Surprise! Surprise! Learn and Adapt

Huma Samin
Exeter University
Exeter, United Kingdom
h.samin@exeter.ac.uk

Dylan Walton
Durham University
Durham, United Kingdom
nelly.bencomo@durham.ac.uk

Nelly Bencomo
Durham University
Durham, United Kingdom
dylan.j.walton@durham.ac.uk

ABSTRACT

Self-adaptive systems (SAS) adjust their behavior at runtime in response to environmental changes, which are often unpredictable at design time. SAS must make decisions under uncertainty, balancing trade-offs between quality attributes (e.g., cost minimization vs. reliability maximization or energy consumption minimization vs. performance maximization), based on the impact of possible adaptation actions. Traditionally, SAS have been designed with fixed assumptions about these impacts, but such assumptions may not always hold during execution. Therefore, SAS require techniques to learn the actual impact of adaptation actions at runtime to support informed decision-making. This paper introduces the concept of *Surprise*, where an SAS detects deviations between its assumed and observed impacts during execution, enabling it to adjust its decisions accordingly. The approach is demonstrated through an application in the networking domain.

KEYWORDS

Surprise, Self-Adaptive Systems, Impacts of Adaptations, Broken Assumptions

ACM Reference Format:
Huma Samin, Dylan Walton, and Nelly Bencomo. 2025. Surprise! Surprise! Learn and Adapt. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Self-adaptive systems (SAS) adjust their behavior through adaptation actions in response to environmental changes that cannot be fully predicted at design time [6, 9, 21, 23, 35]. For instance, in an Internet-of-Things (IoT)-based SAS, dynamic factors such as traffic fluctuations or link failures in the network can affect energy consumption and performance. Consequently, SAS must balance trade-offs between conflicting quality attributes, such as cost minimization vs. reliability maximization or energy efficiency vs. performance optimization.

Traditionally, SAS use fixed assumptions about how adaptation actions influence quality attributes. However, unforeseen changes, such as unexpected link failures, may invalidate these assumptions [4, 35, 40, 41]. We argue that when these assumptions are broken, an SAS must relearn the actual impacts of adaptation actions and update its decision-making accordingly. This highlights the need for techniques that enable SAS to learn how adaptation actions affect quality attributes in dynamic environments.

The concept of *Surprise*, introduced in [4, 5], quantifies uncertainty by measuring deviations between expected and observed behavior in SAS. Surprise helps assess how new data impacts existing models or assumptions about the system's environment. We extend this idea by proposing that an SAS experiences a *Surprise* when the assumed impact of an adaptation action differs from the actual observed impact at runtime. Thus, Surprise serves as an indicator of broken assumptions and can be leveraged to infer updated impacts of adaptation actions, enabling more effective adaptations in uncertain conditions.

This paper presents *a novel Surprise-based learning approach* that allows SAS to dynamically update their assumed world model and refine their decision-making according to evolving environmental conditions. The main contribution of the paper is *a novel Surprise-based learning approach* that:

i) enables SAS to learn the actual impact of adaptation actions in unforeseen environmental conditions.

ii) enhances SAS resilience to uncertainty by updating the assumed world model with newly learned adaptation impacts.

As a proof of concept, we apply our approach to a remote data networking application [28, 33]. Our experiments demonstrate how newly learned adaptation impacts improve SAS behavior and decision-making.

The rest of this paper is structured as follows: Section 2 provides background information, Section 3 introduces the Surprise-based learning approach, and Section 4 details the experimental evaluations and results. Section 5 discusses threats to validity, and Section 6 reviews related work. Finally, Section 7 concludes the paper and outlines directions for future research.

2 BACKGROUND

To explain the proposed Surprise-based learning approach, we first outline the concepts of Surprise and Bayesian Reinforcement Learning using Partially Observable Markov Decision Processes (POMDPs), which support decision-making under uncertainty.

2.1 POMDPs and Design Assumptions in SAS

Bayesian Reinforcement Learning techniques, such as Partially Observable Markov Decision Processes (POMDPs) [36, 37], have been proposed as a framework for modeling decision-making in SAS. They facilitate reasoning about multiple objectives and the uncertainty associated with the environment's state [3, 34]. These models are typically designed based on assumptions made by domain experts at design time while leveraging Bayesian inference to quantify uncertainty. POMDPs provide a decision-making framework where an agent operates in a partially observable environment. The agent,

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

which represents an entity performing actions (i.e., adaptation actions in SAS), must respond to changing environmental conditions. Due to partial observability, the agent cannot directly perceive the actual underlying state but instead maintains a belief over the set of possible states (expressed as a probability distribution). Based on these beliefs and gathered observations, the agent selects an optimal adaptation action. An outline of this process is shown in Figure 1. The authors in [18] proposed a mapping to structure SAS decision-making as a POMDP-based model. Accordingly, a POMDP underpins SAS decision-making to find a policy that maps states to actions to maximize reward [34].

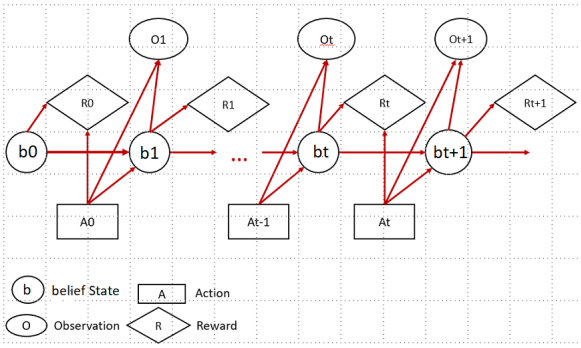

Figure 1: POMDP process [10]

A POMDP is defined as a tuple $< S, A, Z, T, O, R, \gamma >$, where:

- S represents the set of environment states.
- A is the set of possible actions that the decision-making agent can take.
- Z denotes the set of observations related to the environment's state.
- T is the transition function, which gives the probability of transitioning from one state to another.
- O is the observation function, containing the likelihood of a specific observation given the current state.
- R represents the reward function, which the agent seeks to maximize.
- γ is the discounting factor.

