Reducing Uncertainty in the Decision-Making Process for Self-Adaptive Systems by Accounting for Tactic Volatility

XXXXXXX
XXXXXX
XXXXXX
XXXXXX
Email: {XXXX}@XXX.XXX

***Abstract*—Self-adaptive systems use *tactics* to respond to changes in their environment. Tactics enable self-adaptive systems to remain effective, resilient and secure in the face of change. However, these tactics frequently have volatile attributes that can affect their predictability, dependability, and produced benefit. This creates uncertainty in the decision-making process that may adversely impact the system's ability to make the most optimal decision. Unfortunately, state of the art self-adaptive processes do not have any method of accounting for this volatility. To reduce these levels of uncertainty, we developed a new Tactic Volatility Aware (TVA) solution. TVA incorporates previously observed tactic volatility into the decision-making process, while also using Bayesian Ridge Regression to estimate tactic latency, which helps increase decision-making effectiveness and resiliency. We embedded TVA into the popular MAPE-K adaptation control loop, enabling it to be utilized in a variety of existing self-adaptive systems. The results of our experiments indicate that TVA is able to effectively increase the predictability and dependability of tactic execution, leading to a substantial reduction in uncertainty in the decision-making process. We also present our publicly available web-based Intelligent Simulation Service (IS3) tool. This provides researchers with both a real-world dataset that contains tactic volatility, along with an easily adaptable way to evaluate existing and self-created utility equations.**

***Index Terms*—Self-adaptive systems, Uncertainty reduction, Tactic Volatility**

I. Introduction

Self-adaptive systems have the ability to alter their configuration and functionality to react to changing internal or external events, all while continuing to fulfill requirements and maximizing the *utility* (benefit) of the system [35], [37], [40]. A challenging decision for these systems is how they should react to a nearly infinite number of possible scenarios. Decision-making processes frequently use *tactics* to alter the structure or system properties to respond to these changing situations [28]. An example tactic could be provisioning a virtual machine in a web farm when the workload is about to reach a specific threshold, or activating the deicing mechanism on a UAV when defined environmental conditions are encountered. Regardless of the tactic, they possess attributes such as the time required for execution and the tactic's dependability, all of which impact the predictability of the tactic. We define this variability as *tactic volatility*.

Tactic volatility can adversely impact the quality of a system's decisions-making process since self-adaptive systems rely upon accurate information to make all of their decisions. Inaccurate information, and the inability to properly account for volatility that may occur in real-world systems will very likely lead to less than optimal system outcomes [10], [18], [26], [30].

Unfortunately, state-of-the-art decision-making processes do not adequately account for tactic volatility. This is problematic as tactic volatility may frequently occur, and in reality be expected, in real-world systems. For example, if a system observes the provisioning of an extra virtual machine to consistently take longer than expected, and is unable to learn from these experiences, then it will continue to make inaccurate assumptions about the tactic. This could lead to the self-adaptive system continuing to make decisions that lead to less than optimal utility, and will result in high levels of uncertainty for predicting how the system will behave. As described by previous research [10], [18], [26], [30], uncertainty can be highly detrimental to the self-adaptive decision-making process. To address these challenges, a pragmatic approach enabling a self-adaptive system to account for tactic volatility and learn from previous experiences should be implemented. Moreover, the realization of this approach should not require substantial changes to existing adaptation control loops and should minimally impact the current operations necessary by self-adaptive systems.

This paper contributes a novel approach for accounting for tactic volatility, and a public web-hosted simulation tool called IS3 [2]. Unlike previous adaptation processes, our *Tactic Volatility Aware* (TVA) process incorporates real-world tactic variability into the self-adaptive decision-making process. TVA enables the self-adaptive system to learn from previously experienced tactic volatility, using machine learning to make estimates of how a tactic will behave in varying environments. We have integrated TVA into the popular MAPE-K [25] adaptation control loop, enabling it to positivity impact a large

variety of existing self-adaptive systems. TVA has demonstrated the ability to substantially reduce uncertainty during the self-adaptive process, thus enabling the system to make more informed decisions that lead to more optimal outcomes.

Our experiments indicate that TVA can substantially reduce the amount of uncertainty in the decision-making process by using a Bayesian machine learning approach to estimate tactic latency. To summarize, our work makes the following contributions:

- *Approach:* To the best of our knowledge, our TVA approach is the first which incorporates tactic volatility into the self-adaptive process.
- *Experiments:* An analysis of our approach using several simulated environments to demonstrate the adverse impact of not accounting for tactic volatility, along with TVA's effectiveness in assisting the self-adaptive process.
- *Simulation tool:* Our publicly available IS3 tool [2] enables the evaluation of tactic volatility aware process and provides other researchers a robust, open source dataset that contains uncertainty and tactic latency.

The rest of the paper is organized as follows. Section II motivates our research using examples of tactic volatility. Section III formally defines our TVA solution to effectively address tactic volatility. Section IV describes our systematic experimental evaluation of our proposed process. Section V describes IS3, our public simulation and data creation tool. Section VI describes related works while Section VII describes threats and future work. Section VIII concludes our work.

II. Problem Definition

To motivate the research and demonstrate the need for accounting for tactic volatility, we describe two types of tactic volatility that can impact real-world self-adaptive systems.

A. *Tactic Latency*

The amount of time required to implement a tactic is known as *tactic latency*. Due to innumerable unforeseen circumstances, these latency times can be highly volatile. For example, the precise amount of time a system needs to access data across a network, or a UAV needs to acquire a target will be constantly fluctuating. Self-adaptive systems must account for this tactic latency as accurately as possible as it can have a large impact on the decisions the system should make to achieve the highest utility. Previous works have shown that latency aware algorithms offer several advantages compared to those that do not consider latency [11], [19], [28]. Unfortunately, existing work considers tactic latency to be a predetermined, static value. They do not enable the system to learn from its observations to incorporate more accurate information regarding tactic latency into its decision-making process.

Example - Latency Volatility in UAV: A UAV may have the the primary objective of locating and photographing a specific target. When the UAV identifies the possible target, several steps may need to be taken. For this example, we will assume that the UAV needs to prepare the camera for taking photos and will need to communicate with the base station while performing other necessary calculations (planned time of .2 seconds). This anticipated latency time means that the UAV should plan on beginning the photography process .2 seconds before it is within range of the possible target. However, in the event of poor weather conditions or communication disruptions, it could actually take the UAV .4 seconds to communicate with the base station to receive the required data. In this scenario, it would result in the UAV missing its window for photographing the target. If the UAV was able to use previous observations about its communication lag, it would enable the UAV to account for tactic latency volatility, further improving the decision-making process for the next time adaptation is required. This example scenario is shown in Figure 1.

