Reducing Tactic Latency Uncertainty in Self-Adaptive Systems

Hiten Gupta, Jeff Palmerino, Raseshwari Pulle, Stephen C. Cioffi and Daniel E. Krutz
Department of Software Engineering
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, NY, USA
{hg1928,jrp3143,rp3737,scc3459,dxvse}@rit.edu

ABSTRACT

Tactic latency is the time between when a tactic is started and when its effect is produced. Fortunately, many self-adaptive techniques are beginning to account for latency in their decision-making process. These adaptation processes consider tactic latency to be a static, human defined value. Unfortunately, many tactics have a volatile latency value; one that may be constantly changing. In this work, we present a Latency Volatility Aware (LVA) adaptation process. We achieved positive results evaluating LVA in several environments against a baseline process that only considers static latency values. We found that accounting for latency volatility can have a positive impact on the adaptation decision process, specifically by making the system more resilient to large variations in tactic latency. To support the evaluation of LVA, we created *TacSim*, a tool which enables us to quickly simulate multiple adaptation decisions while accounting for uncertainty in tactic latency. This paper advances the self-adaptive decision-making process by reducing tactic latency uncertainty. Our discoveries can serve as a foundation for future work in reducing uncertainty through understanding and accounting for tactic latency volatility.

CCS CONCEPTS

•**Computer systems organization** → *Self-organizing autonomic computing;* •**Software and its engineering** → *Software fault tolerance;*

KEYWORDS

Self-Adaptive Systems, Tactic Latency, Uncertainty Reduction

ACM Reference format:
Hiten Gupta, Jeff Palmerino, Raseshwari Pulle, Stephen C. Cioffi and Daniel E. Krutz. 2018. Reducing Tactic Latency Uncertainty in Self-Adaptive Systems. In *Proceedings of International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Gothenburg, Sweden, May 2018 (SEAMS'18),* 7 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Self-adaptive systems (SAS) have the ability to alter their configuration and/or functionality to react to changing internal or external events while continuing to fulfill system requirements and maximizing the utility (benefit) of the system [28]. 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 [23]. 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. *Tactic latency* is the time required to implement a tactic and varies depending on the chosen tactic. For example, provisioning a virtual machine could take several minutes [22], while fully activating a deicing mechanism could take one minute. The latency of each tactic has a significant impact on the decisions made by the self-adaptive system. When various tactics may be used to address a specific situation, the tactic with the maximum utility should be selected by the system, and tactic latency frequently has a significant impact on the adaptation decision [6, 8].

Previous work has shown that latency aware algorithms offer several advantages compared to those that do not consider latency [9, 16]. Unfortunately, existing prescient and probabilistic adaptation systems treat tactic latency as a static, predetermined value; one that is defined by a human at the establishment of the self-adaptive system [15, 17, 18]. In reality, tactic latency is frequently a volatile value; one that cannot be accurately predetermined as a static value. Tactic latency can change from moment to moment during the operation of a self-adaptive system. For example, network traffic or resource constraints could impact the time required to provision a virtual machine, or a power-saving mode could affect the time required to activate the deicing mechanism on a UAV. Systems may also have tactics with a maximum tolerated latency. For example, one tactic for a server could be to disconnect itself from the network in the event of an intrusion. Perhaps this would need to be done in 5 seconds to prevent data theft. If this tactic took more than the maximum threshold, then the system could face severe consequences due to it being compromised. Some tactics for disconnecting the server might be more efficient than others, but may take longer to implement than the maximum allowed threshold. Hence, this latency would need to be accounted for in the decision-making process. In some cases, perhaps the best tactic to use would be one that might be less efficient, but is more reliable in performing the tactic under the maximum allowed threshold.

Poor tactic-based decisions based on inaccurate latency values also have the capability of not only negatively impacting that specific tactic, but other tactics as well. For example, in some cases tactics may need to execute concurrently, while in others tactics may operate in a linear manner; relying on prior tactics to fully execute before they may be enacted.

