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

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

***Abstract*—Self-adaptive systems frequently use *tactics* to respond to changes in their surrounding environments. One example tactic is a web farm provisioning a virtual machine when workloads reach a specific threshold. Tactics enable self-adaptive systems to remain effective, resilient and secure in the face of change. However, these tactics often have volatile attributes that can affect their predictability, dependability, and produced benefit. This creates uncertainty in the decision-making process that may be highly detrimental to the self-adaptive decision-making process. Unfortunately, state of the art self-adaptive processes do not have any method of accounting for this volatility.**

To reduce uncertainty in the self-adaptive process, we developed a new Tactic Volatility Aware (TVA) solution. We embedded TVA into the widely used MAPE-K adaptation control loop, enabling it to be utilized in a variety of existing self-adaptive systems. To evaluate TVA, we conducted simulations using three independent tools that allowed us to emulate tactic execution environments: R, IS3, and SWIM. We found that our TVA approach can significantly increase the predictability and dependability of executing a tactic, leading to a substantial reduction in uncertainty in the decision-making process.

***Index Terms*—Self-adaptive systems, Uncertainty, Decision-making**

I. Introduction

The world is increasingly relying upon autonomous, self-adaptive systems that have the ability to function independently without any human interaction. Examples of these self-adaptive systems include UAVs, IoT and medical devices. Self-adaptive systems must autonomously react to constantly changing requirements and environmental conditions. For example, a cloud-based self-adaptive system may need to prohibit access to a resource when a security threat is detected, or a medical device will need to increase the dosage of a medicine when specific conditions occur in the human body. Self-adaptive systems must also react to a nearly infinite set of unpredictable situations, many of which are impossible to predict at design-time.

To address encountered challenges and scenarios, self-adaptive systems have the ability to alter their configuration and/or functionality to react to changing internal or external events, all while continuing to fulfill system requirements and maximizing the utility (benefit) of the system [26]. A challenging decision for these systems is when and 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 [21]. 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. However, many of these tactics may become volatile at times due to the conditions they execute in. Therefore, *tactic volatility* is the degree to which a tactic can change rapidly, unpredictably, or erratically.

Tactic volatility can occur for many reasons, but the most significant contributor to this is *tactic latency* [25]. Tactic latency is simply the time required to fully implement a tactic. For example, provisioning a virtual machine could take several minutes [20], while it could take several milliseconds for a UAV to communicate with its base station. When various tactics are used to address a specific situation, the tactic with the highest expected maximum utility should be selected by the system, and tactic latency frequently has the most significant impact on this adaptation decision [3], [6].

Other forms of tactic volatility may also include how dependable a tactic is, i.e. how often the tactic behaves as expected. This is crucial for the decision-making process because it allows a system to weigh the reward of executing a tactic by its associated risk. If the system needs to adapt in a safer manner, it may opt to choose the tactic that returns slightly less utility but is almost guaranteed to work. On the other hand, if the system finds it self in an all or nothing situation, it may gamble and opt to execute the tactic that will return the highest possible utility, even though it has a higher chance of not returning the expected result.

Unfortunately, leading self-adaptation processes do not account for any tactic volatility in the decision-making process. 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, 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 is going to behave.

To address the limitations of tactic volatility and decision-making uncertainty, we created a *Tactic Volatility Aware* (TVA) process. During the decision-making process, TVA takes into

consideration tactic volatility by addressing uncertainties in the tactic's latency time and dependability. TVA also enables the self-adaptive system to learn from previously observed tactic volatility experiences, as machine learning is used to make estimates of how a tactic will behave under varying environments. We have integrated TVA into the widely-popular MAPE-K [18] adaptation control loop, enabling it to positivity impact a large variety of existing self-adaptive systems.

We used the following research questions as guidelines to understand how our approach is able to improve the decision-making process for self-adaptive systems.

RQ#1: Does accounting for tactic latency volatility improve the dependability of the decision-making process? We found that by incorporating latency volatility into the decision-making process, specifically through the utility function, the system became significantly more dependable. Our results saw as much as 50% improvement in terms of how many times a tactic experienced a latency time over its allowed threshold.

RQ#2: How much does modeling a tactic to the environment it is executing in improve the predictability of the tactic? We found that modeling a tactic to the environment it is executing in can increase the predictability of a tactic by as high as 80%. More specifically, we saw the average difference between estimated latency time and actual latency time drop from roughly 2.6 seconds to roughly 0.5 seconds.

RQ#3: Can our proposed utility equation outperform the baseline equation in terms of dependability and predictability? We found that our TVA approach is able to significantly increase the dependability and predictability of the decision-making process, thus leading to a significant reduction in uncertainty within the system.

The rest of the paper is organized as follows. Section II discusses existing research in self-adaptive systems. Our created self-adaptive simulation tool IS3, is described in Section III. Section IV defines the problem our approach addresses, while providing two motivating examples for our work. Section V describes how our TVA approach is implemented and defines our proposed utility equation. Section VI describes the tools used to evaluate our approach. Section VII describes the analysis methods used to determine the benefits of our work. Section VIII discusses the research results and the implications of our TVA approach and study. Section IX discusses limitations to our work and possible future work areas while Section X concludes our work.

