12-2021

Evaluation of Neuro-Evolution Algorithms for Tactic Volatility Aware Processes

Aizaz Ul Haq
au7149@rit.edu

Evaluation of Neuro-Evolution Algorithms for Tactic Volatility Aware Processes

by

Aizaz Ul Haq

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Software Engineering

Supervised by

Dr. Daniel E. Krutz

Department of Software Engineering

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

December 2021

The thesis “Evaluation of Neuro-Evolution Algorithms for Tactic Volatility Aware Processes” by Aizaz Ul Haq has been examined and approved by the following Examination Committee:

Dr. Daniel E. Krutz
Assistant Professor, Department of Software Engineering
Thesis Committee Chair

Dr. Qi Yu
Professor, Department of
Information Sciences and Technologies

Acknowledgments

I am grateful for all of those who took time out of their schedules to aid and guide me in developing this thesis. I would like to give special thanks to the members of the AWARE Lab for contributing to this paper.

Abstract

Our society is increasingly evolving to rely on computer mechanisms that perform a variety of tasks. From a self-driving car to a satellite in space relaying data from Mars rovers, we need these systems to perform optimally and without failure. One such point of failure these systems can encounter is tactic volatility of an adaptation tactic. *Adaptation tactics* are defined workflows that allow systems to navigate their environment. *Tactic volatility* is the variance in the behavior in the attribute of a tactic, such as cost and latency and/or the combination of the two. Current systems consider these tactic attributes to be static. Studies have shown that not accounting for tactic volatility can adversely affect a system's ability to operate effectively and resiliently.

To support self-adaptive systems and address their limitations, this paper proposes a *Tactic Volatility Aware* solution that utilizes eRNN (TVA-E) and addresses the limitations of current self-adaptive systems. For this research, we used real-world data that has been made available for use by researchers and academics. This data contains real-world volatility and helps us demonstrate the positive impact TVA-E when used in self-adaptive systems. We also employ the use of uncertainty reduction tactics and how they can assist in accounting for tactic volatility. This work will serve as an evaluation and a comparison of using different machine learning methods to predict and account for tactic volatility.

We will study different predictive mechanisms in this paper: *Auto-Regressive Moving Average* (ARIMA), *Evolving Recurrent Neural Network* (eRNN), *Multi-Layer Perceptron* (MLP) and *Support Vector Regression* (SVR). These methods will be studied with our TVA-E process and we will analyze how they can enhance a self-adaptive system's performance when it accounts for tactic volatility.

Chapter 1

Introduction

The world is evolving to rely on computer systems that are self-adaptive, autonomous and require minimal to no human input and assistance. An example of these systems include self-driving cars, IoT devices such as smart thermostats and medical devices such as smart insulin pumps. These systems are required to adapt to changes in internal and external environments. For example, an insulin pump will have to change the rate of flow of insulin according to it's hosts blood glucose level. A self driving car will have to adapt its power to accelerate according to the number of passengers with in the vehicle. These self-adaptive systems are required to adhere to rigorous operating standards to maintain functionality, effectiveness and safety. These operating standards are defined in a *Service Level Agreement* (SLA) which layout the exact requirements a system should meet [41].

Self-adaptive systems react to the changes in environment with the help of adaptation tactics. *Adaptation tactics* are defined workflows that allow systems to navigate their environment. These tactics are pre-defined according to each system's *Service Level Agreement* (SLA) [41]. Example for such adaptaion include a server farm adding new *Virtual Machines* (VM) to the stack to meet demand. The amount of machines added and the type (computational power) depends on the demand. Another example of this is *Unmanned Ariel Vehicles*(UAVs) that need to perform an *over the air-update* (OTA).

These tactics have several attributes such as cost and latency associated with them [52, 57, 49]. Tactic cost is the consumption of resources to perform the said tactic. This can be in the form of energy or the monitory value of wear and tear of hardware. Tactic latency is the time between the initiation of a tactic and its completion [52, 56, 57, 49]. Palmerino

*et al.*has previously demonstrated how considering tactic latency can lead to improvement in system performance [62].

Tactic volatility is the variance in the behavior of the attribute of a tactic, such as cost and latency and/or the combination of the two. Tactic volatility has shown to have a significant impact on the performance of a decision making system [51, 57, 55]. For example, the decision by a drone to go "1 level" up or down can have variable outcome as it will not always take the same amount of time every time due to changes in external and internal environment of the drone. Performing the same action can also have variable cost as the drone might be assisted by a wind draft in certain scenarios and worked against in others. Another example is that of a smart insulin pump. The pump has to take into account the life of its battery (cost) while pumping insulin and maximizing its life while also taking into account the time (latency) it will take to inject a full dose of insulin.

The system's SLA also specifies the decision metrics of a system. For example, it specifies how the system should calculate the utility of taking an action and measuring it against a pre-defined threshold. This utility is calculated by taking into account the tactic attributes such as cost and latency.

Real-world systems don't account for tactic-volatility [51, 57, 55]. They usually consider tactic attributes such as cost and latency as fixed as fixed quantities. This is counter-intuitive to a system's decision making abilities in a dynamic and variable environment. Studies have shown that not accounting for tactic volatility to also negatively effect a system's effectiveness, resiliency and security. [33, 69, 39, 61].

This paper addresses the limitations of current self-adaptive systems processes. We propose a *Tactic Volatility Aware* solution that utilizes neuroevolution algorithm, capable of evolving recurrent neural networks, (TVA-E) and addresses the limitations of current self-adaptive systems. Our solution takes into account tactic volatility and also maintains system requirements defined in the SLA. This solution utilizes prediction mechanisms that predict the cost and latency of an adaption. This solution results in the system making more optimal decisions and satisfy the requirements in the SLA. This work also uses real-world

data that has been made available for use by researchers and academics. This data contains real-world volatility and helps us demonstrate the positive impact TVA-E when used in self-adaptive systems. We will study different predictive mechanisms in this paper to support our work: *Auto-Regressive Moving Average* (ARIMA), *Evolving Recurrent Neural Network* (eRNN), *Multi-Layer Perceptron* (MLP) and *Support Vector Regression* (SVR). These methods will be studied along with eRNN to compare and analyze its performance when used with our TVA-E process.