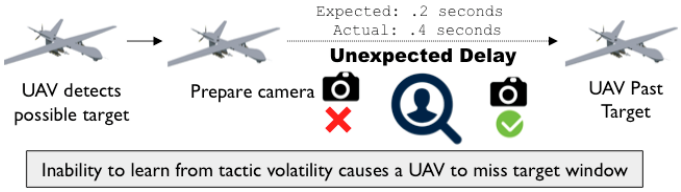

Figure 1. Impact of Unexpected Tactic Latency in UAV

B. *Tactic Dependability*

In a self-adaptive system certain tactics may be more dependable than others. Due to numerous reasons, tactics may sometimes fail to produce their expected results. An example is a physical component on a UAV only intermittently working as expected, or a remote resource only being being sporadically available. We define the measure of how well a tactic correctly produces its expected utility as *tactic dependability*. Accounting for tactic dependability is crucial for making decisions that lead to the highest-utility while completing system critical operations. A system that frequently selects tactics with low dependability may consume extra resources while attempting to attain the desired results from the tactic, or by not completing system-critical operations in the required time frame.

Example - Dependability Volatility in Cloud: A cloud-based system may have three remote resources with identical costs, but with various amounts of produced utility and levels of dependability. If the dependability of each tactic is the same, then selecting the most appropriate tactic is fairly simple; select the one with the highest estimated utility. However, as shown in Figure 2, when the dependability and utility of each possible tactic differs, the system must have a method to account for these variations to determine which tactic is most appropriate. Not accounting for dependability will likely adversely impact the cloud's ability to perform tactic operations that lead to optimal utility.

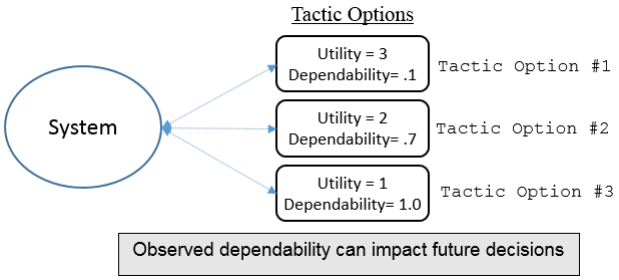

Figure 2. Impact of Dependability Uncertainty

Problems with not accounting for tactic volatility

Self-adaptive systems must be able to accurately account for likely future events and needs [28]. This will enable the system to choose the most proper actions that lead to the most optimal outcome, while also minimizing any unnecessary risk. In many situations, the system will only have enough time to implement a single tactic to address a challenge. This means that the system will not be able to recover from an unsuccessful tactic execution, which could lead to possible system failure or not completing mission/system critical operations. Therefore, attempting to implement tactics with low dependability or low predictability will adversely impact the system's performance, leading to consistently high levels of uncertainty.

III. Proposed TVA Process

In this section, we describe our TVA process in three phases: I) Incorporating the MAPE-K adaptation control loop II) Estimating tactic latency using Bayesian Ridge Regression (BRR) III) Using a utility equation that effectively accounts for tactic volatility, thus leading to higher realized utility.

A. MAPE-K Adaptation Control Loop

We developed our TVA solution to be easily integrated into the popular MAPE-K adaptation control loop [25], enabling easy, widespread adoption. We will next describe how TVA fits into each component of the MAPE-K control loop.

Knowledge Base: For the system to determine if there is a need for adaptation, it needs information that can supplement the decision-making process. In the MAPE-K control loop, this information is stored in the *knowledge base* that is accessible throughout the loop. Thus, it is crucial for the system to have access to this information when determining a need for adaptation. TVA utilizes the knowledge base to provide the necessary data for use in the decision-making process.

Monitor: Self-adaptive systems have the ability to adapt because they are fully aware of their surrounding environments, which in turn gives them the ability to determine when and which adaptations are needed. Therefore, within the monitor phase, data from the knowledge base is monitored so the self-adaptive system can identify when adaptation may be needed.

Analyze and Plan: Traditionally in the MAPE-K loop, the *Analyze* and *Plan* phases would be considered two separate stages. However, since the system must create a plan for adaptation based on the results from the Analyze phase, we combined the two phases in our TVA approach. We did this because we felt the two phases were too dependent on each other to keep them as separate entities in the adaption process.

Within in the Analyze phase of the MAPE-K loop is where our TVA approach makes estimates of utility for each tactic. When finished, TVA takes these estimates and passes them along to the Plan stage. From here, within the Plan stage, our TVA approach simply selects the tactic that is estimated to return the highest utility as the adaptation plan for the system.

Execute: The final phase of MAPE-K loop is where the system brings together all the information and planning it has conducted in the previous phases. Here, the system finally executes the plan from the previous phase that has been estimated to return the highest level of utility to the system.

B. Bayesian Ridge Regression

Bayesian regression is one of the most popular approaches to statistical inferences. Unlike many of the classical regression methods that generate a point estimate, Bayesian processing can produce a predictive distribution, which quantifies the uncertainty of the prediction. This is accomplished by using the classic Bayes' Theorem to update the probability of a hypothesis as more evidence, i.e. data, becomes available to the model. Let's consider a classic linear regression model:

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j x^j = w'x \tag{1}$$

where $x = (x^1, ..., x^M)'$ with $x^0 = 1$ and x^j being the j-th feature of the data point. Given a set of N training data points $(x_1, ..., x_N)$ along with the observed responses $(t_1, ..., t_N)$, we can solve for the best possible weights of this model through minimizing the error function

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\} \tag{2}$$

$E(w)$ is a quadratic function of w, which can be conveniently minmized, leading to

$$w^* = (X'X)^{-1}X't \tag{3}$$

where X is the design matrix whose rows correspond to the feature vectors of the data points and $t = (t_1, ..., t_N)'$ denotes the observed response values of N data points.