II. Related Works

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 [4], [8], [14]. Moreno et al. [24] 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. [19] conducted work on uncertainty in self-adaptive systems, classifying five dimensions of uncertainty: location, source, nature, level/spectrum and emerging time. De Lemoset al. [10] 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 [10]. For example, imperfections in sensors can affect the monitoring phase, while the execution phase can be impacted by effector reliability. Weyns et al. [27] 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.

Cámara et al. [8] 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. This work 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. [11] examined sources of uncertainty in self-adaptive systems and then discuss 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 [15] 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. [13] 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.

A recent work by Moreno et al. [22] 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 [16]. 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. [7] was likely the first work to consider tactic latency, as it discussed how latency consideration could be used to assist the proactive adaptation

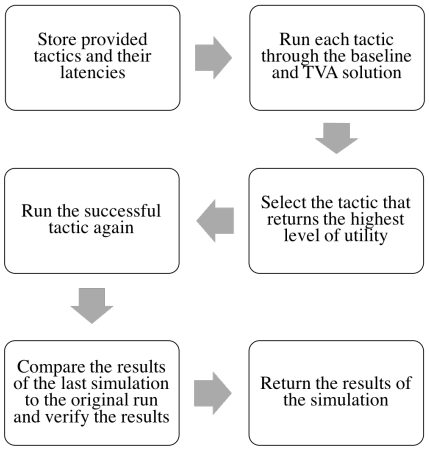

Figure 1. Sequence of IS3 Simulator Operations

process. Moreno [21], [25] discussed the need for a Proactive-Latency Aware (PLA) [7], [23] 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 [21] 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.

III. Intelligent Self-Adaptive Simulation Service

To help evaluate our proposed TVA approach, we created the *Intelligent Self-Adaptive Simulation Service* (IS3) tool. We created IS3 since existing self-adaptive simulation tools were not appropriate for our work. This is due to some tools requiring extensive modifications so they can behave as self-adaptive systems and others do not consider real-world events such as uncertainty. To guarantee the most accurate results, we developed IS3 as unique tool that: I) Includes real-world tactic volatility through observations of real-world systems, such as flight and weather data collected from around the United States II) Accepts custom datasets to analyze III) Is a hosted solution; eliminating the need for users to download, install, or configure the tool.

When analyzing a dataset, IS3 stores each set of latencies with their respective tactics. Each tactic is run using both our TVA solution and the baseline equation. The system then selects the tactic which was estimated to return the highest level of utility. IS3 will then run the successful tactic again to verify that it made the right decision. The result of this is saved and used to help improve future simulations. For full visibility, after a simulation, IS3 will return the dependability outcome and the latency differences of the chosen tactic in both the baseline equation and with our TVA solution. Figure 1 demonstrates this process.

IS3 is an open source, publicly available tool to allow anyone to run existing datasets using our TVA solutions. IS3 itself is continually building new sample datasets to run our tactical utility equation on as well. The hosted nature of IS3 makes it easily usable for both researchers and students. Relevant source code, and usage documentation can be found on the IS3 website: **https://www.is3Tool.com**.

IV. Problem Definition

Despite the growing emphasis on accounting for uncertainty in self-adaptive systems [8], [14], [22], [24], it has not been addressed at a tactic decision-making process level. Examples of tactic uncertainties, along with motivating examples for them, are described in this section.

A. *Tactic Latency*

The amount of time required to implement a tactic is known as *tactic latency*. Due to various circumstances, these latency times can sometimes become highly volatile. One example tactic could be to access information from a remote server, whose latency time could be volatile due to network or server congestion. As a result, this prevents the system from accurately predicting the behavior of a tactic. Previous works have shown that latency aware algorithms offer several advantages compared to those that do not consider latency [7], [12]. However, due to the unpredictable nature of many tactics, their latency times cannot always be accurately estimated. Unfortunately, existing work conducted on decision-making processes in self-adaptive systems also considers tactic latency to be a predetermined, static value. When a system is not able to account for tactic latency volatility or 'learn' from previous implementations about what this value may possibly be, this creates uncertainty in the decision making process.

Example: Latency Volatility in UAVs: An unmanned aerial vehicle (UAV), is often used in reconnaissance missions in order to keep soldiers out of harms way. As an example, there may be times when a UAV is flying over an area to locate a certain person with a main goal of taking hi-resolution photos of the area once the person is potentially located. In a scenario where the UAV identities the possible target, several steps must be taken. For this example, we will assume that the UAV needs to physically 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 something similar, it could take the UAV .4 seconds to communicate with the base station and receive the required data. This would result in the UAV missing its target window for photographing the area, and possibly miss identifying the target entirely. If the UAV