This work also employs the use of *Uncertainty Reduction Tactics* (URTs) and demonstrates how they can assist in accounting for tactic volatility. We studied the performance of TVA-E with and without URTs and we demonstrate why they are useful.

To summarize, this paper makes the following contributions:

1. **Problem Demonstration:** We demonstrates the negative impact to system performance for not accounting for tactic volatility.
2. **Concept:** This is the first known work to apply the novel eRNN approach to predict tactic volatility and is the first to incorporate and evaluate the use of *Uncertainty Reduction Tactics* [53]. It also compares the performance of eRNN to other machine learning models. We also provide the CELIA simulation tool for analyzing the performance of self-adaptive systems. We used CELIA to evaluate the performance of TVA-E using the predictions from different machine learning models enhanced by uncertainty reduction tactics.
3. **Experiments:** We demonstrate the positive impact that our eRNN-based process and uncertainty reduction tactics have in accounting for tactic volatility. This work demonstrates the usefullness of our eRNN-based TVA process and uncertainty reduction tactics when accounting for tactic volatility by running experiments and a statistical analysis on the results.
4. **Tool and Dataset:** This work was done with future research in mind. CELIA creates both tactic volatility data and evaluates tactic volatility prediction processes. CELIA will enables researchers to I) evaluate the effectiveness of their own tactic

volatility aware prediction processes, and II) the CELIA data generation tool with their tactic volatility aware processes. Therefore we are making all the tools, data and results public to encourage research and growth in this domain.

Chapter 2

Problem Definition

The tactic attributes are used to calculate the *utility*, or usefullness [2], for a system to execute a specific tactic [51, 56]. These attributes will frequently experience unexpected change or *tactic volatility* that will impact the performance of a system [61]. This study will focus on two forms of tactic volatility: *latency* and *cost*. These two forms of tactic volatility have been found to be prevalent throughout self-adaptive systems [61, 51, 54].

2.1 Tactic Latency Volatility

Tactic latency is the amount of time a tactic takes to execute i.e. the span of time between the moment the tactic is called into action to the time it is finished producing an effect [56, 51, 15]. For example, a server farm detects incoming load and to manage this workload, it decides to turn on a new server. The tactic latency in this case would be the time between when the system decides to turn on a server to the moment the server is turned on and ready to accept incoming requests. Another example would be the time it takes for a system to download a file from a server across a network. Recent studies have demonstrated the usefulness of accounting for these latency times in self-adaptive decision making systems [14, 51, 56]. Many sate-of-art adaptation systems discount the use tactic latency volatility in their decision making processes by considering them as a static value i.e. one that does not change [61, 54, 55]. However, real-world self-adaptive systems will frequently encounter tactic latency volatility. For example, a UAV will not always take the same amount of time to go up and down or complete a journey. The time to destination

will depend on several factors such as wind-draft, temperature and humidity. Therefore, accounting for tactic latency volatility is one of several important reasons for a system to perform optimally, including:

1. **Determine the most appropriate tactic(s):** Accounting for tactic latency can be a significant determining factor in anticipating the most appropriate tactic to be executed by the system. It can help significantly increase the performance of a system [54].
2. **Augment a slower tactic with a faster one:** Sometimes a self-adaptive system will supplement the optimal tactic with a less optimal, but a faster one [54, 51]. This is done to maintain service and accumulate the reward of an adaptation decision while the system waits for the slower tactic to be ready for execution. The proactive nature of tactic execution can be done by accounting for tactic latency volatility and allows the self-adaptive systems to meet *SLA* requirements.
3. **Understanding when to start a tactic:** A good self-adaptive system process will be proactive with regards to its decision making. It will make optimal decisions beforehand to allow for sufficient time for tactic latency. A self-adaptive system that is unable to account for tactic latency volatility will likely make tactic decisions that start too early and incur cost [61, 51], or too late and incur penalties imposed by the *SLA*
4. **Incompatible tactics, and tactics that must be run in unison or succession:** Tactics may be incompatible with one another and might not be able to run together. Many times, self-adaptive systems would be required to deploy and run multiple tactics in unison. Suppose two or more tactics that are incompatible with each other due to use of resources, latency, or other reasons, are run in unison. In that case, it can result in undesired behaviour and loss of utility. Understanding this behaviour is very important for the operation of self-adaptive systems. When tactics are incompatible with one another, it is imperative to plan their execution ahead of time to make sure they don't cause any system instability or loss of utility. For this, accounting for latency volatility is very important in any decision-making process.

2.2 Tactic Cost Volatility

The definition of tactic cost is domain specific and varies from system to system. It can range from energy consumption to monetary value of completing a tactic. For example, a UAV would consider a tactic cost to be percentage of total energy consumed from the battery while a server farm will define it in terms of total power consumed at the wall and the resources spent cooling the computing hardware[12]. Evaluating tactic cost beforehand is a primary concern for self-adaptive system, as they sometimes have a defined resource limitation and are required to prioritize maximizing reward at the lowest cost possible [56, 55, 54]. However, despite cost volatility being a primary determining factor when it comes to tactic selection, most existing self-adaptive systems consider tactic cost as a static value and do not consider it to be volatile when making tactic-selection based decisions. Therefore, it is imperative for self-adaptive systems to consider tactic cost volatility for several reasons.

