, Wenjian Wang, and Hang Xu. 2020. A Reinforcement Learning Method for Constraint-Satisfied Services Composition. *IEEE Transactions on Services Computing* 13, 5 (2020), 786–800. https://doi.org/10.1109/TSC.2017.2727050

[48] Diederik M Roijers, Shimon Whiteson, and Frans A Oliehoek. 2015. Point-based planning for multi-objective POMDPs. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence (IJCAI)*, Vol. 2015. AAAI Press, 1666–1672.

[49] Eric Rutten, Nicolas Marchand, and Daniel Simon. 2017. Feedback Control as MAPE-K Loop in Autonomic Computing. (2017), 349–373.

[50] Maria Salama, Rami Bahsoon, and N Bencomo. 2017. Managing trade-offs in self-adaptive software architectures: A systematic mapping study. *Managing trade-offs in adaptable software architectures* (2017), 249–297.

[51] Huma Samin, Nelly Bencomo, and Peter Sawyer. 2021. Pri-AwaRE: Tool Support for priority-aware decision-making under uncertainty. In *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 450–451.

[52] Huma Samin, Nelly Bencomo, and Peter Sawyer. 2022. Decision-making under uncertainty: be aware of your priorities. *Software and Systems Modeling* 21, 6 (2022), 2213–2242.

[53] Huma Samin, Luis H. Garcia Paucar, Nelly Bencomo, Cesar M. Carranza Hurtado, and Erik M. Fredericks. 2021. RDMSim: An Exemplar for Evaluation and Comparison of Decision-Making Techniques for Self-Adaptation. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 238–244. https://doi.org/10.1109/SEAMS51251.2021.00039

[54] David Silver and Joel Veness. 2010. Monte-Carlo planning in large POMDPs. *Advances in neural information processing systems* 23 (2010).

[55] Richard D Smallwood and Edward J Sondik. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations research* 21, 5 (1973), 1071–1088.

[56] Harold Soh and Yiannis Demiris. 2011. Evolving policies for multi-reward partially observable Markov decision processes (MR-POMDPs). In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 713–720.

[57] Matthijs T. J. Spaan and Nikos Vlassis. 2005. Perseus: Randomized Point-Based Value Iteration for POMDPs. *J. Artif. Int. Res.* 24, 1 (aug 2005), 195–220.

[58] Alistair Sutcliffe, Pete Sawyer, and Nelly Bencomo. 2022. The Implications of 'Soft'Requirements. In *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 178–188.

[59] Jyrki Wallenius, James S Dyer, Peter C Fishburn, Ralph E Steuer, Stanley Zionts, and Kalyanmoy Deb. 2008. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management science* 54, 7 (2008), 1336–1349.

[60] Erwin Walraven and Matthijs Spaan. 2017. Accelerated vector pruning for optimal POMDP solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[61] Hongbing Wang, Xiaojun Wang, and Qi Yu. 2013. Optimal Self-Healing of Service-Oriented Systems with Incomplete Information. In *2013 IEEE International Congress on Big Data*. 227–234. https://doi.org/10.1109/BigData.Congress.2013.38

[62] Colin Werner, Ze Shi Li, Derek Lowlind, Omar Elazhary, Neil Ernst, and Daniela Damian. 2022. Continuously Managing NFRs: Opportunities and Challenges in Practice. *IEEE Transactions on Software Engineering* 48, 7 (2022), 2629–2642. https://doi.org/10.1109/TSE.2021.3066330

[63] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H.C. Cheng, and Jean-Michel Bruel. 2009. RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *2009 17th IEEE International Requirements Engineering Conference*. 79–88. https://doi.org/10.1109/RE.2009.36

[64] Karl Wiegers and Joy Beatty. 2013. *Software requirements*. Pearson Education.

[65] Jiuyun Xu, Kun Chen, and Stephan Reiff-Marganiec. 2011. Using markov decision process model with logic scoring of preference model to optimize htn web services composition. *International Journal of Web Services Research (IJWSR)* 8, 2 (2011), 53–73.

[66] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. 2017. DESPOT: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research* 58 (2017), 231–266.

[67] K. K. F. Yuen. 2012. Pairwise opposite matrix and its cognitive prioritization operators: comparisons with pairwise reciprocal matrix and analytic prioritization operators. *Journal of the Operational Research Society* 63, 3 (March 2012), 322–338. https://doi.org/10.1057/jors.2011.33

Received 30 June 2023; revised 23 January 2025; accepted 15 April 2025