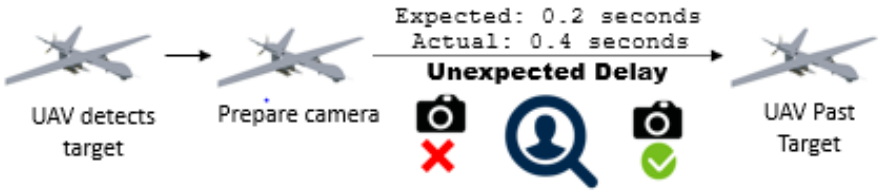

Figure 2. Impact of Unexpected Latency in UAV

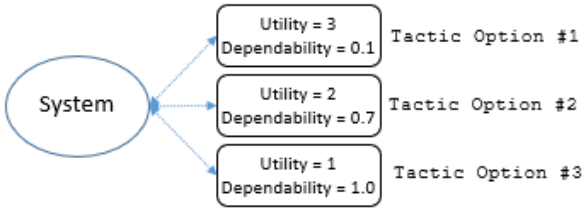

Figure 3. Impact of Dependability Uncertainty

was able to use previous knowledge about its prior mission experiences with 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 2.

B. *Tactic Dependability*

In a self-adaptive system certain tactics may be more dependable than others, and having knowledge of this is crucial for the decision-making process. The measure of how well a tactic correctly produces its expected functionality is known as *tactic dependability*. Due to numerous reasons, tactics may sometimes fail to produce their expected results leading to extra consumption of resources or even disastrous consequences in some cases. Therefore, having the ability to estimate when a tactic may or may not work properly is imperative for a self-adaptive system to remain dependable.

Example: Dependability Volatility in Cloud: In a cloud-based system, there may be times where interaction with remote computers is required for retrieving necessary information or to make an API call. Due to various reasons, the connection to these remote resources may be inhibited by network congestion or even cyberattacks. Regardless of how well the connections have worked in the past, there will be times where the system attempts to connect to one of the remote computers and gets an unexpected result. For instance, the tactic that connects to a remote computer may only return the expected result 80% of the time, while the API-call tactic may only return the expected result 90% of the time. Not being able to account for this in the decision-making process could lead to the system attempting to access the same remote resources every time, even though a similar more dependable resource may be available.

Figure 3 displays a possible scenario where a system has three tactic options 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 3, when the dependability and utility of each possible tactic differs, the system must have a way to weigh these differences in order to determine which tactic is most appropriate.

The Problem Addressed

Self-adaptive systems must be able to accurately predict likely future events and needs [21]. This enables the system to choose the most proper paths that lead to the highest returned benefit, while minimizing unnecessary risk. There may be environments where the system only has enough time to implement one tactic, and in the event of failure, the system will not have enough time to recover. Such an occurrence could lead to system failure, or the inability of the system to react to the environment change. 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.

V. Proposed Technique

In this section we describe how our TVA solution works. First, we discuss how TVA uses the MAPE-K adaptation control loop to aid the decision-making process. Second, we discuss the importance of using regression analysis in our TVA approach. Lastly, we describe our proposed utility equation that can be used in order to determine what tactic, or sequence of tactics, can result in the highest level of returned benefit.

A. *MAPE-K Feedback Loop*

For our proposed TVA technique to reduce uncertainty, the system needs to have access to a knowledge base of information in order to know when and how the system should adapt. Therefore, we have implemented our TVA technique into the widely used MAPE-K adaptation control loop [18] as shown in Figure 4. The following sections describe how our TVA approach is embedded into the popular MAPE-K control loop by using a self-adaptive cloud-based system example.

Knowledge Base: In order for the system to determine if there is a need for adaptation, it requires access to 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. For example, a cloud-based system that can self-adapt operating in a certain environment would store information about the

environment in the knowledge base. This information may consist of current network traffic, average response time, or any other information that may determine a need for adaptation. Thus, it is crucial the system have access to the knowledge base when determining a need for adaptation.

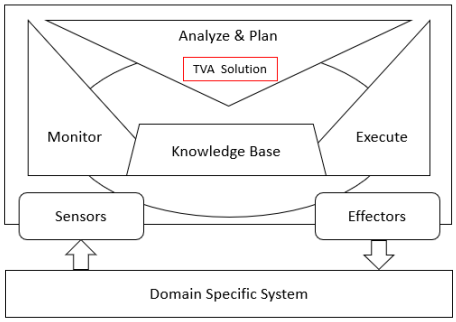

Figure 4. MAPE-K Adaptation Control Loop With TVA Integrated in the 'Analyze & Plan' Stage

Monitor: This phase captures the information necessary from the knowledge base to not only identify, but also make adaptation decisions. For instance, the cloud-based system might autonomously monitor network traffic to know when additional servers may be needed. If the system identifies network traffic levels to be at dangerous levels, the monitor phase would then pass this information on to the *Analyze* phase. By monitoring network traffic the cloud-based system is aware of the environment it is currently operating in, thus allowing the system to identify a possible need for adaptation.

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. More specifically, the analyze portion of this phase may determine that the network traffic identified in the monitor phase is well beyond the allowed variation. Thus, in order to create a plan for adaptation, the *Plan* phase needs to know the analysis results from the Analyze phase.