1. **Tactic Cost can be a determining factor when making tactic-selection decision:** A tactic's cost may be the determining factor for the best viable tactic a system should select and execute. Therefore, it is imperative for self-adaptive systems to accurately estimate tactic cost to make the best decision with respect to the tactic and its viability.
2. **The predicted tactic may impact the system's ability to perform subsequent or concurrent tactics:** Systems usually have limited resources and must utilize them efficiently. Therefore, it is of utmost importance to correctly predict the cost of an action to: I) Determine if there are enough resources to execute the tactic or sequence of tactics, II) Select tactics that attain the highest possible amount of utility for the system.
3. **Cost may exceed reward:** When systems determine the expected cost of executing a tactic and the operations associated with it, they also predict the reward in conjunction to assess whether the tactic is worth deploying and executing. If the cost exceeds the reward, then it may not be feasible for the system to execute the tactic in terms of the parameters defined by the *SLA*.

2.3 Uncertainty Reduction Tactics

Self-adaptive systems frequently use tactics to respond to events and achieve system goals [51, 45, 37]. *Uncertainty reduction tactics* (URT) are tactics or workflows that are designed to increase the system's decision-making knowledge and reduce its uncertainty [53] [1]. An example of an uncertainty reduction tactic is probing a sensor that has been inactive to ensure its availability. Another example is the mechanics of a motor vehicle and its fuel priming system, where the internal combustion engine is supplied with fuel when the car senses that it is about to be started. This allows the engine to start quickly and be immediately availability. Uncertainty reduction tactics can provide the system with different kinds of data depending on the domain. This can include information like the availability, health, response time, and reliability of the system.

There have been numerous uncertainty reduction tactics proposed for self-adaptive systems. However, to our knowledge, there has been very limited work done on implementing or evaluating uncertainty reduction tactics. We hypothesize that uncertainty reduction tactics can augment and supplement machine learning-based predictive processes such as eRNN to help them make more accurate tactic attribute predictions and, therefore, positively impact the system's decision-making process. We have further discussed and described the uncertainty reduction tactics used in our evaluation (model drift, sampling rate) in Section 5.1.

2.4 Evolved Recurrent Neural Networks

Recurrent Neural networks (RNNs) are generally superior to traditional statistical methods. On data that is highly non-linear, acyclic, and not seasonal, RNNs outperform models such as the auto-regressive integrated moving average (ARIMA) family of models. Furthermore,

[1] Take note, Uncertainty reduction tactics differ from the conventional tactics as defined in this study. Their primary objective is to gather data and reduce the uncertainty of the system's decision-making process. They provide the system with additional metadata regarding an operation or a tactic without necessarily exposing the system to the cost and risks of the operation or the tactic itself.

RNNs are also more suited to time series forecasting, which incorporates multiple correlated time series of input data. Classical statistical models usually struggle with such kinds of data. These features of RNNs make them a much better choice for providing predictions for self-adaptive systems.

However, building efficient and effective Recurrent Neural networks structures is a time-consuming, expensive, and relatively difficult process. RNN model building has to go through multiple iterations of training and testing before a good model can be selected for use. This is where the neuro-evolution of models comes into play. Neuro-evolution can be used to train models by selecting, connecting, and combining potential architectural components instead of simply evaluating standard RNN architectures. This results in a more comprehensive and thorough search for available model architectures and automates the design, training, and testing processes [60].

For this study, the Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm [60] was selected for the neuro-evolution process. EXAAM was chosen due to several reasons and benefits:

1. EXAAM uses a minimal seed architecture to progressively grow larger Artificial Neural Networks (ANNs). This method is similar to the popular Neuro-Evolution of Augmenting Topologies (NEAT) algorithm [70]. This method tends to generate more efficient and smaller architectures.
2. Compared to NEAT, EXAAM utilizes Lamarckian weight initialization (or the reuse of parental weights), higher-order node-level mutation operations, and backpropagation through time (BPTT) to conduct a local search. This combination of features has been shown to speed up the model training and the overall evolutionary process [22].
3. EXAAM operates by utilizing an extensible suite of memory cells not found in other ANN algorithms such as *Minimal Gated Unit* (MGU), *Long short-term memory* (LSTM), *Gated Recurrent Unit* (GRU) *Update Gate Recurrent Neural Network* (UGRNN) and Δ-RNN cells. More importantly, it evolves deep recurrent connections over large, variable time lags.

4. EXAAM has demonstrated that it can more quickly and reliably evolve RNNs in parallel compared to sequentially training traditional layered RNNs [27, 30, 29].

2.5 Motivating Example

We will be using a cloud-based multi-tier web application with a self-adaptive component as a motivating example in this study. This self-adaptive system approach is based on the works of Moreno *et al.*[51, 56]. This is a real-world use-case that is regularly utilized in real-world systems [74, 63, 54]. The goal of the system is to minimize cost and maximize utility. The application consists of Virtual Machines (VM) servers in a server farm that are handling incoming web requests. Each Virtual Machine, in this case, has a fixed cost associated with it and is added or removed to the server pool according to the traffic and load of incoming requests. The system also emulates a dimmer feature where the optional content in requests is delivered based on variable workloads, i.e., the more bandwidth available to the server, the more content is delivered. If the system is facing restricted resources, the optional content will be reduced or ”dimmed.”