SEAMS'18, Gothenburg, Sweden

DOI: 10.475/123_4

To properly decide upon the maximum utility of a tactic-based decision, the self-adaptive system must account for potential volatility in tactic latency. In this work, we address this problem by accounting for tactic latency variability by including this potential volatility into the decision-making process.

> *Through our promising research we found that accounting for tactic latency uncertainty can be beneficial in the self-adaptation process, specifically in making the system more resilient to large variations in tactic latency.*

The rest of the paper is organized as follows: Section 2 presents related works and Section 3 describes existing problem with tactic latency, along with a motivating example. Section 4 describes the proposed LVA technique and Section 5 describes the evaluation of LVA. Section 6 addresses our research questions and the implications of our results. Threats and future work are described in Section 7 and Section 8 concludes our work.

2 RELATED WORKS

Several works have examined tactic latency during the adaptation process. Cámara et al.[9] was likely the first work to consider tactic latency. The research discussed how latency consideration could be used to assist the proactive adaptation process. Moreno [23] discussed the need for a Proactive-Latency Aware (PLA) self-adaptation approach that integrated timing considerations into the decision-making process. In addition to supporting pro-activeness and concurrent tactic execution, PLA also considered 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 the time dimensions are recognized. Moreno [23] presented three latency aware approaches PLA-PMC, PLA-SDP and SB-PLA, which are both tactic based approaches. This work found that the addition of considering 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.

Shevtsov et al.[29] presented a new process for addressing new and changing requirements using guarantees in self-adaptive systems. Their work enabled users to specify an allowed threshold and dealt with requirements that changed at runtime. While profound for many reasons, especially due to its ability to address defined thresholds, this work differed from ours in that it did not address tactic latency in any manner; primarily focusing on guarantees.

Many works have discussed how uncertainty can be detrimental to the self-adaption process and how it can lead to decisions that lead to less than optimal utility [10, 14]. Moreno et al.[26] 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.[21] conducted work on uncertainty in self-adaptive systems, classifying five dimensions of uncertainty: location, source, nature, level/spectrum and emerging time. De Lemoset al.[13] 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 [13]. For example, imperfections in sensors can affect the monitoring phase, while the execution phase can be impacted by effector reliability. Weyns et al. [31] 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.

While there are a variety of different and helpful ways of addressing uncertainty in the adaptation process, our work is the only technique of addressing uncertainty/volatility in tactic latency.

3 PROBLEM DEFINITION

Tactic latency is an important consideration in the adaptation process [23, 24, 28]. Many tactics are not instantaneous and often require several seconds to several minutes to implement [22, 27]. Problematically, situations may arise when by the time a tactic is able to be completed, that the need for the tactic may have concluded. Additionally, tactics often rely upon other tactics to be completed before they may begin. For example, a server may not be rebooted if it is still being re-imaged. There may also be cases where tactic a and tactic b are equally good. However, if tactic a has less latency then it is probably a better option. In a utility based approach a similar problem could occur if tactic b was slightly better than tactic a, but tactic tactic a had less latency and would therefore start accruing utility improvement before tactic b. In this situation, it is not possible for the system to make the most appropriate decision unless tactic latency is considered [23].

Adaptation decision-making processes are designed to determine the adaptation decision to take (if any) that will maximize the utility of the system over other available options. While some adaptation decisions may be made in isolation with no latency, other decisions will be more complex having varying latency, with dependencies and conflicts with other adaptation tactics [25]. The adaptation process is sequential, where each decision can affect other subsequent and concurrent decisions in the ever changing system [26]. This means that the adaptation process must evolve with the changing system and account for this volatility.

PLA [23, 24] approaches are an improvement over reactive approaches since they account for both the current and anticipated system adaptation needs, while also considering the latency of the adaptation tactic [9, 25]. PLA adaptation also has the benefit of accounting for concurrent tactics. This means that latency volatility can not only affect the current tactic being considered, but also concurrently executed tactics. Unfortunately, current PLA decision processes consider latency to be a predefined, static value. However, tactic latency may actually be a volatile value. These inaccurate latency values can lead to harmful decisions made by the adaption process ones that lead to less than optimal utility. PLA decision-making techniques require accurate information to make informed decisions, and static latency values only lead to inaccurate input into the decision-making process.