When the system has determined a need for adaptation in the Analyze phase, it will move to the *Plan* phase of the loop. In this phase, the system will analyze each tactic it has available for adaptation, and select the one that is estimated to return the highest utility. For example, the cloud-based system may have more tactics available to it than adding a server to handle increased network traffic. Other tactics may include redirecting network traffic to other online servers or reducing the amount of optional content shown to the user. Once the system has estimated returned utility for all available tactics, it simply selects the one with the highest value.

Execute: The final phase of MAPE-K is where the system executes the plan it has created in the previous phase. This plan will consist of either a single tactic or sequence of tactics that the system has estimated to return the highest level of utility. For example, the cloud-based system may determine that redirecting network traffic to other online servers will return more benefit than turning on an additional server to handle increased network traffic. Therefore, since the system has already identified a need for adaptation and created a plan, the tactic is executed immediately once this phase has begun.

B. Regression Analysis

Certain events have different outcomes depending on the surrounding environments. For example, a self-driving car may be testing how many feet it takes to stop from 60 mph when the road is wet and when the road is dry. Intuitively, this will lead to different results nearly every time. By using regression analysis, we can take these results and turn them into prediction models for estimating how long the vehicle will take to stop depending on how dry/wet the road is. However, today's latency aware approaches in self-adaptive systems do not take this approach to predicting tactic latency times during the decision-making process.

In our TVA approach the purpose of using regression analysis is to understand the relationships that may exist between tactic latency times and its associated predictor variables so we can create a *regression function* that can be used to estimate tactic latency times. Latency is considered to be the criterion (dependent) variable, while its predictor (independent) variables are determined through the regression analysis process. It is important to note that regression analysis estimates the conditional expectation of the dependent variable, that is, the average value of the dependent variable when the predictor variables are fixed. Other related methods like Necessary Condition Analysis (NCA) could be used that would estimate the maximum value of the dependent value based on fixed independent variables. However, we chose to not use this approach since it would estimate the maximum latency time based on the predictor variables. We decided that this was out of scope for our approach since estimating the maximum latency time would be considered as the *worst-case scenario* for the tactic, and that this could be more appropriately addressed in future work.

Due to the process of performing regression analysis, the regression functions created in our approach are not done at time of adaptation for multiple reasons. First, modern day programming languages do not have sufficient libraries for performing regression analysis appropriately. Second, if these models could be created at runtime, it would consume many of the resources available to the system and would be a time consumption burden. Lastly, and most importantly, regression analysis is conducted by specific data analytics teams in real-world organizations. There are many ways to find relationships among different groups of data and in order to build the most precise regression functions possible, this analysis needs to be appropriately addressed. Therefore, all regression analysis done in our TVA approach was performed outside of the simulations that evaluated our proposed utility equation. The associated regression functions developed for each tactic were then hard-coded into our simulations so they would be available at runtime.

C. Proposed Utility Equation

When a self-adaptive system decides which tactic is most appropriate, it selects the one that is expected to return the highest benefit or *utility*. Our proposed utility function (Equation 1) is described in this section.

$$U = \left(\frac{(T - \sum_{i=1}^{c-1} L_i) - L_c}{SD_c}\right)P \quad (1)$$

T*:* Maximum Defined Threshold To give weight to how long a tactic takes to execute, we define T as the maximum defined threshold alloted for a tactic to come to fruition. In dealing with sequential tactic execution, T will vary depending on what sub-tactic is being executed. For example, if a single tactic is made up of two tactics, T_x and T_y, T_x will take S_x seconds to complete and T_y will take S_y seconds to complete. However, T_y will only have $(T - S_x)$ seconds to execute because tactic T_x has already taken up time out of the originally defined maximum threshold. We denote this in our equation by subtracting from T the sum of the previously estimated tactic latencies in the sequence, where c represents the index of the current tactic, to acquire a new maximum threshold for the current tactic to execute in.

$\mathbf{L_C}$*:* Latency Estimate Our regression models have been pre-built and are not created at runtime during the decision-making process, due to reasons stated in Section V-B. Therefore L_c is merely the latency estimate derived from the regression function for the current tactic based on the environment the system is currently operating in. This allows us to fit each tactic to the environment it is trying to execute in, which is essential for the reliability of the decision-making process.

$\mathbf{SD_c}$*:* Latency Distribution Variability Incorporating latency volatility into the decision-making process means accounting for variability within the experienced latency times for a given tactic. To do this, we have defined SD_c as the standard deviation of the latency distribution for the current tactic in a sequence. By dividing the result in the numerator of our equation by the standard deviation of the latency distribution, we have used a process known as standardization. This enables us to conform our utility values to a common standard, allowing us to not only accurately account for latency variability, but it also gives us resulting utility values that can be appropriately compared.

In order to account for latency variability, we could have chosen to weight the average value of latencies for a tactic by its corresponding distribution's standard deviation, as shown in Equation 2. With this method, by multiplying the mean by its corresponding standard deviation, we could penalize tactics with higher standard deviations for being more unreliable than those that have lower standard deviations.

$$U = T - (L_i * SD_i) \quad (2)$$

However, Equation 2 was not appropriate for considering latency volatility. This is due to the fact that distributions with equal latency means and small differences in standard deviations will produce largely different utility values. Table I represents this issue.