A particularly important component is T, which defines the *impacts* of actions on the state of the environment and serves as the system's environmental model (for SAS in this case). Transition probabilities are determined at design time by domain experts who possess knowledge and make assumptions about the system. However, in SAS decision-making, these assumptions regarding the impact of adaptation actions—represented by POMDP transition probabilities—may become invalid under new environmental conditions. Thus, updating the impact of adaptation actions to reflect their current value is necessary.

The proposed approach leverages Surprise as an indicator of broken assumptions, enabling the system to infer or learn the actual impact corresponding to new conditions.

2.2 The Concept of Surprise

Surprise is a measure of unexpectedness that quantifies the gap between expected and actual outcomes [5, 13]. It serves as a mechanism to record 'broken assumptions,' highlighting unexpected events while also indicating the urgency of the identified situation. In other words, Surprise quantifies the deviation of actual events or data from model predictions. There are several types of Surprise, but three specific definitions are particularly relevant to this paper, as discussed below:

Bayesian Definition of Surprise. A Bayesian Surprise is defined as the divergence between an observer's prior beliefs (represented as a probability distribution) and posterior beliefs following an observed event [2, 5]. Surprise is measured using the Kullback-Leibler (KL) divergence [2, 22], which quantifies the divergence between prior and posterior distributions:

$$S_{Bayes} = D_{kl}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)} \tag{1}$$

where $q(x)$ and $p(x)$ represent the prior and posterior belief distributions, respectively, and D_{kl} represents the KL divergence between them. POMDPs provide the prior and posterior beliefs necessary for Surprise computation.

A *wow* unit has been proposed to measure and quantify Bayesian Surprises [2, 29].

Other approaches to measuring Surprise include Confidence Corrected Surprise [13] and Bayes Factor Surprise [25].

Confidence Corrected Surprise. Confidence Corrected Surprise (S_{CC}) [11, 13] represents the difference between an observer's expected outcome (given by the current belief $\pi_0(\theta)$) and the actual outcome observed in the environment. The relevance of a new data point X is indicated by the scaled likelihood $p_x(\theta)$.

The scaled likelihood $p_x(\theta)$ represents the posterior belief derived under a flat prior, meaning the observer has no prior knowledge of the environment. In contrast, the prior $\pi_0(\theta)$ corresponds to the posterior belief without considering the newly acquired data sample (i.e., the current belief). Using KL divergence, S_{CC} quantifies the difference between these two posteriors, $\pi_0(\theta)$ and $p_x(\theta)$, as follows:

$$S_{CC} = D_{kl}(\pi_0(\theta)||p_x(\theta)) \tag{2}$$

Bayes Factor Surprise: The Bayes Factor Surprise S_{BF} [24, 25] is somewhat similar to S_{CC} in that it serves as a comparison between a naive prior (the flat distribution) and the distribution at any subsequent timestep t. S_{BF} of observing a data sample $x \in X$ is defined as the ratio of the probability of observing x under the naive prior π_0 to the probability of observing x under current belief π_t. It is computed as follows:

$$S_{BF} = \frac{p(x_{t+1}|\pi_0)}{p(x_{t+1}|\pi_t)} \tag{3}$$

Next, we present our Surprise-based learning approach to learn the impacts of adaptation actions for SAS.

3 SURPRISE! LEARN-ADAPT APPROACH

This section describes the Surprise-based learning approach for learning the new impact of adaptation actions. The presented approach makes use of the Surprise-Minimization Learning (SMiLe) Rule [11]. It also takes inspiration from the model-based learning approach presented in [42], which is quiet different from the case in this paper, that was used to learn human behaviour. In this paper, these new ideas are applied to the SAS-based domain to learn the new impacts of adaptation actions for SAS. In the SAS domain, the SMiLe rule seeks to minimize surprises to minimize the possibility of unexpected behaviour. Next, the example of a Remote Mirroring application [33] to support the description of the proposed Surprise-based learning approach is presented.

3.1 Illustrative Example: Remote Data Mirroring

Remote Data Mirroring (RDM) is a disaster recovery technique [28, 30]. It supports data recovery by maintaining multiple copies of the content over different servers (also known as mirrors). For the purpose of illustration of our approach, we consider a concrete example of an RDM network presented by a publicly available exemplar RDMSim [33]. The RDMSim artefact presents 3 quality attributes: *minimization of operational cost* (MC)[1], *maximization of performance* (MP)[2], and *maximization of reliability* (MR)[3]. The RDMSim system can perform the adaptation actions of switching between the two topology configurations for the network: i) *Minimum Spanning Tree* (MST) that seeks to use the least number of links reducing cost and performance but also reducing reliability. ii) *Redundant Topology* (RT) that improves reliability at the cost of operational cost and performance. According to the specifications of RDMSim [33], *MC is considered satisfied if the total bandwidth consumed is less than or equal to 3600 GBps, MR is satisfied if the total active links is greater than or equal to 105 active links and MP is satisfied if total writing time is less than or equal to 2700 milliseconds.* The RDMSim comes up with 6 different scenarios that represent different dynamic environmental situations [33].

Algorithm 1 Surprise-based Learning Algorithm

$N \leftarrow$ number of states of SAS
$B \leftarrow \pi_0$ ▷ current beliefs about the impacts of adaptation actions
$W \leftarrow$ flat distribution across the states
$m \leftarrow \frac{Pc}{1-Pc}$
while SAS Execution = True **do**
  Use W to determine next adaptation action
  $X_n \leftarrow$ new adaptation action-state pair
  1) evaluate $S_{Bayes}(X_n|B_{X_n})$
  2) $\gamma = \frac{m*S_{Bayes}(X_n|B_{X_n})}{1+(m*S_{Bayes}(X_n|B_{X_n}))}$
  3) $B_{X_n} = (1-\gamma)B_{X_n} + \gamma\pi_0 + \Delta$ ▷ SMiLE Rule
  4) $W_{X_n(t+1)} = \frac{W_{X_n t}}{\sum_{i=1}^{B_{X_n}} B_{X_n i}}$ ▷ Adaptation Impacts Update
end while

Next, we present our proposed Surprise-based learning approach:

[1]Operational Cost is measured in terms of total bandwidth consumed.
[2]Performance is measured in terms of total time to write data.
[3]Reliability is measured in terms of total active links.