The same solution can be achieved by assuming that each response t follows a Gaussian distribution:

$$p(t|x, w, \beta) = \mathcal{N}(t|y(x, w), \beta^{-1}) \tag{4}$$

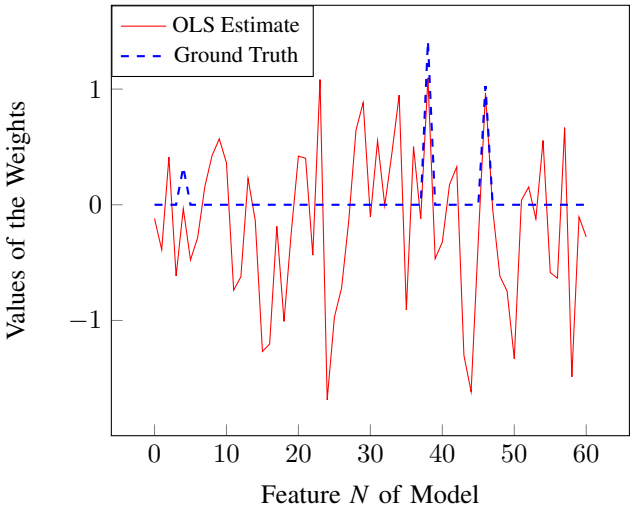

Figure 3. OLS Feature Weight Estimation vs Ground Truth

where β is the precision of the Gaussian. By further assuming that each data point is independent, the joint likelihood of all the data points is given by

$$p(t_1, ..., t_N|X, w, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, w), \beta^{-1}) \quad (5)$$

By maximizing the likelihood term or its logarithm as shown above, it can be seen that it is equivalent to minimizing the error function $E(w)$. Such an approach is commonly known as Maximum Likelihood Estimation (MLE).

However, since the estimate of w^* is a single data point and not a distribution, we cannot quantify our uncertainty around the estimate. More importantly, MLE will most likely be overfit to the data especially for high dimensional problems (i.e., M is large), meaning the predicted value only fits the training data well and can not be generalized to new and previously unseen data points. Bayesian Regression addresses this issue by introducing a prior distribution over w:

$$p(w|\lambda) = N(w|0, \lambda^{-1} I_M) \quad (6)$$

where I_M is an identify matrix of dimensionality M and λ is the precision parameter of the prior. Given the prior and the likelihood term and using the Bayes' theorem, we can derive the posterior distribution for w:

$$p(w|X, t_1, ..., t_n, \beta, \lambda) \propto p(w|\lambda) p(t_1, ..., t_N|X, w, \beta) \quad (7)$$

The most probable w given the data can be estimated by maximizing $p(w|X, t_1, ..., t_n, \beta, \lambda)$, a technique referred to as *maximum posterior or MAP*. Solving MAP is equivalent to minimizing the following term

$$\frac{\beta}{2} E(w) + \frac{\lambda}{2} ||w||^2 \quad (8)$$

This has the same form as ridge regression, sometimes known as *Tikhonov Regularization*, which is one of the most commonly used methods of regularization in statistics, specifically for *ill-posed* optimization problems.

The term 'regularization' refers to minimizing the weights of a model, or forcing them to be as close to 0 as possible. Figure 3 represents the issue of over-fitting that often occurs with Ordinary Least Squares (OLS). As shown, the estimated weights of the model with OLS are quite far off in both the positive and negative directions from 0, indicating high amounts of over-fitting. By forcing the weights of less important features to be close to zero, regularized models can help effectively address the over fitting issue.

While MAP or regularization helps model over fitting, it still only provides a point estimate, which lacks the capability to quantifyyeah the uncertainty of the estimation. A full Bayesian treatment addresses this by producing a *predictive distribution* for a new data point x to be estimated

$$p(t|x, X, t) = \int p(t|x, w) p(w|X, t_1, ..., t_N) dw \quad (9)$$

where $p(t|x, w)$ and $p(w|X, t_1, ..., t_N)$ are defined by (4) and (7), respectively. Due to the nice property of a Gaussian distribution, the integration in (9) can be analytically computed, resulting in another Gaussian $p(t|x, X, t) = \mathcal{N}(t|m(x), \gamma(x))$, where $m(x)$ and $\gamma(x)$ are the mean and variance, respectively. In this way, the mean will be used for prediction and the variance is used to quantify the uncertainty of the prediction.

C. Proposed Utility Equation

When a self-adaptive system decides which tactic is the most appropriate for adaptation, it selects the one that is expected to return the highest level of *utility* or benefit to the system. Therefore, being able to accurately estimate what each tactic can return to the system is crucial for the decision-making process. In this section, we describe our proposed utility function that can be used in a self-adaptive system to guide the adaptation process:

$$U = \frac{T - \sum_{i=1}^{N} L_i}{SD} * P \quad (10)$$

The key terms of our proposed utility equation are explained in detail below.

Maximum Defined Threshold: In various types of self-adaptive systems, specific tactics may need to be executed in a specific amount of time. Otherwise, the system may be adversely impacted. An example is a system taking longer than tolerated to restrict access to a sensitive component when a security intrusion is detected, thus leading to the malicious actor having access to the sensitive data. Another example is the system taking longer than tolerated to activate additional system resources when facing a heavy workload, thus possibly resulting in a system failure. Depending on the tactic and scenario, the system's resiliency, security and overall effectiveness can be severely impacted whenever this maximum tolerated latency is surpassed. We define this maximum tolerated time as the *Maximum Defined Threshold*, T.

Latency Sum Estimate: Regardless whether a self-adaptive system is performing single or sequential tactic execution, capturing the sum of the estimated latencies provides TVA with the overall estimated latency time. For example, if Tactic A is made up of only itself, then the latency sum will simply be from $i = 1$ to 1 of L_i, where L_1 is Tactic A and the sum is simply the estimate of Tactic $A's$ latency. However, if Tactic A is made up of a sequence of tactics, then the latency sum is simply from $i = 1$ to n where n is the total number of tactics in the sequence and L_i is the latency estimate of the i^{th} tactic in the sequence.

Latency Distribution Variability: Incorporating latency volatility into the decision-making process also means accounting for variability within experienced latency times. To do this, we have defined SD as the standard deviation of the latency distribution for the current tactic being analyzed in the decision-making process. By including the standard deviation of the latency distribution in the denominator, we used the process known as standardization. This allows us to penalize tactics who have sporadic latency times, which causes higher levels of uncertainty in predicting their behavior. Furthermore, this enables us to conform our utility values to a common standard, which is a must for acquiring resulting utility values that can be appropriately compared.

Percent Chance of Success: Self-adaptive systems may encounter situations where only a portion of tactics in a sequence are estimated to complete in time. To account for this challenge, P was included in our utility equation to represent the percentage of tactics that have been estimated to complete successfully. For example, holding all other variables in our utility equation the same, the sequence of tactics that is estimated to complete 75% of it's underlying sub-tactics is a more optimal decision than one with 25%.

IV. Evaluation

To evaluate TVA, we used the simulation tools R, IS3, and SWIM [6] to address and answer the following research questions:

RQ 1. *Impact:* How does not accounting for tactic volatility adversely impact the the decision-making process for a self-adaptive system?

RQ 2. *Predictability:* How effective is TVA in improving the predictability of tactic latency time?