Table I
WEIGHING MEAN LATENCY BY DISTRIBUTION'S SD

	Mean(L_i)	**SD**(L_i)	(T - L_i*SD_i)	**Utility**
Tactic 1	10	1.0	(14 - 10*1.0)	4
Tactic 2	10	2.0	(14 - 10*2.0)	-6

These misrepresentations could lead to false interpretations about the true differences between two tactic latency distributions, and the utilities they produce; leading to inaccurate decisions during the adaptation decision-making process. Therefore, by using the standardization process, we can appropriately account for latency variability.

P*:* Percent Chance of Success There may be situations where the system estimates that some of the tactics in a sequence are not expected to be completed in time. For example, tactic T_a may be made up of three sequential sub-tactics. During the decision-making process, the system may estimate the second tactic in T_a's sequence to take longer than the maximum threshold. This would cause not only the second tactic to not complete in time, but would also cause the third tactic in the sequence to complete well above the defined threshold. Therefore, we have defined P as the percentage estimate of successfully completed tactics in a sequence so we can weigh tactics by the estimated percent chance of completion. Using the previous example, tactic T_a would have a P value of $1/3$ since one out of three sub-tactics were estimated to complete on time.

VI. EVALUATION

We evaluated TVA against three different platforms. First, we used R to compare our proposed utility equation against a baseline equation. Next, we performed further analysis using our custom-built tool, *Intelligent System Simulation Service* (IS3), which takes a data set of previously observed tactic latencies as input, and implements the MAPE-K control loop to govern the self-adaptation decisions. Lastly, we evaluated TVA by using *SWIM* [2], a web infrastructure and management simulator that is commonly used to evaluate proposed changes to self-adaptive systems.

A. Baseline Equation

Equation 3 displays the baseline equation that our proposed TVA solution was evaluated against. This baseline consists of inputting the mean latency for each tactic as L_i. T remains as the maximum threshold constant. However, P is no longer the percent chance of success of the underlying sub-tactics. Due to the assumption in current self-adaptive systems that all tactics will always execute within their time limits, P must always have a value of 1. Therefore, the baseline utility function becomes a function of the mean of latencies, i.e. $U(mean(L))$.

$$U = (T - L_i)P \tag{3}$$

Equation 3 serves as our baseline approach because it *does not* account for variations in latency times, and does not make any predictions about tactic latency times based on the environment the system is running in. For example, two tactics could have the same mean latency of 5 seconds, but one could have latency values ranging from 0 to 10 seconds, and the other ranging from 3-7 seconds. Topically, these tactics might seem equal due to their equal means. However, the one with the higher variability poses a much larger threat to the reliability of the adaptation decision. Figure 5 demonstrates this issue.

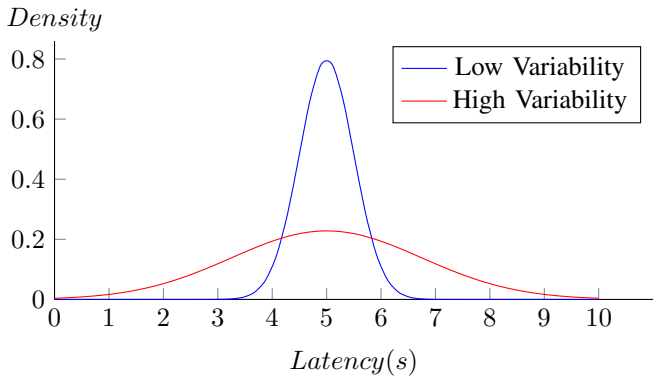

Figure 5. Tactics with High and Low Latency Variability

B. Simulations Using R

We used R to perform simulations that could be easily modified, and to efficiently perform regression analysis during the simulations. For example, instead of running a VM to collect data about latency times and tactic decisions that could take up to several minutes to complete, R ran simulations in a fraction of the amount of time. Furthermore, R has built-in functions for building regression functions and performing regression analysis since it was designed for performing statistical analysis. Therefore, the R simulations serve as an initial demonstration of the benefits of including TVA in the adaptation decision-making process.

We began our simulations in R by generating a truncated normal distribution of latency times for all tactics based off of already witnessed latency data. We used the truncated distribution over the regular normal distribution because it is impossible to have a latency time of less than 0. Therefore since we needed a lower bound, the truncated distribution was more appropriate.

In order to simulate an experienced tactic latency time in R, we trained our dataset using 60% of the latency times in the truncated distribution and used the remaining 40% as our testing data. To reduce potential variances caused by differences in training sets, we performed 10-fold cross-validation and used the resulting root-mean-squared error (RMSE) scores to evaluate the baseline and TVA approaches.

C. Simulations Using IS3

To assist with our evaluations, we built a feature into IS3 to fetch a file from different existing download mirrors from other countries from around the world. This component emulated the tactic latency and dependability volatility a UAV may encounter when interacting with different remote stations. If a mirror's latency began to have inconsistent results when compared to prior calculations, the dependability of that server would come into question. The benefit of using real-world latency data is that other factors naturally affect the latency of a tactic, which makes this dataset a more robust collection of tactical latencies than simulated datasets.