3.2 Surprise-based Learning Approach

At the core of the proposed Surprise-based learning approach lies the concept of assumptions made by domain experts regarding the impact of adaptation actions (e.g., MST or RT in RDM) on the quality attributes of an SAS (e.g., MC, MP, and MR in RDM). Due to unforeseen environmental changes, the initial assumptions established at design time may no longer hold under newly encountered execution contexts. For instance, equipment failure during the execution of the MST topology may alter its impact on the satisfaction of quality attributes. In the Surprise-based learning approach, these assumptions are represented as probabilities. Specifically, the probability $P_{aiqj} = P(qj \text{ is satisfied} \mid \text{the adaptation action } ai)$ expresses the likelihood that quality attribute qj is satisfied given adaptation action ai. In the case of RDM, this can be written as $P(MR \text{ is satisfied} \mid MST \text{ adaptation action})$.

In the context of the POMDP, which is the decision-making engine used, these probabilities are known as beliefs, which here relate to the world being modelled. The beliefs in our context measure whether a quality attribute qj is satisfied given the adaptation action ai that has been chosen, which can be represented as $belief_{aiqj}$ and corresponds with the belief B in Algorithm 1. Moreover, the impacts of adaptation actions are represented as the transition probabilities, because they describe the probabilities of transition from one state to another after an adaptation action. Further, based on the changes in the environment of SAS, the Surprise (that suggests the existence of broken assumptions) hints at the need to update the beliefs about the impacts of adaptation actions. This is done by the use of SMiLe rule [13] in our proposed approach. The newly updated beliefs are then used to update the impacts of adaptation actions and the model of the world for SAS. The proposed Surprise-based learning approach is outlined in Algorithm 1.

In the algorithm, π_0 is the initial (flat) beliefs B about the impacts of adaptation actions, W represents the world model for SAS (specified by the transition probabilities in POMDPs). Delta Δ is a Kronecker function [12] which is used to indicate if the prior state and current state of the SAS are the same. The process is dependent on the following factors: the number of states of the SAS, the number of possible adaptation actions for the SAS and the type of Surprise used (i.e., S_{CC} or S_{BF}). We also need information about the adaptation rate γ which is modulated by a variable m referring to the rate of change in the environment. The variable m is further derived from an additional variable P_c, which represents the volatility of the environment.

Next, we outline the steps to learn the impacts of adaptation actions (POMDP transition probabilities) and update the world model W, where "beliefs" refer to assumptions about impacts.

Step 1: Initialization of the World Model for SAS and beliefs about the Adaptation Action Impacts:

Before SAS execution, the learner is initialized with beliefs B about the impacts of adaptation actions and the transition probabilities that form the world model W, using flat distributions. These beliefs predict how adaptation actions affect the state of the SAS, which is defined by the satisfaction of its quality attributes. For example, in self-adaptive RDM, the initial beliefs and world model for MST and RT actions on MC, MR, and MP attributes are set

uniformly. After initialization, SAS begins its loop to learn and improve transition probabilities during execution.

Step 2: Selection of Adaptation Action:
Once the SAS starts executing, it selects the adaptation action considering the environmental context and the initialized world model W to achieve the satisfaction of its quality attributes. Once the adaptation action given the current state for SAS is determined, the selected adaptation action is applied to come up with the next state for SAS. For example, given the environmental situation of link failures during MST topology, the self-adaptive RDM will plan to select the adaptation action of RT topology to satisfy reliability (MR) for the network. The application of the RT topology will have an impact of the state of satisfaction of all of the quality attributes MR, MC and MP leading the self-adaptive RDM to a new state. Based on this state-adaptation action pair, the respective transition probability X_n is selected that supports the transition of the SAS from one state to another.

Step 3: Calculation of Surprise:
Considering the above state-adaptation action pair, the next step is to compute the Surprise which, in this case, indicates the deviation between the actual impacts of adaptation actions (represented as transition probability X_n) and the assumed impacts i.e., the Belief B_{X_n} over the impact of adaptation action.

Step 4: Computation of Adaptation Rate:
Once the Surprise is computed, the next step involves the computation of the adaptation rate γ which (as previously stated) is based on the value of Surprise and the rate of change represented by m, itself dependent on the environments volatility as specified by a probability of change P_c. The P_c value is bound between 0 and 1 and is set by the domain experts. The rate of change m and adaptation rate γ are calculated as follows:

$$m = \frac{P_c}{1 - P_c} \tag{4}$$

$$\gamma = \frac{m * S_{Bayes}}{1 + (m * S_{Bayes})} \tag{5}$$

Hence, the adaptation rate for the SAS is informed by the Surprise to denote a change in the assumptions about the impacts of adaptation actions and the rate at which the environmental fluctuations occur indicated by the m. This information then supports the SAS to update the beliefs and the world model for SAS.

Step 5: Updating of the beliefs about the Impacts of Adaptation Actions:
Step 5 is based on the SMiLe rule [13] that involves updating the beliefs about the impacts of adaptation actions B_{X_n} considering the state-adaptation action pair selected in Step 2. The update in the belief is performed based on the adaptation rate γ, the initial belief π_0, and the current belief B_{X_n} as follows:

$$B_{X_n} = (1 - \gamma)B_{X_n} + \gamma\pi_0 + \Delta \tag{6}$$

Based on the updated Beliefs, the world model W is updated by updating the impacts of adaptation actions done in the next step.

Step 6: Update of the Impacts of Adaptation Actions:
The final step updates the impacts of adaptation actions (transition probabilities in POMDPs) using the learner's updated beliefs B and the world model W. Steps 2 to 6 are repeated throughout the SAS execution. At each timestep, the updated transition probabilities guide the SAS in selecting and performing adaptation actions. After each action, the results and new SAS state are fed back to the Surprise-based learner to update the impacts of the adaptation actions, adapting to the new environment.