RQ 3. *Dependability:* Is TVA effective at increasing the dependability of the self-adaptive system?

Experimental Design

For the research questions addressed in our work, we used the simulation tools R, IS3 and SWIM. R was used as an initial proof of concept source. SWIM was used because it was built to simulate a self-adaptive web architecture, therefore no modifications were needed in order to make the tool self-adaptive. Lastly, IS3 was used because it provided us with real-world data. Therefore, some factors that might naturally affect tactic latency, could show up in our dataset, Thus, this makes the IS3 dataset a more robust collection of empirical latency times, rather than simulated latency times.

For our simulations in R, we generated data distributions for tactic latency times to show the impact of not accounting for tactic volatility. In these simulations, we defined two tactics that had different latency distributions: positively skewed and normal. By using two divergent values, we are able to demonstrate the impact of not accounting for tactic volatility in a perfect environment (the normal distribution) and in a volatile environment (the positively skewed distribution).

When using the simulation tool IS3, we required data containing latency and dependability volatility. This was created by recording the time necessary to continuously download an identical 75MB file from three different download mirrors from the US, Canada, and Germany to a server located in the United States. We presently have $> 50,000$ latency and dependability values in this growing publicly available dataset. Table I shows the minimum, maximum, and average latency time for each download mirror location, along with the standard deviation and number of downloads.

Table I
Mirror Download Latency(ms) Statistics from IS3 Tool

	Min	Max	Avg	SD	# Downloads
USA	881	15,855	1,757	1,093	16,873
Canada	910	93,502	2,097	2,551	16,874
Germany	3,997	772,149	18,056	27,829	16,865

Figure 4 represents a portion of the time series data gathered from our simulations in IS3, and helps to visualize the standard deviations reported in Table I. The latency times gathered from Germany were quite volatile, which was essential for our evaluation because it enabled us to stress test our process and further evaluate the robustness of our TVA approach.

In our final set of simulations that addressed tactic latency, we used the three tactics that are made available to the user in SWIM: adding a server, removing a server, and setting the dimmer value. To construct our training set, we performed 420 simulations that measured how long it took for each tactic to complete once a need for adaptation was recognized. We then performed another 420 simulations to use as our test set. Running this many simulations provided us with enough data to build regression models without causing any over fitting issues, a common mistake made during the data collection process. It is imperative to balance the amount of data you collect, because with too much data significance tests will too frequently report significant relationships.

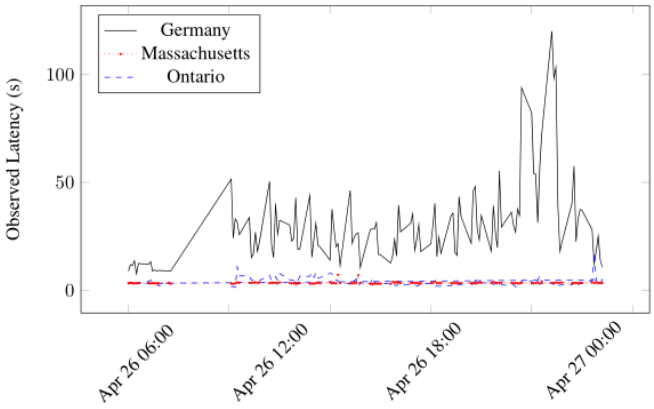

Figure 4. One-Day Snippet of Latency Time Gathered from Mirror Requests

Experimental Data Analysis

To analyze the results found from our work, we chose several statistical tests to evaluate if our TVA approach can significantly reduce uncertainty and increase the effectiveness of the decision-making process for self adaptive systems. The methods used are discussed in this section.

Mann-Whitney U: To test for statistically significant differences between our TVA approach and the ground truth, we used the Mann Whitney U (MWU) significance test. MWU is a non-parametric test that states, "it is equally likely that a randomly selected value from one sample will be less than or greater than a randomly selected value from a second sample". This was more appropriate for our analysis because MWU allows for more robust analysis in detecting significant differences between non-normally distributed data, without sacrificing the ability to detect significant differences between normally distributed data.

To determine significance, we set our α level to 0.05. However, unlike the traditional use of α where we aim to have significant p-values returned, we are interested in seeing *non-significant* results for our analysis. This is because we don't want the differences between feature weight estimations in TVA and the ground truth to be significantly different. If they were, this means our TVA process utterly failed to predict tactic latency. Therefore, we aim to have our statistical tests of significance return p-values greater than our α level.

Cliff's δ: Determining significant differences between two groups, although important, only allows us to answer the question of *"Are these groups different?"*. However, it is equally important to analyze the question, *"How much different?"*. To do this, we chose to use the non-parametric effect size statistic *Cliff's* δ. The resulting effect size values are classified as follows: negligible ($\delta \leq 0.147$), small ($0.147 < \delta \leq 0.33$), medium ($0.33 < \delta \leq 0.474$), and large ($0.474 \geq \delta$).

Root Mean Squared Error: To analyze and further quantify the results found from our simulation environments, we used the popular Root Mean Squared Error (RMSE) statistic as a measure of predictive power.

$$RMSE = [\sum_{i=1}^{N}(z_i - z_o)^2/N]^{1/2} \tag{11}$$

As shown in Equation 11, RMSE represents the quadratic mean of the differences between the estimated values and observed values, or the *residuals*, across multiple models. It serves to aggregate the magnitudes of prediction errors, so the resulting values can be used as a measure of predictive power. Therefore, this means that since RMSE is a measure of accuracy, it can concretely tell us how models compare to each other in terms of predictive ability.

RQ1: Impact

We first explored how not accounting for tactic volatility can negatively impact the self-adaptive system. This was accomplished by performing a proof of concept evaluation using the statistical tool R. We began by defining two tactics that had associated costs with their latency times. These "costs" are an arbitrary representation of resources that a self-adaptive system may consume during the execution of a tactic. For example, the cost of executing a tactic may be the consumption of 3 GB of RAM, which may slow the other processes running in the system.

Table II
CHARACTERISTICS OF SAMPLE TACTICS FOR PROOF OF CONCEPT SIMULATION

	Cost	Distribution	Average	SD
Tactic A	5	Positively Skewed	3	0.5
Tactic B	7	Normal	3	0.5

We next generated a normal distribution and positively skewed distribution with the same means and standard deviations to represent latency times. We gave the tactic with the higher cost the normal distribution, while tactic with the lower cost was given the positively skewed distribution. Table II shows the characteristics of each tactic.

After generating the characteristics of each tactic, we then performed 100 simulations of random sampling from each tactic and multiplied the sampled latency value by its associated cost. This enabled us to build a distribution of overall cost values that could be used to compare how varying data distributions can effect the predictability of the cost for executing a tactic without accounting for tactic volatility.