We measured the time (latency) it took to download a 75MB file from the three different servers, and scheduled this component to run every 10 minutes; adding each result to the expanding dataset. Each mirror was considered a tactic, and the download time it took to serve the file was its latency. By correlating latency times with each tactic, IS3 was able to verify our self-adaptive TVA solution. Using this process, we are able to include real-world volatility into our evaluation.

D. Simulations Using SWIM

The third and final tool used for our evaluation process is a web infrastructure and management simulator called *SWIM*. Unlike other web applications such as RUBiS [1], SWIM does not simulate the user interface (UI) functionality of the web application. Rather, the processing of requests is simulated as a computation that takes time [26]. More importantly, SWIM was built to simulate a self-adaptive web architecture, therefore no modifications were needed in order to make the tool self-adaptive for our simulations.

Within SWIM are three built in tactics that are available for adaption: add server, remove server and set dimmer; where dimmer is simply the proportion of responses for which optional content is included in the computation. Since SWIM already simulates network traffic and decides what tactic should be implemented based on available resources, acquiring latency data for each tactic was simple. To gather this data, we ran multiple instances of SWIM and recorded how long it took to execute the chosen tactic, resulting in hundreds of latency time data points.

To test our TVA approach, instead of the system deciding on what tactic to execute only by analyzing available resources, we included the baseline and TVA utility functions to aid in the decision-making process. Once the simulations were finished, we analyzed how the baseline approach performed against our TVA approach in terms of predictability and dependability.

VII. Data Analysis

To analyze the results found from our simulations, we chose several significance tests to evaluate if our TVA approach can significantly reduce uncertainty in the decision-making process of a self adaptive system. The methods used for analyzing the results from our simulations are described in this section.

A. Shapiro-Wilk's Test of Normality

The first step to determine what further methods would be used in our analysis was to see if our data is normally distributed. Determining normality within our results is important because certain test methods make assumptions that data follow certain distributions. In practice, this is often the debate between whether a parametric or non-parametric test should be conducted. Parametric tests, which should be used when assumptions can be made about the distribution of the data, e.g. it follows a *Gaussian Distribution*, are most of the well-known elementary statistical methods like t-tests and one-way ANOVA. On the other hand, non-parametric tests are "distribution free", meaning they make no assumptions about what distributions our data must follow. Therefore, we chose to use Shapiro-Wilk's test of normality to test the null hypothesis that our data is normally distributed.

Table II
NORMALITY TESTS OF SIMULATION RESULTS

	Critical Value	p-value
R	0.98526	0.3317
IS3	0.75012	$2.2e^{-16}$
SWIM	0.16249	$2.2e^{-16}$

To determine significance, we set our alpha-level, α, to 0.05. As shown in Table II, we found that two of our three simulations resulted in non-normally distributed data. The normally distributed data found in our R simulations came at no surprise seeing as the generated latency data was based on a truncated normal distribution. The two non-normal distributions, however, are the main factors for our methods described in the next two sections.

B. Mann-Whitney U Significance Test

To test for statistically significant differences between our TVA approach and the baseline approach, we used the Mann Whitney U significance test (MWU). MWU is a non-parametric test of the null hypothesis that 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 we had both normal and non-normally distributed data. Unlike the typical t-test used to test for significant differences between two independent samples, MWU does not require the assumption that our data come from normal distributions. Therefore, this enables MWU to provide more robust analysis for detecting significant differences between non-normally distributed data, without sacrificing the ability to detect significant differences between normally distributed data. The MWU results are described in Section VIII.

C. 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 calculate effect size statistics for each MWU significant test conducted. Effect size, which is a simple way to quantify the difference between two groups, has many forms. Commonly, the *Cohen's D* effect size statistic is calculated to quantify the difference between two groups, however, it is only appropriate for comparing the effect size of two group means. Therefore, since our analysis consists of MWU significance tests that look at differences in group medians and do not require distribution assumptions, we chose to use the non-parametric effect size statistic *Cliff's* δ. The resulting effect size values suggested by [9] 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 ($0.474 \geq \delta$).

VIII. DISCUSSION

A. Research Results

Our study examined the following research questions:

RQ#1: Does accounting for tactic latency volatility improve the dependability of the decision-making process? To assess if accounting for tactic latency volatility can improve the reliability of the decision-making process, we standardized the baseline equation by the latency distribution. As shown in Equation 4, we are not using the utility function that we have proposed in this work. The is due to how we identify reliability improvements. Had we used our proposed utility equation, we would not know whether improvements occurred because we standardized the utility value, used regression analysis to predict the latency time, or incorporated an estimate rate of success value. Therefore, the utility equation used to address this research question is only the baseline equation, divided by the standard deviation of the latency distribution. In this work, we refer to the utility equation used to address this research question as *Modified-A*.

$$U = \frac{T - L_i}{SD_i} P \tag{4}$$

To determine if accounting for latency volatility improves the dependability of the system, we defined a variable in our simulations as *critical failures*. This variable simply measured how many times the system experienced a tactic latency time over its maximum defined threshold, allowing us to measure how dependable the decision-making process could remain, even when encountering latency volatility.