The choice of P_c value: An important concept in the Surprise-based learning approach is the P_c value, which frames the volatility of the environment. Environmental volatility changes among different environmental situations, as certain situations are worse than others. Capturing these differences is important to produce a workable solution. When P_c is small and closer to 0, the environment is thought to be less volatile, so more importance can be set to environmental change. In contrast, when P_c is closer to 1, the environment is considered as unstable and prone to frequent and large changes. In the Surprise-based learning approach, the P_c value is used to control the adaptation rate, $\gamma > 0$, with gamma approaching 1 as P_c approaches 1. The γ value of 1 is equivalent to discarding all new information and entirely relying on the original distribution, γ set to 0 is the reverse, i.e., discarding all prior information and relying on the current belief solely. This means that in less volatile environments newer beliefs B are prioritized as any large changes to the beliefs mean something has unexpectedly gone wrong and needs to be amended promptly. For example, let's consider the case of the RDMSim: scenario 6 represents a highly volatile environmental situation where a significant site failure occurs. Hence, for such a situation, the P_c closer to 1 is considered.

To select a suitable P_c value, we start by testing values near 0 and 1, checking if the Surprise-based learning approach produces the desired SAS behavior under environmental fluctuations. Based on performance, we refine by choosing values above or below 0.5. For example, if a P_c value near 1 works better than one near 0, we test values between 0.5 and 1.0 to find the best adaptation behavior. Once acceptable P_c values are identified, one is selected as the "base" value for achieving desired adaptation impacts and satisfaction of quality attributes. Due to environmental variability, a single P_c value may not fit all scenarios, so the value is chosen based on quality attribute priorities for each specific scenario.

4 EXPERIMENTS

In this section, we present the experimental evaluation of the proposed Surprise-based learning approach. We employ POMDPs as the decision-making technique for selecting adaptation actions in SAS. The SAS learns the real-time impact of adaptation actions performed under dynamic runtime conditions. In the case of POMDP-based decision-making, these impact values are represented as transition probabilities that define the SAS's world model. For the experimental evaluation, we use the public exemplar RDMSim [28, 33], as described in Section 3. The experiments are conducted using S_{CC}, the Surprise measure utilized by the SMiLe rule [11]. Additionally, we perform further evaluations using the Bayes Factor Surprise (S_{BF}), which introduces the perspective of a naive observer for comparison purposes.

For benchmarking, we compare our approach with an existing POMDP-based decision-making method presented in [17, 18], which relies on expert-defined transition probabilities that remain

fixed during execution. In our experiments, we apply the technique to the RDMSim network case study. The proposed Surprise-based learning approach underpins the learning process of transition probabilities, enabling the system to update the impact of adaptation actions dynamically.

Evaluation metrics include **occurrence rates** (i.e., the number of times a specific topology is selected), **satisfaction rates** (i.e., the total number of times a given quality attribute is considered "satisfied"), and **individual performance metrics** for each quality attribute (i.e., read/write time, bandwidth consumption, and number of active links). Due to the nature of trade-offs, whether a given result is favorable depends on the scenario and quality attributes. A high MP/MC occurrence rate may be beneficial in scenarios where these values are expected to underperform but may be undesirable in others—especially if it comes at the expense of MR.

We conducted experiments for all seven RDMSim scenarios, each representing different dynamic environmental conditions in an RDM network. The results for all scenarios are available in a GitHub repository [39]. The execution of the algorithm took, on average, 0.016 seconds to learn the world model at a given time step for the RDM case (or just under 8 seconds to complete a run of 500 timesteps). All experiments were performed using an AMD Ryzen 5 7600X processor.

4.1 Initial Setup

As the first step of our experiments, we initialized the world model W for the self-adaptive RDM, represented by the transition probabilities and the initial beliefs B about the impacts of adaptation actions. Both the transition probabilities and beliefs were set to a uniform distribution. Next, we present the experimental evaluations carried out using S_{CC}. We describe the topology selection behavior of the Surprise-based learning approach by selecting different P_c values to better match the environmental fluctuations represented by various RDM scenarios [33].

4.2 Experiments using Confidence-Corrected Surprise

As noted, different P_c values are needed for various RDMSim scenarios due to their volatility. These values help the Surprise-based learning approach gauge environmental dynamics and improve adaptation. Initially, we tested $P_c = 0.333$, which did not produce the desired behavior—though it showed learning of adaptation impacts. Beyond 100 timesteps, only the MST topology was selected, establishing 0.333 as the lower bound. Additional values below 0.5 showed similar behavior, while values above 0.5 introduced more variation. $P_c = 0.583$ was the first to exhibit significant changes, confirming that larger P_c values are better suited for dynamic environments like RDM, which experience higher link failure rates. Further testing showed that P_c values above 0.667 did not yield the desired topology selection across different RDM scenarios. Although minor variations occurred at higher values, topology selection became inconsistent. Based on these observations, 0.667 was chosen as the upper bound.

Experiments then progressed to iterative testing of P_c values between 0.583 (the lowest value above 0.5 that showed variation in topology selection behavior) and 0.667. For clarity, we discuss the

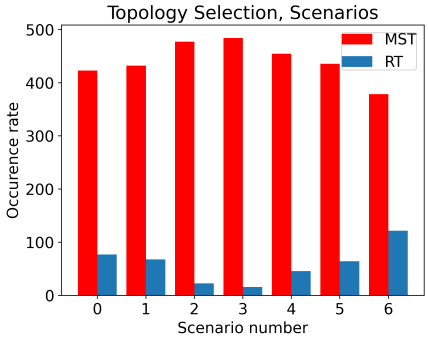

Figure 2: Average topology selections for scenarios 0 to 6, over 5 runs for the use of a $P_c = 0.6$

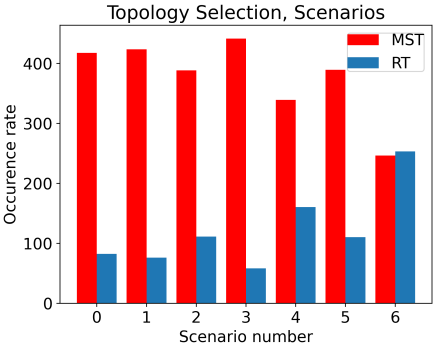

Figure 3: Average topology selections for scenarios 0 to 6, over 5 runs for the use of a $P_c = 0.655$

results for $P_c = 0.6$, 0.636, and 0.655 in detail. To begin, $P_c = 0.6$ was the first value to show promise regarding topology selection behavior. While previous values had demonstrated comparable satisfaction rates (at least when considering all three quality attributes together), they lacked judicious prioritization—often leading to under-prioritization of MR in terms of satisfaction. The approach exhibited the intended topology behavior for the first four scenarios (scenarios 0 through 3), though this behavior did not persist in the latter three. This outcome, while not ideal, was expected, as a single P_c value is unlikely to be optimal across all scenarios.