Motivating Example To illustrate the necessity of accounting for volatility in tactic latency, we will use a generic web-based system as an example. This example system has many potential

trusted and untrusted actors, and is often maliciously attacked in a variety of methods in an attempt to gain access to sensitive information and functionality. In the event of an unknown event in the system, the system has two options: A) Verify the authenticity of the actor internally (expected latency .5 seconds) B) Connect to an external resource to verify actions (expected latency .3 seconds). This would be a time sensitive action since the longer a potentially unauthorized user had access to a system, the more damage they could cause. In this example, the tactic that would provide the most utility would be 'B' since it would provide the verification faster. However, since the connection to a remote server was being made, it can be reasonably assumed that the system should account for some volatility in the tactic latency. For example, network traffic or an overloaded connecting server could lead to unexpected latency. A security attack could also affect the latency of this tactic. If this volatility of the tactic latency is not being accounted for, then the utility of the decision made by the self-adaptive system could be negatively impacted. Our proposed technique will account for this volatility, thus leading to an optimized adaptation decision with higher utility.

4 PROPOSED TECHNIQUE

Our proposed Latency Volatility Aware (LVA) adaptation technique accounts for tactic latency volatility by including previous tactic latency observations in the self-adaptation process. In many real-world systems, tactic latency should be considered to be a distribution of values as their times may be variable. To more accurately account for the latency times of future tactic implementations, the system should have the knowledge of such tactic latency distributions. To account for this tactic latency variability, we created the formula shown in Equation 1.

$$U = \frac{(T - L_i)W}{SD_i} \tag{1}$$

LVA considers utility (U) to be dependent on the decision period (T), the latency distribution (L) of the tactic being executed, and the impact of the tactic, noted as (W). To select the appropriate tactic, our proposed approach considers latency as a distribution, not a single deterministic value. Therefore, any uncertainty that might exist in a tactic's latency distribution can be minimized, thus leading to more accurate adaptation decisions. Equation 1 divides the overall result in the numerator by the standard deviation of the tactic's latency distribution. This process, otherwise known as standardization, enables us to conform our utility values to a common standard. Therefore, our proposed technique allows 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, two latency distributions that have the same average but varying standard deviations would compute different utility values. By multiplying the mean by its corresponding standard deviation, we could penalize tactics with higher standard deviations for being more unreliable, or inconsistent, than those that had lower standard deviations.

$$U = (T - L_i * SD_i)W \tag{2}$$

However, Equation 2 would not have properly accounted for latency variability, and it would not have been scalable. 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 1 represents this kind of event.

Table 1: Differences in Utility

	Mean(L_i)	**SD(L_i)**	**(T - L_i*SD_i)*W**	**Utility**
Tactic 1	10	1.0	(14 - 10*1.0)*2	8
Tactic 2	10	2.0	(14 - 10*2.0)*2	-12

Misrepresentations of these tactics could lead to false interpretations and conclusions about the true differences between their 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.

In adaptive systems, the time taken for each adaptation can be divided into discrete time periods (T). Each period has the capacity to accommodate one or more tactics depending on the latency of the tactics that are executed. Therefore, a tactic's utility is largely affected by the proportion of time it takes to execute, to the time alloted for execution. Thus, overall efficiency of the system can be increased when tactics with a low mean latency time and standard deviation are executed first. Our proposed technique can be used in numerous decision-making adaptation processes, as LVA is especially suited to those that use Probabilistic Model Checking (PMC) [23, 28]. LVA offers several primary advantages over existing state of the art techniques that consider tactic latency to be a predetermined static value:

(1) Observe changing volatility in the system and update the expected latency value for the adaptation.
(2) Factor the standard deviation of a given tactic's latency distribution in the decision-making process.
(3) Adjust human predefined tactic latency values to more accurately convey the current system and environment.
(4) An optional domain specific feature to create a maximum allowed threshold for tactic latency. This could help the resiliency of the system by avoiding tactics with outlaying latency values that are outside of a tolerated system threshold.