Table III
CRITICAL FAILURE RESULTS

	Baseline	Modified-A	p-value	Cliffs δ
R	16.5	8.01	$2.2e^{-16}$	0.988
IS3	15.72	14.99	0.137	0.121
SWIM	11.83	9.76	$1.4e^{-05}$	0.353

As shown in Table III, incorporating the standard deviation of the latency distribution into the utility equation significantly

Table IV
AVERAGE DIFFERENCES AND RMSE RESULTS

	Baseline	Modified-B	RMSE-Baseline	RMSE Mod
R	2.651	0.504	2.747	0.655
IS3	4.856	4.300	7.185	6.240
SWIM	0.068	0.051	0.081	0.062

reduced the amount of critical failures witnessed within the system. Denoted by the p-values that are less than our α level of .05, we saw statistically significant differences between the modified equation and the baseline equation across two simulations; with the modified equation always producing the fewer critical failure values. We believe that a significant difference in critical failures was not observed with our IS3 simulations due to the extremely consistent behavior of two of the mirror requests. Therefore, since the requests already had naturally low latency volatility, the slight modification we made to the baseline equation was not sufficient enough for decreasing critical failures in this scenario.

Since our simulations resulted in two much lower critical failure results, *Cliffs* δ effect size statistic was needed to quantify these differences. In our R simulations, δ came out to be quite large at .988. Meaning, the difference between the groups in our R simulations is so large that it could be seen with the naked-eye. On the other hand, the significant results we saw in our SWIM simulations had a fairly lower effect size statistic, one that is classified as medium [9]. This tells us that further analysis should be conducted in order to determine more precise relationships between the simulated groups.

Although we were unable to eliminate critical failures, our results demonstrate the importance of considering tactic latency volatility in the decision-making process.

RQ#2: How much does modeling a tactic to the environment it is executing in improve the predictability of the tactic? To determine how modeling a tactic to the environment it is executing in can improve the predictability of the tactic, we examined the differences between the estimated latency time and the experienced latency time of a tactic. For the same reasons stated previously, we have only modified the baseline equation to incorporate regression functions for predicting latency times. We will refer to this modified equation as *Modified-B*.

In the simulations that addressed tactic predictability, we calculated RMSE scores to identify how well we could predict the behavior of a tactic. RMSE aggregates the magnitudes of the errors between prediction values and actual values for various simulations, and combines them into a single measure of predictive power [17]. Since RMSE is a measure of accuracy, we can use it to compare forecasting errors of different models for particular datasets.

As demonstrated in Table IV, by modeling a tactic to the environment it is executing in, we can significantly increase the predictability of the tactic. In our R simulations, the average difference between the experienced latency time and estimated latency time when using the baseline approach was 2.651 seconds. However, when we included the regression functions, we saw a significant improvement of latency time difference to 0.504 seconds. Our RMSE values from all three simulations further validate our findings, as the lower values indicate the models that included regression functions were more appropriate for estimating latency times.

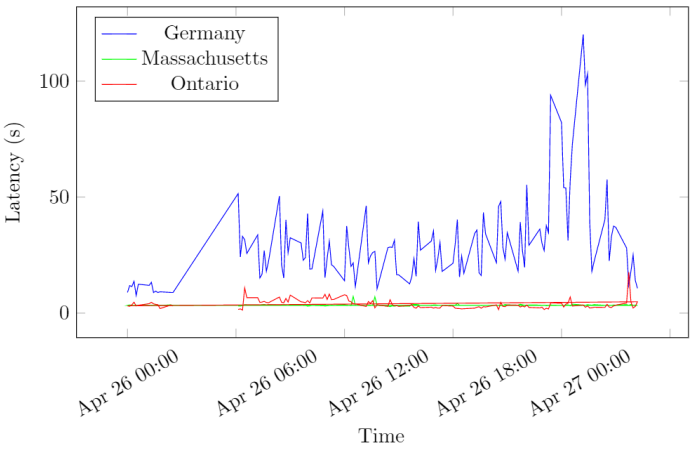

Figure 6. Time Series Data of Mirror Requests and Latency Time

Figure 6 represents some of the time series data gathered from our simulations in IS3, and helps to demonstrate why it is essential to model a tactic to its environment. As shown, the latency time for retrieving and downloading the file from the mirror located in Germany had extremely erratic behavior. Without being able to account for this, it would be nearly impossible to predict how long the file will take to download, creating high levels of uncertainty. Therefore, tactics must be modeled to the environments they operate in so the system can properly address tactics with erratic behavior, thus improving the predictability of the system.

RQ#3: Can our proposed utility equation outperform the baseline equation in terms of dependability and predictability? In the previous research questions we demonstrated the importance of using regression analysis for predicting the behavior of a tactic, and the importance of accounting for tactic latency volatility in the decision-making process; further justifying their use in our proposed utility equation. To test how our TVA approach and proposed utility equation compared to the baseline, we recorded both dependability and predictability data from our three simulation tools.