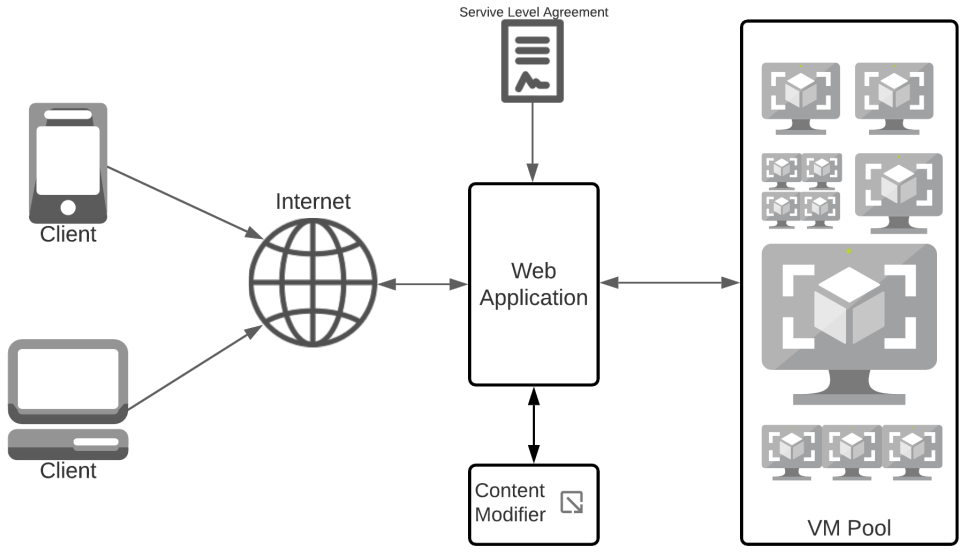

Figure 2.1: Web Application with VM Pool

The system's *Service Level Agreement* (SLA) defines how the target response time (T) and utility (U) is calculated (Equation 2.1). The system is penalized every time the target response time is not met, while it is rewarded for meeting or exceeding the target average response time against the measurement interval. The average response time is represented by r, the average response rate is listed as a, the length of each interval is defined as τ, and the maximum request is denoted as k. The cost (C) is proportional to the number of active VMs, and a dimmer (d) reduces provided content as needed.

$$U = \begin{cases} (\tau a(dR_O + (1-d)R_M)/C & r \leq T \\ (\tau \min(0, a-k)R_O)/C & r > T \end{cases} \tag{2.1}$$

In this motivating example, we can consider two tactics that the system can deploy to account for an increase in user traffic: Add a new VM server to the cluster, or II) Utilize the *dimmer* to reduce the optional content proportional to the specific response. In this case, the first tactic of adding a new server can take several minutes to execute, while the second tactic has negligible latency.

Tactic latency volatility

During operation, if the system starts to sense that the response time threshold will be surpassed in the immediate future, the system could proactively start the tactic of adding a VM server to the cluster in order to meet the SLA-defined response time threshold. If the system overestimates the latency and the system is running in the cluster longer than required, it could incur additional costs. Similarly, if the system underestimates the latency, it will incur penalties according to the SLA. Therefore, the system could take the more appropriate action of utilizing the "dimmer" tactic and reducing the optional content in the request while it waits for the new VM to be added. Incorrect tactic latency predictions can harm the system, its operation, and its ability to satisfy the SLA. It can result in scenarios where tactic execution occurs too early or too late. It could also lead to the execution of unsuitable tactics for the scenario at hand. *Accounting for tactic latency volatility is a paramount concern, especially when utilizing a proactive adaptation approach or when*

utilizing complementary tactics.

Tactic cost volatility

The cost of a tactic is vital for determining the utility of a tactic. Therefore, accounting for tactic cost volatility is of utmost importance. If the cost of executing a tactic is defined lower than what is actually experienced by the system, then it could result in scenarios where the optional content in a request is frequently being full-filled. This will incur the system penalties according to the SLA. Conversely, if the tactic's cost is defined to be higher than what the system is encountering, it could result in inaccurate utility calculation. This leads to a scenario where the optional content is being shown too infrequently even though the system has the resources to do so. This leads to less reward, and the system will again incur penalties according to the SLA. *A volatility aware solution that enables the system to more accurately predict cost will enable the system to make decisions that lead to more optimal outcomes.*

Tactic Utility

Utility is as defined as the quality or state of being useful [2]. This work expands this definition to cater to tactics. This study defines *Tactic Utility* as the usefulness of performing an action or a series of actions. Tactic Utility can be calculated by taking into consideration pre-existing conditions and attributes of a tactic(s) and a system's operating environment. This study mainly uses the equation listed in Equation 2.2 as the basis of the calculation of tactic utility. Equation 2.2 takes into account the *latency* and *cost*, which are the attributes of a tactic. It also takes into account *Reward* which is an attribute derived from the system's SLA. It refers to the potential reward the system can achieve by performing an action at a particular junction.

$$\hat{Utility} = (\frac{Reward}{(\hat{Latency} + \hat{Cost})}) \tag{2.2}$$

Uncertainty Reduction Tactics

Uncertainty reduction tactics (URTs) are system actions or workflows that are designed to increase a system's decision-making knowledge and reduce its uncertainty [53]. An example of an uncertainty reduction tactic is probing a sensor that has been inactive to ensure its availability. Another example is in terms of the mechanics of a motor vehicle and its fuel priming system, where the internal combustion engine is supplied with fuel when the car senses it is about to be started. This allows the engine to start quickly and immediate availability in terms of driving. Uncertainty reduction tactics can provide the system with different kinds of data depending on the domain. This can include information like the availability, health, response time, and reliability of the system. Uncertainty reduction tactics can provide insight into determining the internal and external variables that affect our system. In our case, predicting the cost and latency of a tactic that adds a new VM to a cluster correctly could help the system maintain smooth operation, maximize utility and minimize cost according to the parameters defined by the SLA. There have been numerous uncertainty reduction tactics proposed for self-adaptive systems. However, to our knowledge, there has been very limited work done on implementing or evaluating uncertainty reduction tactics. *Uncertainty reduction tactics may provide a lightweight, supportive input mechanism into making more accurate predictions regarding tactic volatility.*

Chapter 3

Proposed Tactic Volatility Aware Process

Traditionally, self-adaptive processes that are used by autonomous intelligent agents use *adaptation control loops* [54, 42, 36]. An adaption control loop is responsible for cycling through all possible adaption decision options at specific intervals and selecting the best adaption strategy(s) that would yield the highest utility. An adaption strategy composes of *adaptation tactics(s)*. Using the motivating example of a server farm as described in Section 2.5, the adaption control loop selects the adaption strategy, which is expected to return the highest utility. This process also takes into account tactic cost volatility and tactic latency volatility. As described in Section 2, tactic cost volatility and tactic latency volatility is a crucial component of decision making as the quality of their prediction can have a huge impact on the performance of the system, its effectiveness, and its ability to full fill its goals [61]. *The Tactic Volatility Aware Process* (TVA) described in this study does not present a new adaption-making process, but rather a more accurate information aware process that can provide higher utility when making self-adaptive decision processes.