LVA improves the adaptation process by providing more accurate tactic latency values to the decision-making process.

5 EVALUATION

To demonstrate the benefits of using our LVA approach, we evaluated it across two different platforms. We first used R to compare our proposed utility function to a baseline equation. Next, we performed further analysis using our custom-built tool, *Tactic Simulator* (TacSim). This tool takes a data set of previously known tactic latencies as input, and implements the MAPE-K control loop to govern self-adaptation decisions. We evaluated LVA in both

of these stages to provide further confidence and demonstrate its effectiveness.

Baseline Equation. The baseline equation used to evaluate LVA consists of inputting the mean latency for each tactic as L_i which is represented by Equation 3. Therefore, since T and W are constants in our equation, the baseline utility function becomes a function of the mean of latencies, i.e. $U(mean(L))$. Because of its nature, this equation produces only a single utility value for each tactic, from which the simulations can then simply select the tactic with the highest utility value.

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

This approach *does not* account for variations in latency times (i.e. the standard deviation (SD) of a latency distribution). For example, two separate vectors could have the same mean tactic latency of 50, but one vector could range from 0 to 100, and the other could range from 35-65 (a rough example shown in Figure 1). Topically, the vectors might seem equal. However, the one with the higher variability poses a much larger threat to the reliability of the adaptation decision.

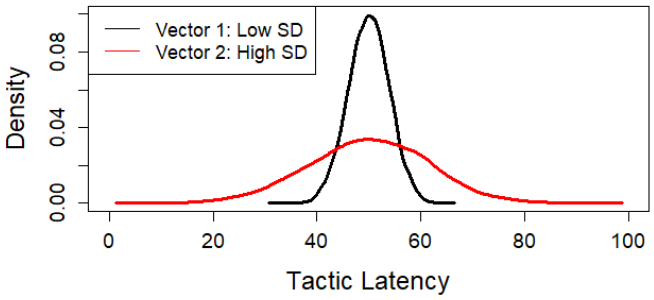

Figure 1: Distributions with Low and High Variability

Figure 1 demonstrates why Equation 3 is our baseline approach, and the motivating factor for our proposed LVA technique that accounts for variability in tactic latency times. By collapsing everything we know about a distribution into its mean, as shown in Equation 3, we cannot account for latency distributions with high volatility.

5.1 Simulations Using R

The purpose of using R was to run simulations that could be easily modified, without extensive time consumption. For example, instead of running a VM (or something similar) to collect data about latency times and tactic decisions that could take up to several minutes to complete, R allowed us to run simulations in a fraction of the amount of time, while also enabling simpler simulation modification (i.e. change input, output, analytical process etc). The R simulations serve as an initial demonstration of the benefits of including LVA in the tactic decision-making process.

We began our simulations in R by first generating a truncated normal distribution of latency times for all tactics, based off of already witnessed latency data. We chose to use the truncated distribution over the regular normal distribution because it is impossible to have a latency time of less than 0, so since we needed a lower bound, the truncated distribution was more appropriate. We then implemented our proposed LVA utility equation and compared it to the baseline. This step demonstrated that our LVA approach was able to provide a more reliable adaptation decision.

Results. As previously described, the latency data we used in our simulation was generated by a truncated normal distribution in R. The final data set used for analysis consisted of three tactics that each had a sample size of 10,000 latency times. These were all generated from the truncated distribution. A sample size this large provided sufficient data for analysis to be conducted, and was small enough to where running our simulations did not take an extensive amount of time. The baseline approach was our simplest approach in that no standardization methods were used or involved with the equation, only the mean latency time for each tactic was used.