Using our TVA approach and our proposed utility function, we were able to significantly reduce the amount of critical failures in the R simulations and SWIM simulations, as seen with the previous research questions. Represented in Table V, the δ effect size statistics remained relatively the same demonstrating that not only was our TVA approach able to significantly reduce the amount of critical failures, but was also able to replicate the magnitudes of these differences.

Table V
CRITICAL FAILURE RESULTS: TVA VS BASELINE

	Baseline	TVA	p-value	Cliffs δ
R	16.22	7.49	$2.2e^{-16}$	0.934
IS3	15.18	14.63	0.104	0.145
SWIM	12.23	9.45	$2.24e^{-05}$	0.395

Table VI
RMSE RESULTS: TVA VS BASELINE

	Baseline	TVA	RMSE-Baseline	RMSE-TVA
R	2.513	0.731	2.681	0.814
IS3	5.035	4.282	7.364	5.721
SWIM	0.091	0.045	0.103	0.077

As shown in Table VI, our TVA approach was also able to significantly reduce the differences between estimated latency times and actual latency times across all three simulation tools; further justifying the importance of modeling a tactic to its environment so the system can become more predictable, leading to lower levels of uncertainty.

B. Implications

Our results have important implications for both researchers in self-adaptive systems, and in developers of the self-adaptive systems. This work is important from a research perspective because it demonstrates that: I) Tactic volatility can accurately be accounted for in the adaptation process II) Accounting for tactic volatility can have positive effects on the system. III) TVA can be easily implemented into the popular MAPE-K adaptation control loop IV) TVA can be an effective solution in addressing tactic volatility.

Our findings are significant from a real-world perspective for several reasons. As autonomous systems become more ubiquitous, the importance of improvements to the self-adaptation loop will only continue to grow. We also demonstrated how TVA can reduce uncertainty in self-adaptive systems, which is a problem that has long been shown to be detrimental to the decision-making process [5], [14], [22].

IX. LIMITATIONS AND FUTURE WORK

Although our TVA approach has demonstrated the ability to reduce uncertainty in the decision-making process for self-adaptive systems, there are 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 enough real-world data can be collected about how long each tactic takes to execute, latency values will continue to be pre-determined or a distribution will have to be generated like the process we used in R. Additionally, observing tactic latency will not be practical for addressing several types of infrequently utilized tactics. For example, a self-destruction tactic in a UAV will not be run more than once. This means that traditional latency-based decisions will need to continue to be used.

There may be instances where all available possible tactics are estimated to complete outside of the maximum defined threshold. In future work, we 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 regression analysis to model tactics to the environments they execute in. However, because the regression functions have to be built off-line and hard-coded into the system, there will come a point in time where the stored regression functions become outdated. For example, as a self-adaptive system records more and more data on how long it takes for tactics to execute, new relationships within the dataset may arise. Thus, the functions created from performing regression analysis may differ significantly. Therefore, there must be a process in place where the system can update its regression functions so it can continue to make the most accurate predictions possible in the decision-making 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 [21], [23]. Future work should be done to examine how poor tactic-based decisions due to latency volatility can affect both subsequent and concurrent tactic-based decisions. These results have the possibility of further demonstrating the importance of considering tactic latency 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. Future work will consist of including our adaptation process into physical equipment such as IoT devices and small UAVs. In our work explored tactic volatility in the form of latency and dependability. Future work could also explore other forms of tactic volatility including availability. While one could argue that this is outside of the scope, future work could also be done to explore ways of making the actual tactics more reliable.

X. CONCLUSION

Self-adaptive systems are beginning to account for tactic latency in their decision-making processes, however, our work is the first known of its kind to consider tactic volatility as a whole in the decision-making process. We demonstrated the effectiveness of our Tactic Volatility Aware (TVA) technique using R simulations. Furthermore, we validated our approach using a custom-built tool *IS3*, which simulated real-world latency data by requested data files from three separate mirrors. Lastly, we demonstrated the value of our TVA approach by using an existing self-adaptive system simulator, SWIM. Through our promising research we found that accounting for tactic volatility can be extremely beneficial for the system, specifically by reducing significant levels of uncertainty in the decision-making process.

ACKNOWLEDGEMENTS

Elements of this work are sponsored by [Hidden].

REFERENCES

[1] Rubis: Rice university bidding system. http://rubis.ow2.org/.

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

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

[4] 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.

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

[6] J. Cámara, G. Moreno, and D. Garlan. Reasoning about human participation in self-adaptive systems. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2015 IEEE/ACM 10th International Symposium on*, pages 146–156. IEEE, 2015.

[7] 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.

[8] 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.

[9] J. Coraggio, L. Devine, J. Kromrey, J. Romano, and J. Skowronek. Exploring methods for evaluating group differences on the nsse and other surveys: Are the t-test and cohen's-d indices the most appropriate choices? In *Annual Meeting of the Southern Association for Institutional Research*, 2006.

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. In *International Journal of Forecasting*, 2006.

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

[19] 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.

[20] M. Mao and M. Humphrey. A performance study on the vm startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE, 2012.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.