In our analysis, we found that tactic B may not have had the lowest cost values in the simulations, but it was significantly more consistent, represented in Figure 5. However, tactic B's data followed a normal distribution, which is highly unlikely occurrence for real-world systems. Furthermore, even though tactic A had a lower cost of execution with a value of five, it resulted in much more sporadic, and extremely high, overall costs to the system.

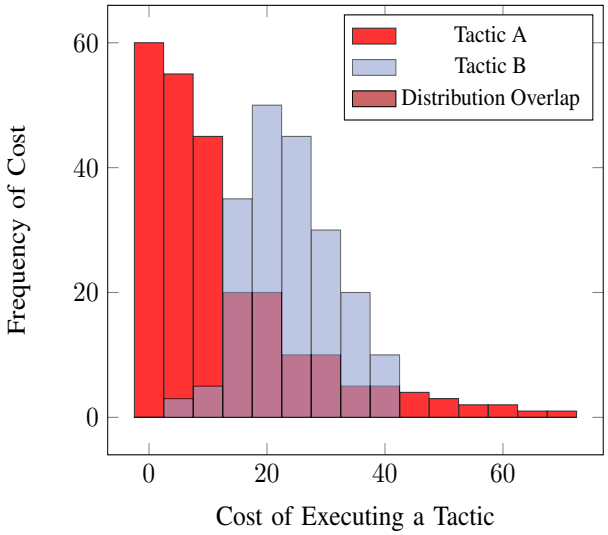

Figure 5. Overall Costs of Executing a Tactic to Demonstrate the Benefits of Accounting for Tactic Volatility

Outcome: *Our proof of concept simulations in R were able to successfully demonstrate that accounting for tactic volatility is essential in self-adaptive systems, especially when the system is known to have unpredictable behavior.*

RQ2: Predictability

Unpredictability is intuitively considered to be an undesirable trait of a self-adaptive system and is frequently associated with much of the uncertainty that surrounds self-adaptive systems [18], [26], [36], [39]. To determine how well TVA can improve the predictability of a self-adaptive system by better estimating latency times, we examined how well the ground truth feature weights were estimated in with our BRR model. The closer we are to the ground truth, the better the model is able to predict latency, thus reducing uncertainty. To quantify how well TVA can improve this this, we examined the RMSE values generated from each of the simulation environments used in this research question and the overlapping fit between the ground truth feature weights and the model generated weights.

As demonstrated in both Figure 6 and Figure 7, we found that using Bayesian Ridge Regression performed extremely well for estimating the ground truth feature weights of the model. We saw the most significant performance from our model when the ground truth feature weights began to deviate away from zero, whether it be small or large deviations. Furthermore, being able to better estimate feature weights that deviated away from zero did not hinder the model's ability to estimate other feature weights. The average differences between feature weight estimation in the BRR model and the ground truth for our SWIM simulations was 0.0054, while the difference in our IS3 simulations was -0.0145. The closer these differences were to zero, the better "overall performance" from our model. Had we been able to accomplish an average difference of zero, then the sums of all our differences between estimated and observed values would have been zero, indicating an "overall" perfect performance from them model.

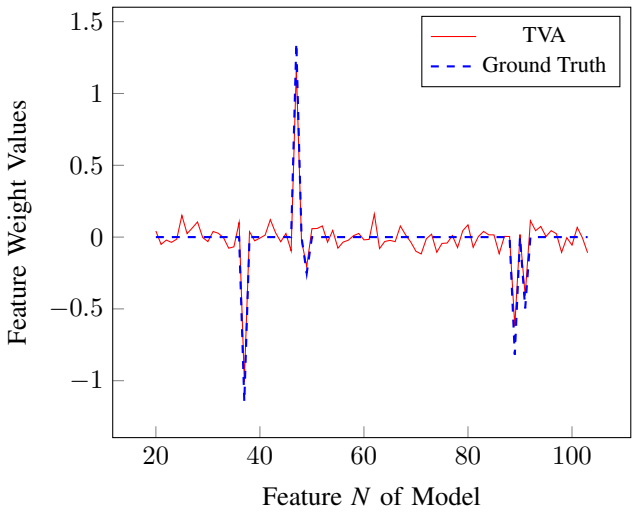

Figure 6. TVA Feature Weight Estimation VS Ground Truth in IS3

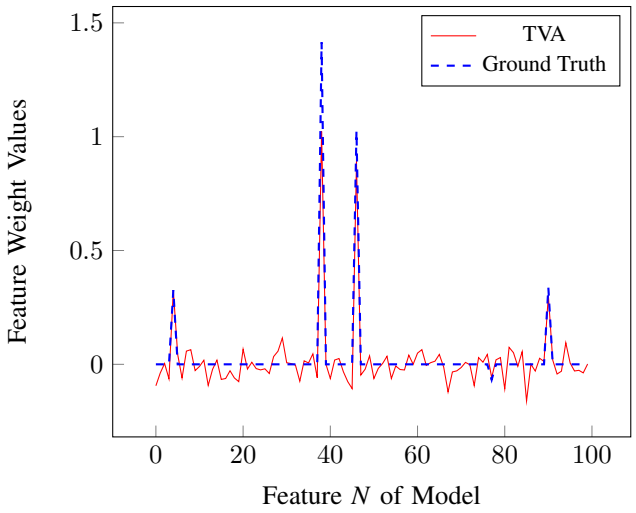

Figure 7. TVA Feature Weight Estimation VS Ground Truth in SWIM

Table III
FEATURE WEIGHT ESTIMATION - MWU RESULTS

	p-value	**Cliffs δ**
IS3	0.8533	0.014
SWIM	0.6779	0.033

Table III displays the results from our MWU significance tests and Cliff's δ calculations from the IS3 and SWIM simulations. As shown, both the IS3 and SWIM simulations reported non-statistically significant differences between feature weights in our generated model and the ground truth feature weights. This does not mean that the feature weights in our model were the same as the ground truth feature weights. Rather,

it means the differences can be attributed to random chance and that there is no significant relationship between the two. The observed Cliff's δ results also supported these findings by reporting negligible effect sizes.

Outcome: *Our simulations demonstrated that TVA is able to successfully reduce the amount of uncertainty in the decision-making process through improved predictability of tactic execution.*

RQ3: Dependability

To determine how our TVA approach can improve the dependability of a self-adaptive system and its decision-making process, we examined how our approach performed over numerous simulations of adaptation decisions. By performing these simulations with varying data sets, we can verify that no matter the distribution of the data or how volatile it may be, our approach can handle this appropriately. This helped us to emulate real-world data sets, especially those with high amounts of s variability.