Table 2: Baseline Tactic Utilities

	Tactic 1	**Tactic 2**	**Tactic 3**
Utility	64.03	71.36	61.02
		Decision	Tactic 2

Table 2 represents the results from our baseline simulation. As demonstrated, Tactic 2 should be chosen as the adaptation tactic as it produced the highest utility. However, what Table 2 doesn't tell us is how reliable each of the utility values are. Meaning, the underlying latency distributions for each tactic could vary significantly, and if so, the resulting utility values would be considered unstable. Therefore, as mentioned before, collapsing everything we know about a distribution into the mean, we cannot account for latency variability appropriately. Thus, our proposed technique addresses this problem in a way that we can appropriately, and consistently penalize tactics for having unreliable latency distributions.

Table 3: Standardized Utility Values

Tactic	**Mean Utility**	**SD**	**STD(U)**
T_1	64.17	19.77	3.24
T_2	71.46	14.10	5.07
T_3	60.85	13.23	4.60
		Decision	Tactic 2

Table 3 represents the results from the simulation that used our proposed utility equation. Although the final adaptation decision was the same, T_2 was chosen in both scenarios, this method did appropriately account for latency variability. This can be seen as our baseline equation results, Table 2, showed the two remaining tactic utility values, ranked in order, were T_1 and T_3. However in our alternative approach based off standardized utility values, the ranking of the two remaining utility values changed to T_3 then T_1. This change in order demonstrates the exact purpose of our

proposed utility equation, and what it accomplishes. For example, although T_1 had a higher mean utility (64.17) compared to T_3 (60.85), its final standardized utility value was penalized significantly. Thus, T_3 is a more reliable adaptation tactic.

Our initial simulations in R demonstrate that our proposed technique is well-suited for adaptation decisions as it can appropriately account for variations in tactic latency times. This leads to more precise adaptation decisions at run-time.

5.2 Simulations Using TacSim

To perform the next analysis step of LVA, we created a custom simulation tool *TacSim*. This was used to further analyze LVA.

TacSim Tool TacSim simulates a cloud-based hosting structure containing multiple possible tactics with varying latency distributions. This tool implements the MAPE-K control loop for all self-adaptation decisions, with our baseline and proposed utility equations embedded in the 'Plan' phase of the loop. Depending on the instance of TacSim specified, the simulator will either run with the tactic utility equation equal to the baseline approach or our proposed utility equation.

We created TacSim since existing simulation tools do not account for the possibility that tactic latency values may be variable. Therefore, by creating TacSim, we were able to simulate thousands of adaptation decisions, with the ability to see how these decisions change based on tactic latency uncertainty.

TacSim also monitors "critical failures", which we define as a tactic that took longer than the permitted execution time to complete. For example, if a server encounters a potential threat where the most appropriate decision must be determined and executed within a 5 second maximum threshold, then the tactic latency must not take longer then 5 seconds. By keeping track of these critical failures throughout the simulations, we were able to evaluate our proposed utility equation against the baseline equation.

Results. In order to keep track of critical failures in the system, we defined a *Maximum Completion Time* (MCT) variable that represented the allocated time allowed for an adaption tactic to complete, if chosen. To provide robustness, we conducted simulations with MCT values ranging from 5 to 8 seconds. We selected these values because in extremely rare cases, a self-adaptive system may have to choose between tactics whose mean latencies are above the defined MCT. Thus, handling these situations is out of scope for this initial research. Since the data we used for our study had mean latencies of less than 5, the range of 5 to 8 seconds provided realistic time constraints for tactic execution.

Within each individual simulation, we conducted 100 iterations of the MAPE-K control loop. For example, if we ran an instance of the baseline equation, with a MCT value of 5, the result was a total number of critical failures for that simulation out of 100. We then repeated this process 50 times for each MCT value. The large sample sizes chosen for MAPE-K loop iterations and overall process iterations were chosen to help ensure our study had sufficient statistical power, which is the ability to correctly identify any relationships that may exist.