In this section of the study, we examine how the TVA-E process fits into a workflow of a self-adaptive system. We describe the TVA-E approach in the following steps:

1. An examination of the use of prior data and uncertainty reduction tactics to Forecast Tactic cost and latency
2. The use of *eRNN* as a prediction mechanism for TVA-E processes
3. Step-by-step breakdown of TVA-E inside an adaption control loop.

3.1 Data, URTs and Forecasting

Data

TVA-E uses prior time-series data to anticipate future tactic variability. This time-series data is usually domain-specific. However, most basic forms of time series data utilized in systems include observed tactic latency and cost values. The presence of existing data is paramount for the success of TVA-E. Our TVA-E is powered by data-driven prediction models rather than a rule-based model. This allows for easy adaptation and integration of TVA-E into any adaptation control loop of the self-adaptive system. Section 4 does a further deep dive into how the data for this study was generated.

URTs

This study also incorporates the idea of *Uncertainty Reduction Tactics*(URTs). As discussed in Section 4, URTs are system actions or workflows that are designed to increase the system's decision-making knowledge and reduce its uncertainty [53]. URTs are domain-specific and therefore are integrated in conjunction with the prediction mechanism rather than being hard-wired into the TVA-E process. This also allows TVA-E processes to function in case of the absence of URTs. Section 5 further analyses the performance of TVA with and without URTs.

eRNN

Predicting the volatility of the tactic is a key aspect of a decision-making process in a Tactic Volatility Aware processes [61]. A good prediction model can have a huge impact on the system's effectiveness, resiliency, and ability to complete system and mission-critical operations [61]. For this study, we used eRNN with our TVA-E process. *eRNN* has previously shown its ability to make accurate and reliable predictions using limited data in comparison to other methods [27, 30, 29, 28]. We will also be using other comparative mechanisms such as *Long short-term memory* neural networks (LSTM), *Multi-Layer Perceptron* and a *Support Vector Regressor* (SVR). We will also be using an *Auto Regressive*

Integrated Moving Average (ARIMA) as a baseline prediction mechanism due to its use in previous studies [61]. The use of myriad prediction methods will demonstrate our TVA-E processes' ability to integrate with any suitable prediction mechanism. For this study, we will be conducting a study of how eRNN is suitable for use with TVA-E inside adaption control loops to account for tactic volatility.

3.2 Tactic Volatility Aware Process

Adaptation Control Loops

Self-adaptive systems have a control cluster which is responsible system's run-time decision-making in self-adaptation [21]. Control loops realize the adaptation of software systems and can be used in parallel and in series to each other. Adaptation control can consist of a simple sequence of four activities: monitor, analyze, plan, and execute (MAPE). These activities form a feedback control system from control theory [68]. MAPE-K [21] (*K* stands for shared knowledge base) is the most influential reference control model for autonomic and self-adaptive systems [9, 67, 7, 25]. TVA-E can easily integrate into the $Analyze$ component of MAPE-K by using existing time-series data to make predictions regarding the attributes of the tactic (*e.g.*latency, cost, *etc.*) and provides enhanced tactic values to the $Plan$ component.

TVA-E Algorithm

The TVA-E process is featured in Algorithm 1. The TVA-E process proposed by this study integrates into a system's existing adaption control loop(`L2`). It loops through all the available adaptation decision options (`L3`). While it is iterating through the possible tactic decisions, it performs predictions regarding the attributes of the tactics (*e.g.*latency, cost, *etc.*) using time-series data (`L4`). These predictions are used to determine the quality of each adaption decision option (`L5`) by plugging them into the system's utility equation as specified by the *Service Level Agreement* (SLA). These results are then used as a primary input into the system's self-adaptive decision-making process (`L6`) to select the tactic

Algorithm 1 Integration of TVA-E into Adaptation Loop

Input: Time series data
Output: Adaptation decision

```
1: —Existing adaptation control loop—
2: procedure ADAPTATION DETERMINATION
3:     for each Adaptation Option do
4:         TacticVariables ← prediction(TimeSeriesData)
5:         predictedUtility ← UtilityCalc(TacticVariables)
6:     adaptationToExecute ← getmax(predictedUtility)
7:     executeMaxAdaptation(adaptationToExecute);
8:     TimeSeriesData ← ObservedTacticAttributes()
9:     TimeSeriesData ← UncertaintyReductionTactic()
10: —Existing adaptation control loop—
```

with the highest utility or as according to the specifications in the system's SLA. Once the system executes the tactics in the strategy, the observed tactic attributes are recorded as a time-series data (`L8`). The algorithm also documents the data from the execution of the uncertainty reduction tactic in the form of a time series (`L9`). This is done for future predictions and allows the system to learn from its previous actions.