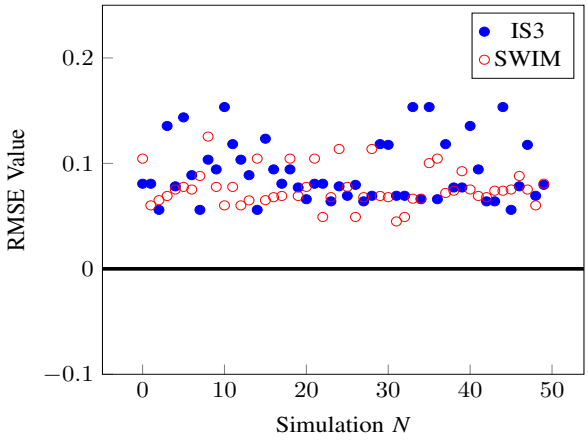

Figure 8. RMSE Values Over 50 Independent Simulations Using TVA

Figure 8 displays the results from the first portion of our stability simulations. In this analysis, we ran 50 cycles of simulations using both IS3 and SWIM. For example, in our SWIM simulations, we considered a simulation to be a run of 420 adaptations before calculating RMSE. Therefore, by doing 50 cycles of 420 adaptations, we ended up with more than 5,000 adaptation decisions to use for our stability analysis.

Table IV
RMSE Statistics Over 50 Independent Simulations

	Min	Max	Avg	SD
IS3	0.0559	0.1534	0.0911	0.0629
SWIM	0.0457	0.1254	0.0766	0.0315

In our analysis we found that TVA was quite stable for both our IS3 and SWIM simulations. To evaluate stability, we examined how much the calculated RMSE values deviated away from zero. We did this because if you could perfectly estimate the value of some unknown variable, i.e. the differences between the estimated value and observed value are all zero, you would end up with an RMSE of zero. Therefore, the horizontal line at value zero in Figure 8 represents a perfect RMSE score.

Table IV shows just how stable TVA was across all of the simulations. In the 100 total simulation cycles run between IS3 and SWIM, the highest RMSE value was only 0.1534, while the lowest was considerably low at 0.0457. Additionally, the distribution of RMSE values was also tightly bound around the mean, as the standard deviation for both simulation tools was also considerably low. Therefore, TVA was able to considerably maintain it's predictive power, demonstrated by the low RMSE values, even over the course of many simulations. This was particularly impressive as our simulations in IS3 had significant volatility in downloading the 75 MB file from the Germany mirror, as shown in Figure 4.

The second portion of our stability simulations examined how frequently our TVA process was able to return *at least* the expected utility to the system. Ideally, we would like to see TVA returning the exact amount of utility that was expected because this would mean TVA is perfect in predicting latency, therefore completely eliminating uncertainty. However, since perfect prediction accuracy is improbable, we felt it was necessary to ensure that TVA was not over estimating the amount of returned benefit to the system as this could lead to decisions that result in unexpected resource consumption. For this analysis, we used the same data collected from the first portion of our stability simulations.

Our results from this analysis were extremely promising, as TVA was able to return at least the expected benefit of executing a tactic 90% of the time in our SWIM simulations, and 86% of the time in our IS3 simulations. The slightly lower successful return rate for the IS3 simulations came at no surprise, as its data had much higher variability than the data collected from SWIM.

Outcome: *The RMSE and returned benefit results gathered from addressing this research question have demonstrated that TVA is very effective in improving the dependability of a self-adaptive system. TVA is able to do this by providing consistently stable predictive power, even in the presence of volatile data, thus increasing the dependability of the system and it's decision-making processes.*

V. Intelligent Simulation Service Tool

Our contributions include the publicly available *Intelligent Simulation Service* [2] (IS3) web-based tool that I) Generates datasets from real-world events that contain variable tactic latency and dependability information to emulate uncertainty II) Enables researchers to quickly and easily evaluate utility equations against one another using existing and user provided datasets. IS3 was created as researchers are limited by the

availability of existing robust simulation tools and datasets to aid in the evaluation of decision-making mechanisms.

While there are several simulation tools and datasets available [1], [3]–[7], they I) Do not consider real-world events such as latency and dependability uncertainty II) Are typically not easily extensible III) Are not free, open source tools that are easily ready to sufficiently provide the necessary data to conduct the desired simulations. This lack of existing, easy to adopt, robust simulation tools not only make the evaluation of potential decision-making mechanisms much more difficult, but can also lead to inaccurate findings due to the 'one-off' nature of self-created simulation tools and datasets. Additionally, using simulation tools and data that do not accurately represent real world events, such as latency, can lead to inaccurate/incorrect conclusions; ones that do not properly account for tactic volatility. Our IS3 tool was systematically developed to address current gaps in emulating uncertainty. Developing such a tool was crucial for the successful development and evaluation of our TVA process.

Enabled Research IS3 [2] provides support for researchers in the self-adaptive systems field. IS3 enables the user to evaluate existing and custom utility equations against one another using the hosted, web-based tool. The objective is for IS3 to become an important tool in developing and evaluating self-adaptive equations by providing researchers the opportunity to easily evaluate their own utility equations. The public dataset containing latency will provide other researchers an important resource for evaluating their own latency-aware equations, through either using the IS3 tool or in other simulation environments. Accounting for tactic latency is an emerging concept in self-adaptive systems [8], [28], [31], [33], [34], and we firmly believe that tactic volatility will become a more examined topic as well. Our tool and dataset would prove to be a valuable asset to others exploring this area.

VI. Related Works

In this section we provide a discussion of the related works to our research.

Addressing Uncertainty in Self-Adaptive Systems Many works have discussed how uncertainty can be detrimental to the self-adaption process and how it can lead to decisions with less than optimal utility [9], [12], [21]. Moreno et al. [32] addressed uncertainty in self-adaptive systems by helping to reduce the run-time overhead of the MDP process by constructing most of it offline. However, this work did not address the volatility of tactic latency. Mahdavi-Hezavehiet al. [27] conducted work on uncertainty in self-adaptive systems, classifying five dimensions of uncertainty: location, source, nature, level/spectrum and emerging time. De Lemoset al. [16] described three main categories of regarding requirements, design and run-time uncertainties. Uncertainty has been shown to affect virtually every phase of the MAPE-K loop [16]. For example, imperfections in sensors can affect the monitoring phase, while the execution phase can be impacted by effector reliability. Weyns et al. [40] described sources of uncertainty in systems. They described sources of uncertainty related to the system, the goals of the system, context execution and human related aspects. The main focus of this work was to present perpetual assurance cases for self-adaptive systems. Jamshidiet al. [24] proposed dealing with uncertainty using a fuzzy control theory combined with machine learning.