The key takeaway is that the Surprise-based learning approach successfully produced the intended behavior for some RDM scenarios, aligning with RDMSim specifications [33]. Regarding transition function quality (representing impacts), assessed by the average satisfaction rate per scenario, results for $P_c = 0.6$ and the expert-defined transition function (from [18], presented in Section 4.4) were comparable for the first four RDMSim scenarios. Figure 2 shows results for $P_c = 0.6$, while Figure 9 illustrates the expert-defined function's topology selection behavior. However, in the final three scenarios, average quality attribute satisfaction remained similar despite differing topology behavior. Both approaches prioritized quality attribute satisfaction according to RDMSim specifications [33].

Therefore, we can deduce that Surprise-based learning generated transition probabilities with satisfaction rates and behavior

comparable to those of expert-defined probabilities—without requiring expert effort and time. The achieved satisfaction levels for quality attributes are presented in Figure 4. A caveat, however, is that Surprise-based learning did not consistently produce transition probabilities as effective as those defined by experts. While it achieved good satisfaction rates, it occasionally misallocated resources, prioritizing variables that did not require them at the expense of those that did. This resulted in deceptively high overall satisfaction rates—where MR and MP were sometimes extremely high while MR remained too low, despite the intended goal of balancing them.

The results discussed so far represent mean values across multiple runs, making them sensitive to outliers and drastic shifts. In RDM scenario 0 and scenarios 4 to 6, values fluctuated significantly in about half of the runs. However, this does not discredit $P_c = 0.6$; rather, it suggests it may be unsuitable for certain scenarios. These variations offer insights into result changes, averaging effects, and how Surprise-based learning should generate transition probabilities for the World Model. The average topology selection results for $P_c = 0.6$ are shown in Figure 2, highlighting that a single run is insufficient for accurate transition probabilities in RDMSim.

The approach behaves most erratically in this case, exhibiting the worst failures. However, higher P_c values only exacerbate the instability and overall dissatisfaction of quality attributes in scenario 6. For example, the most common behavior observed was that RT was selected as frequently as, if not more than, MST—this pattern was repeated in all but one of the runs. The dilemma is that while it does not seem logical to use a lower P_c value for scenario 6 than for the others, doing so appears necessary to correct the observed behavior. Potential solutions will be presented later, but this issue remains a challenge for the base Surprise-based learning implementation, which we intend to explore further in future work.

The final P_c value demonstrated is $P_c = 0.636$, briefly discussed as it exhibits a mix of $P_c = 0.6$ and $P_c = 0.655$, as expected given its intermediate position. Scenarios 0 to 3 maintained reasonable satisfaction rates, while scenarios 4 to 6 showed some improvement over $P_c = 0.6$ despite continued poor performance. Although other values could have been tested, $P_c = 0.636$ is included for comparison with a proposed solution to address behavioral inconsistencies in later scenarios. Satisfaction levels for $P_c = 0.655$ and $P_c = 0.636$ are shown in Figures 5 and 6.

The remaining P_c values not explicitly covered here generally followed the established trend of increasing volatility for P_c values greater than 0.5. Although some anomalies were observed (e.g., $P_c = 0.615$ inexplicably exhibited low RT selection throughout), most deviations were attributable to the stochastic nature of the Surprise-based learning algorithm and "unlucky" runs that obscured the actual behavior. Based on the experimental results, the precise P_c values are not critically important as long as they remain within the bounds of 0.583 and 0.667 and follow the general trend where increasing P_c values correspond with grouped scenarios (i.e., scenario 0; scenarios 1 and 2; scenario 3; scenarios 4 and 5; scenario 6). The results for all tested P_c values are available in [39].

To further validate the topology selection behavior of the Surprise-based learning approach, we conducted additional evaluations by randomizing action selection during the initial iterations. The objective was to determine whether the Surprise-based learning approach inherently favors one topology over another. To this end, the first 50 action selections were randomized. If the approach exhibited a preference for a particular topology, this would significantly influence the rest of the run. Conversely, if no inherent bias was present, behavior should revert to the previously observed patterns after the first 50 timesteps. Surprisingly, the results revealed that not only did behavior change—suggesting that early topology selection plays a role in how the Surprise-based learning approach evolves—but also that this initial randomization significantly improved topology selection for the last three scenarios, as shown in Figure 7. This behavior in the final three scenarios can be attributed to the fact that they represent situations where frequent link failures occur due to equipment issues affecting both MST and RT topologies. Scenario 6, in particular, involves a major site failure in the RDM network. To ensure that this behavior was not a statistical anomaly, five additional runs were conducted for each scenario, in addition to the initial five. The results consistently reproduced the same behavior, confirming that the observed effects were not merely coincidental. These findings are presented in Figure 7.

A drawback, however, was that the results exhibited some degree of erratic behavior, particularly in scenario 4. Nevertheless, this still marked an improvement over previous implementations. As stated earlier, the reason for this improvement is that the increased volatility of later scenarios was reflected in the random actions, ultimately leading to better results. Additional P_c values of 0.6, 0.636, and 0.655 were also tested. However, only $P_c = 0.636$ showed any improvements, as presented in Figure 7.