3.3 Usage

TVA-E is built by keeping integration with popular adaptation control loops in mind. TVA-E can be easily integrated into the $Analysis$ component of MAPE-K and provides enhanced tactic values to the $Plan$ component. MAPE-K [21] is the most influential reference control model for autonomic and self-adaptive systems [9, 67, 7, 25], and can easily be integrated with TVA-E. TVA-E has the ability to easily integrate into well-known adaption processes such as Proactive Latency-Aware (PLA) [54, 55, 51]. This has been enabled with the help of using existing data to perform predictions and then providing improved tactic predictions to the system's existing decision-making process. Due to these features, TVA-E

is highly applicable to a vast number of existing self-adaptive processes and systems, ranging from simple cyber-physical systems to autonomous Unmanned Ariel Vehicles (UAVs).

Chapter 4

Tactic Simulation Tool and Dataset

Self-adaptive systems react to the changes in the environment with the help of adaptation tactics. These systems frequently encounter *tactic volatility*. Academic research in this domain is observing that there is a need to account for tactic volatility [56, 61, 57, 55]. However, they also note that the research is limited by a lack of I) Simulation tools for evaluating self-adaptive processes and II) Suitable data to evaluate tactic volatile-aware processes. [61, 51]

During the course of the study, we developed a tool to assist in our experiments and address this problem. The *CELIA* (taCtic EvaLuator sImulAtor) tool emulates a simple, hypothetical intelligent self-adaptive system that autonomously performs i/o operations and computational tasks while adhering to its configurable Service Level Agreement (SLA). The main objective behind developing CELIA was not to mimic any specific system but to provide researchers with a platform with a reasonably general simulation tool and dataset that can cater to a diverse range of autonomous processes and evaluations.

CELIA assesses prediction techniques by comparing achieved vs. predicted *utility*, the number of correct vs. incorrect adaption decisions, and the achieved system goals. To assess the success of machine learning approaches, predictions created with provided or user-supplied machine learning components and data can be easily compared to the ground truth.

To summarize, CELIA consists of the following two components:

1. **Generated Tactic Volatility Data:** A publicly available dataset that will benefit researchers in their work on tactic volatility aware processes.
2. **Simulation Tool:** An easily configurable simulation tool emulating a self-adaptive system that provides researchers with a platform to evaluate their own tactic volatility aware processes.

4.1 Data Generation Component

For any research, data is very important. This study contributes to the area of Tactic Volatility with a data generation tool that accounts for a system's tactic volatility. The dataset generated by this tool is intended for high-level research and therefore does not emulate any specific system.

The operations carried out by the CELIA dataset generation tool are commonly carried out by real-world autonomous self-adaptive systems. These steps are included to replicate communication, i/o, and computational processes that are being performed by any self-adaptive system. CELIA emulation of a self-adaptive system consists of multiple steps:

1. **Download file:** : Download a compressed file hosted on multiple remote servers across the world. This emulates the tactic of communication or a file transmission operation. This operation is invaluable to the data creation process as it introduces the notion of tactic latency volatility due to the variability in network traffic and server availability encountered during each operation. CELIA provides configuration files to change the specific file to download and the servers to download from. During this operation, CELIA monitors the time taken to download a file, and the CPU resources used and considers them as Latency and Cost of doing the operation, respectively.
2. **Extract the file's contents:** Decompress the file and extract its content to the disk. This operation emulates the tactic of performing a simple file I/O operation. During this operation, CELIA monitors the time taken to extract the contents of the file and the CPU

resources used during the operation. It considers them as Latency and Cost of doing the operation, respectively.

3. **Perform Grep operation:** CELIA performs a `grep` on the extracted file contents, emulating the tactic of performing a simple function or system task. `grep` was used because it always performs a similar operation every time it searches through each file in the directory for a specific pattern
4. **Compress the file:** CELIA then re-compresses the extracted file using the `tar` command in GNU. This operation provides an additional data point of a file I/O operation.
5. **File deletion:** CELIA removes the downloaded, and the re-compressed compressed files (step 1 & 4). This operation provides an additional data point of a file I/O operation.
6. **Ping Server:** Ping the download server to check for its availability and response time. This action is an example of a simple uncertainty reduction tactic .

The data generated by CELIA is then output to a *csv* file and can be either fed into CELIA's simulation tool or used with other Simulation and statistical analysis tools. Figure 4.2 shows the workflow of the Data generation tool.

CELIA's operations are fundamentally variable and uncertain because they are carried out on "live" servers distributed all over the world, resulting in effects such as periodic latency and communication problems, just as they would in any real-world activity. We utilized mirrors of the same file situated in the USA, Canada and Switzerland, to generate our dataset. The amount of time it takes to conduct each operation is known as tactic latency, and the average CPU resources utilized for the operation is known as tactic cost. The dataset generated by CELIA includes information about all tactics and servers used through out the data generation process. Fig 4.1 demonstrates a sample of datapoints in the data generated by the tool. The dataset contains a timeseries containing the timestamps and the associated information of those timestamps. The information includes the:

1. Tactic performed,
2. Server involved (1,2,3),
3. Cost of the action,

Timestamp	Server	Tactic	Cost(%)	Latency(s)	Reliability	Ping(ms)
16:32:53.358	1	1	1.7166788	15.9	1	111.847
16:33:01.897	1	2	8.5347721	12.3	1	111.847
16:33:02.0817	1	3	0.17643117	28.8	1	111.847
16:33:08.508	1	4	6.4260485	26.7	1	111.847
16:33:17.875	1	5	9.3658041	2.1	1	111.847
16:33:24.318	2	1	1.7081530	5.3	1	85.731
16:33:27.836	2	2	3.5130341	25.4	1	85.731
16:33:28.017	2	3	0.1732807	25.0	1	85.731
16:33:34.476	2	4	6.45843195	26.6	1	85.731
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Table 4.1: This table provides the sample of the dataset. Each datapoint has an associated information of the operation performed at that moment. Take note, the ping column is concatenated with the main log later on as it is monitored separately. The ping column represents a URT.