To analyze our simulations and test differences between the baseline equation and our proposed utility equation, we used the Mann-Whitney U (MWU) test. MWU is a nonparametric significance test that can be used to determine whether two independent samples were selected from populations having the same distribution. The results from our analysis are shown in Table 4.

Table 4: MWU Results of Differences in Critical Failures Across Utility Equations

MCT	Baseline	Proposed	p-value
5*sec*	12.680	7.637	$3.2e^{-8}$
6*sec*	4.210	0.822	$5.1e^{-10}$
7*sec*	1.027	0.042	$3.8e^{-7}$
8*sec*	.2142	0.001	$2.4e^{-3}$

In all four cases of MCT, we found that the differences in the amount of critical failures experienced between our LVA equation and the baseline equation was statistically significant. These results provide further evidence that our proposed LVA equation is more resilient to volatility in tactic latency.

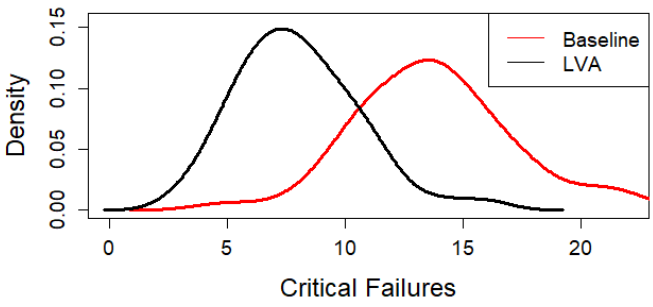

Figure 2: Critical Failures for MCT of 5 seconds

As shown by Figure 2, LVA consistently experienced fewer critical failures per simulation compared to the baseline equation. We found that overall, LVA produced roughly 60% less critical failures throughout the simulations, versus the baseline equation.

6 DISCUSSION

Our research has several important implications. Only a few techniques currently account for tactic latency, and our research is the first known of its kind to consider the volatility of tactic latency. We not only demonstrated that latency uncertainty can be incorporated into the decision-making process, but in that accounting for this variability can assist in making the system more resilient. This is important since resiliency is essential for many mission-critical self-adaptive systems [7, 11, 12, 30].

System resiliency is defined by Laprie [20] as one whose service can justifiably be trusted when facing changes. Examples of resiliency in real-world self-adaptive systems include properly reacting to unforeseen environment changes or cyberthreats. Improving resiliency is important for reducing failures in both the ability to

conduct the mission and possibility even in the self-adaptive system itself. For example, a drastic failure of resiliency for a UAV could lead to it crashing due to not properly addressing environmental conditions. Self-adaptive systems considering resiliency is a critical concern since the number of changes that the self-adaptive system and environment can face is virtually infinite.

As self-adaptive systems are likely to become more ubiquitous, they will likely continue to grow to be more reliant on other self-adaptive systems and components. This means that as tactics rely upon various internal and external components, that latency volatility will likely become more of a widespread concern.

7 THREATS AND FUTURE WORK

Although we believe our initial research demonstrates the benefits of accounting for tactic latency uncertainty, there are several challenges that need to be overcome. In the initial phases of incorporating our LVA approach, a tactic's latency will still be predetermined by the human engineer 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 also be instances where the maximum allowed threshold is surpassed by all of the possible tactics available to address the situation. We will need to incorporate this possibility into the decision-making process in future work, as there will likely be some overhead in accounting for tactic latency volatility. This cost may not be justifiable by all adaptive systems.

Our LVA process includes using prior values in its decision-making process. Future work will need to be done to determine when, and if these prior values should ever be limited. For example, outdated latency information could lead to inaccurate decisions, thus determining when latency information should no longer be included in the decision-making process will need to be done. Additionally, there may come a point where it becomes wasteful to continue including latency volatility into the decision-making process if a specific tactic has consistently been determined to produce the best utility. In this case, to maximize system efficiency there may be a point where including latency volatility into the decision-making process becomes a redundant, wasteful process.