4.3 Confidence Corrected Surprise vs Bayes Factor Surprise

For the experiments, both S_{CC} and S_{BF} were used to determine appropriate P_c values. However, it was quickly observed that S_{CC} produced more varied results, adapting more dynamically as environmental volatility fluctuated. In contrast, most P_c values for S_{BF} resulted in minimal change until exceeding 0.5, behaving similarly to S_{CC} at higher thresholds. While S_{CC} introduced both benefits and drawbacks through its variability, S_{BF} generally underperformed in topology selection, as shown in Figure 8. Additionally, satisfaction rates of quality attributes were inadequate, often trailing behind those achieved with S_{CC}. Consequently, S_{BF} was ultimately discontinued in favor of refining and improving S_{CC} results.

4.4 Comparison against Expert-Defined Impacts

We also compared our technique with the existing POMDP-based approach [18], which relies on expert-defined transition probabilities. Figure 9 illustrates the topology selection behavior of the expert function. For the first four scenarios, topology selection behavior closely aligns with the expert-defined technique, demonstrating that the Surprise-based learning approach can generate world models without requiring expert knowledge. However, in later scenarios, behavior deviates from expectations, negatively affecting the satisfaction rate of quality attributes. While using different P_c values for the first four scenarios can maintain these

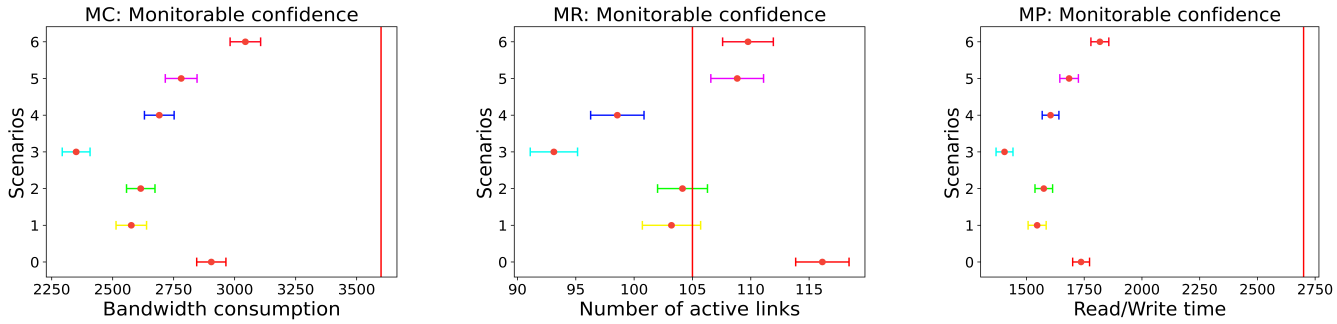

Figure 4: Satisfaction of quality attributes under RDM Scenarios for P_c value of 0.6. The red line indicates the threshold which all values should be beneath (or above for MR) to prevent failure.

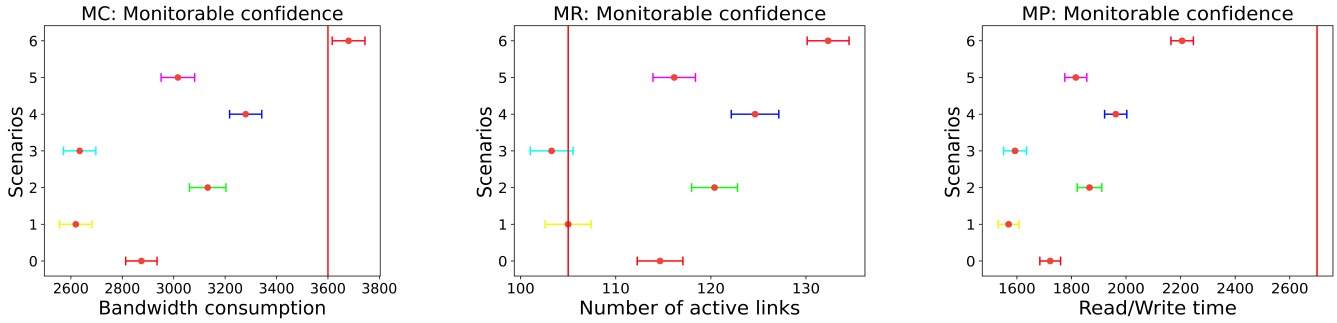

Figure 5: Satisfaction of quality attributes under RDM Scenarios for P_c value of 0.655. The red line indicates the threshold which all values should be beneath (or above for MR) to prevent failure.

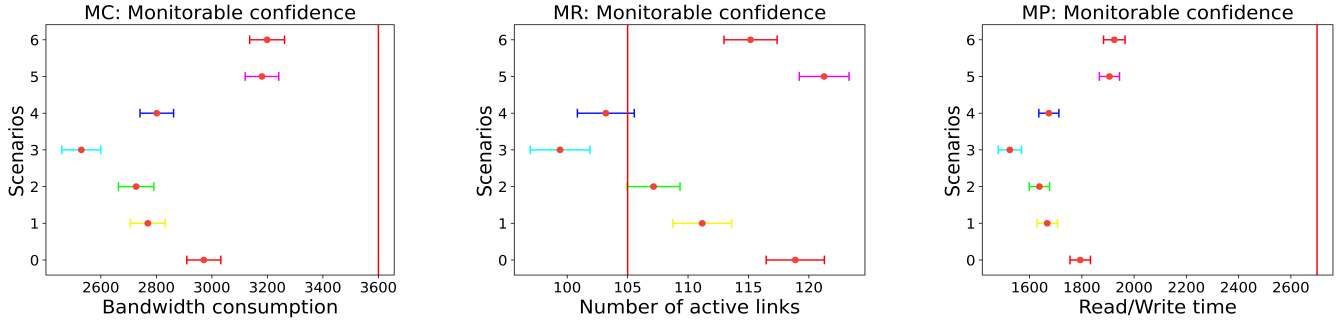

Figure 6: Satisfaction of quality attributes under RDM Scenarios for P_c value of 0.636. The red line indicates the threshold which all values should be beneath (or above for MR) to prevent failure.

benefits, higher P_c values in later scenarios still underperform compared to expert-defined models. An interesting observation is that introducing random actions for the first 50 timesteps appears to mitigate these issues, though it results in increased behavioral variability. Nonetheless, later scenarios again demonstrated topology selection behavior similar to expert-defined adaptation impacts.