4. Latency of the action,
5. Ping of the server,
6. Reliability of the action (*i.e.*whether the action was successful or not).

The dataset contains subsets of these repeated actions where each subset represents one operation cycle of the data generation tool (demonstrated in figure 4.2). For the purpose of the experiment, only tactic 1 was used from each of these subsets due to time and resource constraints. We only utilized the 1st tactic (*Download File*) because it demonstrated the highest amount of volatility. After cleaning and preparing the dataset, 52,106 tactic-1 records were used from all the data obtained after iterating over the system operations for around two weeks. We discarded the Reliability feature which specified whether a specific action failed or not. The only volatile action or tactic in our process was *Download/Tactic-1*, which had a failure rate of less than 0.01%. Therefore, we didn't use it in our study. The Ping feature was used as an uncertainty reduction tactic. Figure 4.1 demonstrate the spread of the *Cost* and *Latency* features in our dataset using box plots. Table 4.1 breaks down the dataset in detail and discusses the spread of *Cost* and *Latency*.

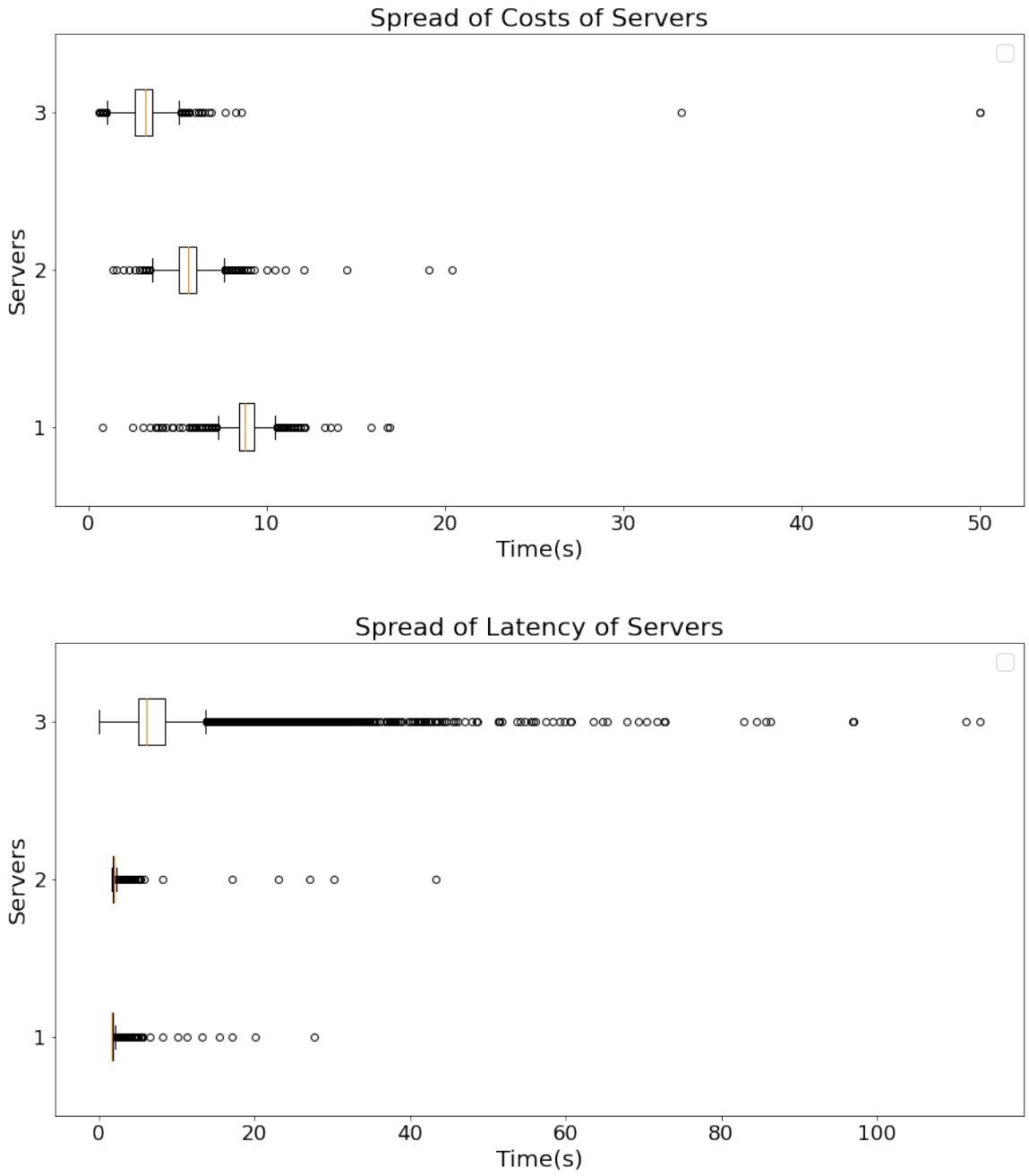

Figure 4.1: The box plots show the spread of the data. The physical distance of the servers is demonstrated in the spread of costs and latencies of the servers. Server 1 is USA and thus has the lowest center of mass. While Switzerland is the furthest away and its center of mass is the highest. The cost also corroborate the nature of tactic volatility of our dataset. This is because downloading a file from Switzerland takes the longest time and results in the average CPU usage being low.

	Latency				Cost		
Server	**1**	**2**	**3**		**1**	**2**	**3**
mean	1.800578	1.953712	8.042404		8.939763	5.673922	3.130603
std	0.432465	0.595287	5.917548		0.692191	0.800211	1.029041
min	1.678576	1.684291	0		0.8	1.4	0.6
25%	1.704469	1.836947	5.08959		8.5	5.1	2.6
50%	1.740055	1.958758	6.150736		8.8	5.6	3.2
75%	1.844233	2.033431	8.576069		9.3	6.1	3.6
max	27.771008	43.323958	113.186285		16.9	20.4	50

Table 4.2: Various metrics regarding the dataset we used. The dataset consists of real-world data of Tactic-1, captured during the operations of the Data generation tool. The latency is in seconds while the cost is a percentage value of the CPU usage. The differences among the standard deviations of the servers demonstrates the volatility in our dataset.