Cámara et al. [12] created a formal analysis technique that considers uncertainty during the determination process for the most appropriate adaptation decisions. This technique focused on aleatoric (random) uncertainties created by inaccurate sensor readings. They found that uncertainty-aware adaptation processes do not always provide better performance in comparison with techniques that do not account for uncertainty. However, uncertainty-aware techniques were found to perform better in boundary regions and at least comparable in all other circumstances. Esfahani et al. [18] examined sources of uncertainty and discussed the impacts of uncertainty on the ability of the system to meet defined goals. Some sources of defined uncertainty include 'simplified assumptions', 'model drift', 'noise', and 'humans in the loop'.

Fredericks [22] used two search-based techniques 'Ragnorok' and 'Valkyrie' for hardening self-adaptive systems against uncertainty. This work found that these techniques can improve both the design and implementation of self-adaptive systems. Esfahani et al. [20] created a method known as "PossIbilistic SElf-aDaptation" (POISED) to address challenges created by uncertainty during the self-adaptation process. POISED used a 'POssIbilistic' process to determine positive and negative implications of uncertainty in its analysis. POISED was evaluated on a prototype of a robotic software system.

Moreno et al. [29] focused on reducing uncertainty in self-adaptive systems through the use of *uncertainty reduction tactics*. Examples of such tactics include sending a request to a web server that has not recently been used to attain information about its response time and availability [23]. While this work is promising, our work differs in that our process does not implement proactive uncertainty reduction tactics into the adaptation process. TVA improves the decision-making process exclusively through learning from prior experiences. TVA does not force the system to take any new, or different actions to reduce uncertainty. In many cases, more proactive approaches are not reasonable to implement. For example, in a UAV it may not be reasonable to test the deicing mechanism to reduce uncertainty due to the power consumption of such an uncertainty reduction tactic.

Research has examined tactic latency during the adaptation process. Cámara et al. [11] was likely the first work to consider tactic latency, as it discussed how latency consideration could be used to assist the proactive adaptation process. Moreno [28], [33] discussed the need for a Proactive-Latency Aware (PLA) [11], [30] self-adaptation approaches that integrate timing considerations into the decision-making

process. In addition to supporting pro-activeness and concurrent tactic execution, PLA also recognized latency awareness. This approach considers the amount of time taken for tactics to execute, to account for both the delay before their utility can be realized and to avoid situations that are unachievable when time dimensions are recognized. Moreno [28] presented three latency aware approaches PLA-PMC, PLA-SDP and SB-PLA, which are both tactic based approaches. This work found that including tactic latency was beneficial to the proactive adaption process. Although previous work has demonstrated the benefits of considering tactic latency, all consider latency to be a static predetermined value. They do not account for any volatility in the tactic latency in the adaptation process as is done by our TVA process.

Learning Support in Decision-making Several previous works have examined the use of learning to support the decision-making process of self-adaptive systems. Elkhodary [17] combined feature-orientation, learning, and dynamic optimization processes to create a new group of self-adaptive systems that are able to adjust their adaptation logic at run-time. Their FUSION framework automatically learned the impact of feature selection. This learned behavior was used to constantly update the system's adaptation logic. Our work differs from the FUSION framework which focused on understanding the impact of adaptation decisions in terms of feature selection on the system's goals, whereas our TVA approached focused on reducing tactic volatility to reduce system uncertainty. Sykeset al. [38] used probabilistic rule learning [14], [15] to alter the behavioral model using observations of the system and environment. The amended models are next used to guide new system behavior, providing a better opportunity for success when the specific station is encountered.

Cheng et al. [13] proposed a language for self-repair that includes a utility function incorporating the success rate of tactics, and thus the history of applying tactics. While similar to our work in that it incorporates prior history into its utility equation, this work does not incorporate learning mechanisms or address tactic volatility as we have.

VII. Threats and Future Work

Our TVA approach has demonstrated the ability to reduce uncertainty in the decision-making process for self-adaptive systems, leading to more effective decisions. However, there are several limitations to our work. In the initial phases of incorporating our TVA approach, a tactic's latency may still be predetermined due to a lack of prior data to consider. Therefore, until an adequate amount of data can be collected about tactic volatility, values will continue to be pre-determined or a distribution will have to be generated such as the process we used in R. Additionally, observing tactic volatility will not be practical for addressing several types of infrequently utilized tactics. For example, many tactics may be ran only a single time, meaning that traditional processes will need to continue to be used due to a lack of relevant available data.

There may also be instances where all available possible tactics are estimated to complete outside of the maximum defined threshold. Future work will incorporate this possibility into the decision-making process, as there will likely be some overhead in accounting for tactic volatility. However, this cost may not be justifiable by all adaptive systems.

Our TVA approach has demonstrated the importance of using Bayesian Ridge Regression to estimate tactic latency. However, when applied to real-world systems, we may find that the data returned from experiments is too non-normal for Bayesian and its ability to account for it. Therefore, future steps may be necessary to clean the volatility data before applying Bayesian Ridge Regression, or other forms of regression will have to be weaved into the Bayesian process.

Tactic-based decisions frequently do not occur independently from other tactics. Often, tactics occur simultaneously with other tactics and this may impact other tactic-based decisions [28], [30]. Future work should be done to examine how poor tactic-based decisions due to tactic volatility can affect both subsequent and concurrent tactic-based decisions. These results have the possibility of further demonstrating the importance of considering tactic volatility in the decision-making process.

Although we have demonstrated the benefits of our TVA approach in several simulated environments, we have yet to implement it into any physical devices. This means that all of our evaluations were conducted on artificial or largely simulated data. Future work will consist of including our adaptation process into physical equipment such as IoT devices and small UAVs. Our work explored tactic volatility in the form of latency and dependability. Future work could also explore other forms of tactic volatility including availability.

VIII. Conclusion

Self-adaptive systems are fortunately beginning to account for tactic latency in their decision-making processes. However, our work is the first known of its kind to enable the system to learn from experienced tactic volatility and incorporate this into the decision-making process. We demonstrated the effectiveness of our Tactic Volatility Aware (TVA) technique using R, along with the simulation tools IS3 and SWIM. In our work, we I) demonstrated the importance of accounting for tactic volatility in the self-adaptive process II) demonstrated the ability of TVA to increase the predictability of a tactic, leading to reduced levels of uncertainty III) demonstrated that TVA is effective at improving the dependability of a self-adaptive system, even in the presence of high tactic volatility. Our work not only demonstrates that accounting for tactic volatility can be extremely beneficial for the system, but lays the foundation for future work in this area as well.