These findings suggest that the Surprise-based learning approach can replicate expert-defined functions, particularly in earlier scenarios, and has the potential to do so for later scenarios with further refinement. Despite its limitations, this approach requires only a fraction of the time needed for an expert to manually design and analyze transition probabilities. Identifying appropriate P_c values simply necessitates basic analysis of the expected runtime environment and preliminary testing in either a real or simulated setting.

5 THREATS TO VALIDITY

The key threats to validity [14] are discussed as follows:

External Validity: We have executed experiments using the application of RDMSim [33] focusing on decision-making in a centralized setting while the adaptation decisions affect the whole network by making a change in topology. The approach has not been tested in a decentralized setup yet. More experiments are required to evaluate the application feasibility of the Surprise-based learning approach in both centralized and decentralized settings.

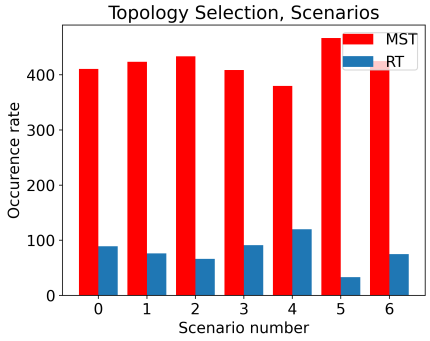

Figure 7: Average topology selections for scenarios 0 to 6 over 10 runs for the use of random action + P_c value of 0.636

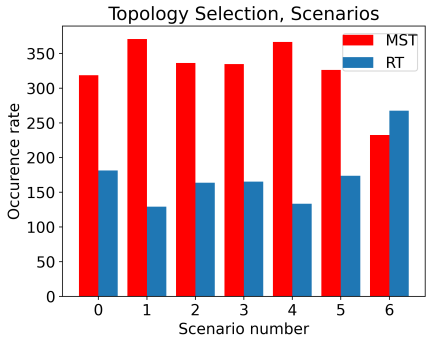

Figure 8: Average topology selections for scenarios 0 to 6, using Bayes Factor Surprise

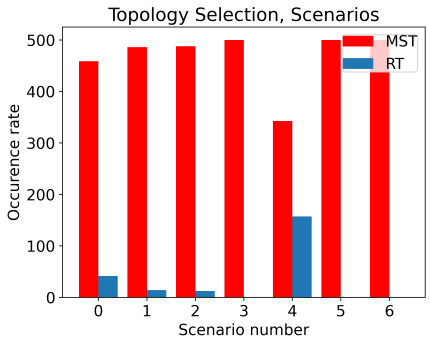

Figure 9: Average topology selections for scenarios 0 to 6, over 5 runs for Expert-defined Adaptation Impacts

Internal Validity: The internal threat to validity refers to the extent to which the Surprise-based learning approach performs in an actual environmental setup. In this paper, we have used a case study based on the exemplar/simulator RDMSim. The experiments' results are based on the environmental situations simulated by the RDMSim, and not an actual physical network. However, the RDMSim [33] is a well-accepted application in the research community and has been used by other resarch teams in its different versions [15, 30]. The simulation exemplar [33] provides support for simulations that are close enough to the real settings.

6 RELATED WORK

Several studies have explored the concept of *Surprise* to learn and model human behaviour [13, 26, 42]. In [26], Surprise modulates learning of world model through Bayesian inference. The approach balances the trade-off between forgetting old observations and integrating new ones to support the learning of human behaviour. Similarly, [13] uses a Surprise-Minimization framework (SMiLe) to model human behaviour by balancing new and old data in dynamic environments. In [42], Surprise is used to enhance learning in Reinforcement Learning (RL), improving both model-based and model-free branches. While inspired by this, our approach focuses on learning adaptation impacts for SAS, not human behaviour.

Surprise has also been applied in intelligent agents and autonomous robots for planning and learning action impacts [27, 31, 32, 38]. These methods use surprise to help robots autonomously plan, learn, and adapt to environmental changes. [7, 8] introduced Surprise-based learning to predict state transitions in robots using Stochastic Distinguishing Experiments (SDE) and Surprise-based POMDP (sPOMDP). Surprise has also been used for optimal exploration in dynamic environments [1, 27, 38] , motivating exploration by learning transition probabilities and rewards based on surprise. In contrast to these approaches, which focus on planning, exploration, or learning for robots and RL agents, our Surprise-based learning approach is designed to learn the impacts of adaptation actions in SAS. By focusing on the dynamic adaptation of SAS, our work addresses a distinct and underexplored application of Surprise in self-adaptive systems.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a *Surprise-based learning approach* to help SAS learn the impacts of adaptation actions on quality attribute satisfaction under changing environmental conditions. The approach leverages Surprise as an indicator of broken assumptions and enables the system to update these impacts accordingly. As a proof of concept, we applied our approach to an RDM network, with promising results demonstrating that the learned impacts aligned with the expected behavior of self-adaptive RDM [33].

A POMDP-based technique was employed for decision-making, where we compared two different transition functions used by the POMDP decision maker: one defined by an expert and another generated by our Surprise-based learner. While our learner shows significant potential, it still has room for improvement, particularly in ensuring consistency. However, our findings suggest that it can refine existing transition functions and assist in designing more effective decision-making models.

For future work, we aim to explore additional domains, such as Internet-of-Things (IoT) applications[20], and enhance the learner's adaptation capabilities to improve consistency and overall performance. Additionally, we plan to investigate alternative decision-making approaches beyond Bayesian methods [16, 19].

ACKNOWLEDGMENTS: This work was partially supported by Durham SRF JusTN0W (1815820) and EPSRC Project Twenty20Insight (EP/T017627/2).

REFERENCES

[1] Joshua Achiam et al. 2017. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732* (2017).

[2] Pierre Baldi and Laurent Itti. 2010. Of bits and wows: A Bayesian theory of surprise with applications to attention. *Neural Networks* (2010).