We evaluated our proposed technique using both R and a custom-built TacSim tool. Despite our positive results, we have yet to incorporate LVA into a real-world environment and setting for further analysis. This should be done to further justify and evaluate our LVA approach. Future work may also be conducted to evaluate our technique using other existing artifacts such as *exemplars* [3].

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 [23, 24]. 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.

We did not use RUBiS [2], DART [19], or SWIM [4] to evaluate our LVA approach, instead we chose to create our own evaluation tool TacSim, which we felt offered a more robust simulation environment for our specific needs. Future work could be done to evaluate LVA using these other, preexisting tools. However, based on our evaluation, we believe that the conclusions will not be significantly different from what we found. Future work could also including using popular sample datasets such as FIFA'98 World Cup [5] to further evaluate our work.

8 CONCLUSION

Although self-adaptive systems are beginning to account for tactic latency in their decision-making processes, our work is the first known of its kind to consider tactic latency volatility in the decision-making process. We first demonstrated the effectiveness of our Latency Volatility Aware (LVA) technique using R simulations. We further validated our technique using a custom-built tool *TacSim*, which simulated a cloud-based hosting structure. Through our promising research we found that accounting for tactic latency uncertainty can be beneficial in the self-adaptation process, specifically in making it more resilient. Further project related information, including relevant data, is available on the project website [1].

REFERENCES

[1] Latency volatility aware (lva). https://lvatactic.github.io/.
[2] Rubis: Rice university bidding system. http://rubis.ow2.org/.
[3] Seams exemplars. https://www.hpi.uni-potsdam.de/giese/public/selfadapt/category/exemplar/.
[4] Simulator of web infrastructure and management. https://github.com/cps-sei/swim.
[5] M. Arlitt and T. Jin. A workload characterization study of the 1998 world cup web site. *IEEE network*, 14(3):30–37, 2000.
[6] Ca. Analyzing latency-aware self-adaptation using stochastic games and simulations.
[7] J. Camara, R. Lemos, M. Vieira, R. Almeida, and R. Ventura. Architecture-based resilience evaluation for self-adaptive systems. *Computing*, 95(8):689–722, Aug. 2013.
[8] J. Camara, 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.
[9] 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.
[10] B. H. Cheng, R. De Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, et al. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, pages 1–26. Springer, 2009.
[11] M. Correa, J. B. C. Jr., M. A. Rossi, and J. R. A. Jr. Improving the resilience of uav in non-segregated airspace using multiagent paradigm. In *2012 Second Brazilian Conference on Critical Embedded Systems*, pages 88–93, May 2012.
[12] R. De Lemos, D. Garlan, C. Ghezzi, and H. Giese. Software engineering for self-adaptive systems: Assurances (dagstuhl seminar 13511). In *Dagstuhl Reports*, volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.
[13] R. de Lemos, H. Giese, H. A. Müller, and M. Shaw. Software engineering for self-adaptive systems ii. 2010.
[14] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer, 2013.
[15] N. Esfahani. *Management of uncertainty in self-adaptive software*. PhD thesis, George Mason University, 2014.
[16] 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.

[17] 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.

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

[19] S. A. Hissam, S. Chaki, and G. A. Moreno. High assurance for distributed cyber physical systems. In *Proceedings of the 2015 European Conference on Software Architecture Workshops*, page 6. ACM, 2015.

[20] J.-C. Laprie. From dependability to resilience. In *38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks*, pages G8–G9. Citeseer, 2008.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] G. A. Moreno, J. Camara, 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.

[27] 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.

[28] 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.

[29] S. Shevtsov, D. Weyns, and M. Maggio. Handling new and changing requirements with guarantees in self-adaptive systems using simca. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 12–23. IEEE Press, 2017.

[30] M. Villarreal-Vasquez, B. Bhargava, P. Angin, N. Ahmed, D. Goodwin, K. Brin, and J. Kobes. An mtd-based self-adaptive resilience approach for cloud systems. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 723–726, June 2017.

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