Acknowledgements

Elements of this work are sponsored by [Hidden].

REFERENCES

[1] Agent-based simulation modeling – anylogic. https://www.anylogic.com/, 2018.

[2] Intelligent system simulation service. http://www.is3tool.com, 2018.

[3] Omnet++ discrete event simulator. https://www.omnetpp.org/, 2018.

[4] Rubis: Rice university bidding system. http://rubis.ow2.org/, 2018.

[5] Seams artifacts. http://drops.dagstuhl.de/opus/institut_darts.php?fakultaet=10, 2018.

[6] Simulator of web infrastructure and management. https://github.com/cps-sei/swim, 2018.

[7] Simulink: Simulation and model based design. https://www.mathworks.com/products/simulink.html, 2018.

[8] Ca. Analyzing latency-aware self-adaptation using stochastic games and simulations.

[9] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic qos management and optimization in service-based systems. *IEEE Transactions on Software Engineering*, 37(3):387–409, 2011.

[10] J. Cámara, D. Garlan, W. G. Kang, W. Peng, and B. Schmerl. Uncertainty in self-adaptive systems. *MONTH*, 2017.

[11] J. Cámara, G. A. Moreno, and D. Garlan. Stochastic game analysis and latency awareness for proactive self-adaptation. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 155–164. ACM, 2014.

[12] J. Cámara, W. Peng, D. Garlan, and B. Schmerl. Reasoning about sensing uncertainty in decision-making for self-adaptation. In *International Conference on Software Engineering and Formal Methods*, pages 523–540. Springer, 2017.

[13] S.-W. Cheng and D. Garlan. Stitch: A language for architecture-based self-adaptation. *J. Syst. Softw.*, 85(12):2860–2875, Dec. 2012.

[14] D. Corapi, A. Russo, and E. Lupu. Inductive Logic Programming as Abductive Search. In M. Hermenegildo and T. Schaub, editors, *Technical Communications of the 26th International Conference on Logic Programming*, volume 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54–63, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[15] D. Corapi, D. Sykes, K. Inoue, and A. Russo. Probabilistic rule learning in nonmonotonic domains. In *Proceedings of the 12th International Conference on Computational Logic in Multi-agent Systems*, CLIMA'11, pages 243–258, Berlin, Heidelberg, 2011. Springer-Verlag.

[16] R. de Lemos, H. Giese, H. A. Müller, and M. Shaw. Software engineering for self-adaptive systems ii. 2010.

[17] A. Elkhodary. A learning-based approach for engineering feature-oriented self-adaptive software systems. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '10, pages 345–348, New York, NY, USA, 2010. ACM.

[18] N. Esfahani. *Management of uncertainty in self-adaptive software*. PhD thesis, George Mason University, 2014.

[19] N. Esfahani, A. Elkhodary, and S. Malek. A learning-based framework for engineering feature-oriented self-adaptive software systems. *IEEE transactions on software engineering*, 39(11):1467–1493, 2013.

[20] N. Esfahani, E. Kouroshfar, and S. Malek. Taming uncertainty in self-adaptive software. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 234–244, New York, NY, USA, 2011. ACM.

[21] N. Esfahani and S. Malek. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*, pages 214–238. Springer, 2013.

[22] E. M. Fredericks. Automatically hardening a self-adaptive system against uncertainty. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '16, pages 16–27, New York, NY, USA, 2016. ACM.

[23] J. Hielscher, R. Kazhamiakin, A. Metzger, and M. Pistore. A framework for proactive self-adaptation of service-based applications based on online testing. In *Proceedings of the 1st European Conference on Towards a Service-Based Internet*, ServiceWave '08, pages 122–133, Berlin, Heidelberg, 2008. Springer-Verlag.

[24] P. Jamshidi, C. Pahl, and N. C. Mendonça. Managing uncertainty in autonomic cloud elasticity controllers. *IEEE Cloud Computing*, 3(3):50–60, May 2016.

[25] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[26] C. Kinneer, Z. Coker, J. Wang, D. Garlan, and C. L. Goues. Managing uncertainty in self-adaptive systems with plan reuse and stochastic search. 2018.

[27] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. A classification of current architecture-based approaches tackling uncertainty in self-adaptive systems with multiple requirements, managing trade-offs in adaptable software architectures. *Managing Trade-offs in Adaptable Software Architectures. Elsevier*, 2016.

[28] G. A. Moreno. Adaptation timing in self-adaptive systems. 2017.

[29] G. A. Moreno, J. Cámara, D. Garlan, and M. Klein. Uncertainty reduction in self-adaptive systems. 2018.

[30] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 1–12. ACM, 2015.

[31] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl. Proactive self-adaptation under uncertainty: A probabilistic model checking approach. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 1–12, New York, NY, USA, 2015. ACM.

[32] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl. Efficient decision-making under uncertainty for proactive self-adaptation. In *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pages 147–156. IEEE, 2016.

[33] G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl. Flexible and efficient decision-making for proactive latency-aware self-adaptation. *ACM Trans. Auton. Adapt. Syst.*, 13(1):3:1–3:36, Apr. 2018.

[34] G. A. Moreno, A. V. Papadopoulos, K. Angelopoulos, J. Cámara, and B. Schmerl. Comparing model-based predictive approaches to self-adaptation: Cobra and pla. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '17, pages 42–53, Piscataway, NJ, USA, 2017. IEEE Press.

[35] G. A. Moreno, O. Strichman, S. Chaki, and R. Vaisman. Decision-making with cross-entropy for self-adaptation. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 90–101. IEEE Press, 2017.

[36] A. Rodrigues, R. D. Caldas, G. N. Rodrigues, T. Vogel, and P. Pelliccione. A learning approach to enhance assurances for real-time self-adaptive systems. *arXiv preprint arXiv:1804.00994*, 2018.

[37] M. Salehie and L. Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, 4(2):14, 2009.

[38] D. Sykes, D. Corapi, J. Magee, J. Kramer, A. Russo, and K. Inoue. Learning revised models for planning in adaptive systems. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 63–71, Piscataway, NJ, USA, 2013. IEEE Press.

[39] S. Tomforde, S. Rudolph, K. Bellman, and R. Würtz. An organic computing perspective on self-improving system interweaving at runtime. In *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pages 276–284. IEEE, 2016.

[40] D. Weyns, N. Bencomo, R. Calinescu, J. Cámara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek, et al. Perpetual assurances in self-adaptive systems. 2017.