[3] Bencomo et al. 2019. RaM: Causally-connected Requirements-aware Models using Bayesian Inference. *MODELS* (2019).

[4] Nelly Bencomo. 2015. Quantun: Quantification of uncertainty for the reassessment of requirements. In *IEEE RE*. IEEE, 236–240.

[5] Nelly Bencomo and Amel Belaggoun. 2014. A world full of surprises: Bayesian theory of surprise to quantify degrees of uncertainty. In *ICSE*. 460–463.

[6] Betty H. Cheng and et al. 2009. Software Engineering for Self-Adaptive Systems. Springer-Verlag, Berlin, Heidelberg, Chapter Software Engineering for Self-Adaptive Systems: A Research Roadmap, 1–26.

[7] Thomas Collins et al. 2018. Surprise-based learning of state representations. *Biologically inspired cognitive architectures* 24 (2018), 1–20.

[8] Thomas Joseph Collins et al. 2017. A robust cognitive architecture for learning from surprises. *Biologically Inspired Cognitive Architectures* (2017).

[9] Rogério De Lemos et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*. Springer.

[10] Koller et.al. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[11] Mohammadjavad Faraji. 2016. *Learning with Surprise*. Technical Report. EPFL.

[12] Mohammadjavad Faraji. 2016. Learning with Surprise Theory and Applications.

[13] Mohammadjavad Faraji et al. 2017. Balancing New Against Old Information: The Role of Surprise in Learning. arXiv:1606.05642 [stat.ML]

[14] Robert Feldt and Ana Magazinius. 2010. Validity threats in empirical software engineering research-an initial survey.. In *Seke*. 374–379.

[15] Erik M. Fredericks. 2015. Mitigating Uncertainty at Design Time and Run Time to address Assurance for Dynamically Adaptive Systems. *Michigan S.University. PhD Thesis*. (2015).

[16] Erik M Fredericks et al. 2015. Automated generation of adaptive test plans for self-adaptive systems. In *SEAMS*. IEEE, 157–167.

[17] Luis Garcia et al. 2024. Decision Making for Self-adaptation based on Partially Observable Satisfaction of Non-Functional Requirements. *TAAS* (2024).

[18] Luis Garcia-Paucar and Nelly Bencomo. 2018. Re-STORM: Runtime Non-Functional Requirements trade-off supported by Partially Observable Markov Decision Processes. *SEAMS* (2018).

[19] Omid et.al. Gheibi. 2021. Applying machine learning in self-adaptive systems: A systematic literature review. *TAAS* 15, 3 (2021), 1–37.

[20] Muhammad Usman Iftikhar, Gowri Sankar Ramachandran, Pablo Bollansée, Danny Weyns, and Danny Hughes. 2017. DeltaIoT: A Self-Adaptive Internet of Things Exemplar. In *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 76–82. https://doi.org/10.1109/SEAMS.2017.21

[21] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17 (2015), 184–206.

[22] Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

[23] Emmanuel Letier, David Stefan, and Earl T. Barr. 2014. Uncertainty, risk, and information value in software requirements and architecture. In *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*. ACM Press, Hyderabad, India, 883–894.

[24] Vasiliki Liakoni et al. 2020. An Approximate Bayesian Approach to Surprise-Based Learning. *arXiv:1907.02936 [cs, q-bio, stat]* (Feb. 2020).

[25] Vasiliki Liakoni et al. 2020. Learning in Volatile Environments with the Bayes Factor Surprise. *Neural Computation* (Sept. 2020).

[26] Vasiliki Liakoni et al. 2020. Learning in Volatile Environments with the Bayes Factor Surprise. arXiv:1907.02936 [stat.ML]

[27] Daniel Y Little and Friedrich T Sommer. 2011. Learning in embodied action-perception loops through exploration. *arXiv preprint arXiv:1112.1125* (2011).

[28] Minwen et al. 2003. Seneca: remote mirroring done write.. In *USENIX*.

[29] Alireza Modirshanechi et al. 2022. A taxonomy of surprise definitions. *Journal of Mathematical Psychology* 110 (2022), 102712.

[30] Andres Ramirez et al. 2009. Applying genetic algorithms to decision making in autonomic computing systems. In *ICAC*. 97–106.

[31] Nadeesha Ranasinghe et al. 2009. Surprise-based developmental learning and experimental results on robots. In *2009 IEEE 8th International Conference on Development and Learning*. IEEE.

[32] Nadeesha et.al. Ranasinghe. 2008. Surprise-based learning for developmental robotics. In *2008 ECSIS LAB-RS*. IEEE, 65–70.

[33] Huma Samin et al. 2021. RDMSim: an exemplar for evaluation and comparison of decision-making techniques for self-adaptation. In *SEAMS 2021*.

[34] Huma Samin et al. 2022. Decision-making under uncertainty: be aware of your priorities. *SoSyM* (2022), 1–30.

[35] Pete Sawyer et al. 2010. Requirements-Aware Systems: A Research Agenda for RE for Self-adaptive Systems. In *18th IEEE RE*.

[36] Adhiraj Somani et al. 2013. DESPOT: Online POMDP planning with regularization. *NeurIPS* 26 (2013).

[37] Spaan et al. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of AI research* 24 (2005).

[38] Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. 2011. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *Artificial General Intelligence: 4th International Conference, AGI 2011, Mountain View, CA, USA, August 3-6, 2011. Proceedings 4*. Springer, 41–51.

[39] Dylan Walton et al. 2025. Surprise-based Learning Approach Results. https://github.com/TheSequel02/SurprisebasedLearning.git

[40] K Welsh et al. 2011. Towards requirements aware systems: Run-time resolution of design-time assumptions. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. 560–563.

[41] Kristopher Welsh and Sawyer Pete. 2009. Requirements tracing to support change in dynamically adaptive systems. In *REFSQ*. Springer, 59–73.

[42] He A Xu et al. 2021. Novelty is not surprise: Human exploratory and adaptive behavior in sequential decision-making. *PLOS Computational Biology* 17, 6 (2021).