CELIA is a data creation process and a simulation tool that improves upon the features of existing simulation software. For example, the *Rice University Bidding System* (RUBiS) [3] and *'ZNN.com'* [19] are multi-tiered web programs that have been used to evaluate self-adaptive processes but do not feature an adaptation component. DARTSim [50] does not include volatility in its decision-making process and solely considers static latency. Existing resources, such as 'The Internet Traffic Archive' [1], do not contain both latency and cost for executed actions. Hence CELIA's data is critical. The project website [1] provides our produced data, source code, and Docker image.

This tool and dataset are the first to include uncertainty reduction tactics, as far as we know. The uncertainty reduction tactic performs the data gathering operation of probing a remote server, collecting information about its availability and response time for the demonstration of capabilities conducted in this work. Future researchers and practitioners can use our CELIA tool to incorporate their own uncertainty reduction tactics. The precise functionality conducted by uncertainty reduction tactics will be domain-specific, as with most tactics in each implemented system, and it is, therefore, unreasonable to assume

[1] https://github.com/valet-tool/valet-tool

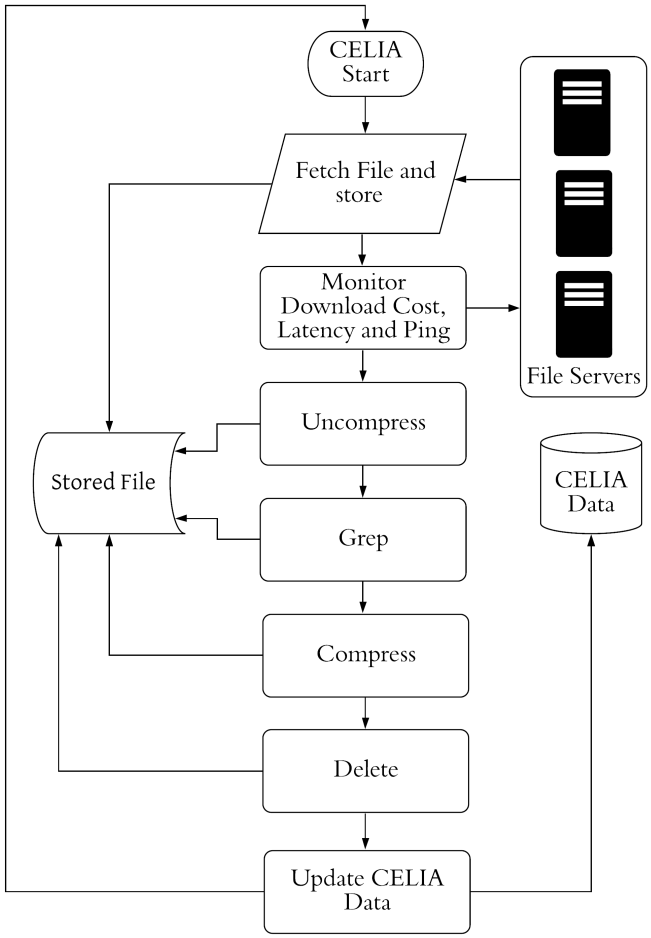

Figure 4.2: CELIA ETL data generation process

that any evaluation system could account for all the functionality and variations of uncertainty reduction operations in all systems. Furthermore, one of the goals of this project is to demonstrate the fundamental capabilities of uncertainty reduction operations; it is not intended to be a comprehensive assessment of every type of uncertainty reduction tactic.

4.2 Simulation Component

The simulation component of CELIA makes it possible to evaluate tactic volatility-aware processes in a robust and simple way. The tactic volatility data, *utility* equation and SLA values can be customized according to need.

CELIA simulates an autonomous intelligent agent that is responsible for making critical decisions with system-wide dependencies. The goal of CELIA's simulation component is to collect an updated version of a file from a remote location on a regular basis in order to maximize *utility*.

CELIA's simulation component iterates through provided data records making decisions with the *utility* equation serving as a determining factor as to the decisions made by the system. The system evaluates the user-supplied tactic volatility data from a prediction algorithm against the user-supplied ground truth data and observes different metrics resulting from the different data points. The primary components of the CELIA simulation component are highlighted in Figure 4.3 and are described below:

1. **Time-series data:** CELIA's ability to evaluate various tactic volatility-aware processes is one of its main advantages. The expected tactic attributes (e.g., latency, cost, *etc.*) that each tactic operation will encounter are a central component of many self-adaptive processes. CELIA allows users to use their own time-series-based data. The CELIA simulation tool takes as input two sets of time-series data. The first dataset is the data from a prediction algorithm that will be used to make adaptation decisions. The second is the ground truth dataset which will be used to analyze the performance of the system on the prediction dataset.
2. **Calculate Utility:** The goal of CELIA's simulation component is to collect an updated version of a file from a remote location on a regular basis in order to maximize *utility*. CELIA uses the equation supplied in Equation 4.1 to determine whether the expected *utility* in comparison to update threshold warrants the file download and extraction operations, or if the system should 'pass' and wait for another decision-making iteration.

$$\hat{Utility} = (\frac{Reward}{(\hat{Latency} + \hat{Cost})}) \tag{4.1}$$

3. **Adaptation Decision:** The expected *utility* is measured against a pre-determined threshold value according to the system's SLA. If the *utility* to perform the update is greater than the pre-defined threshold value in the SLA, the system performs an update.

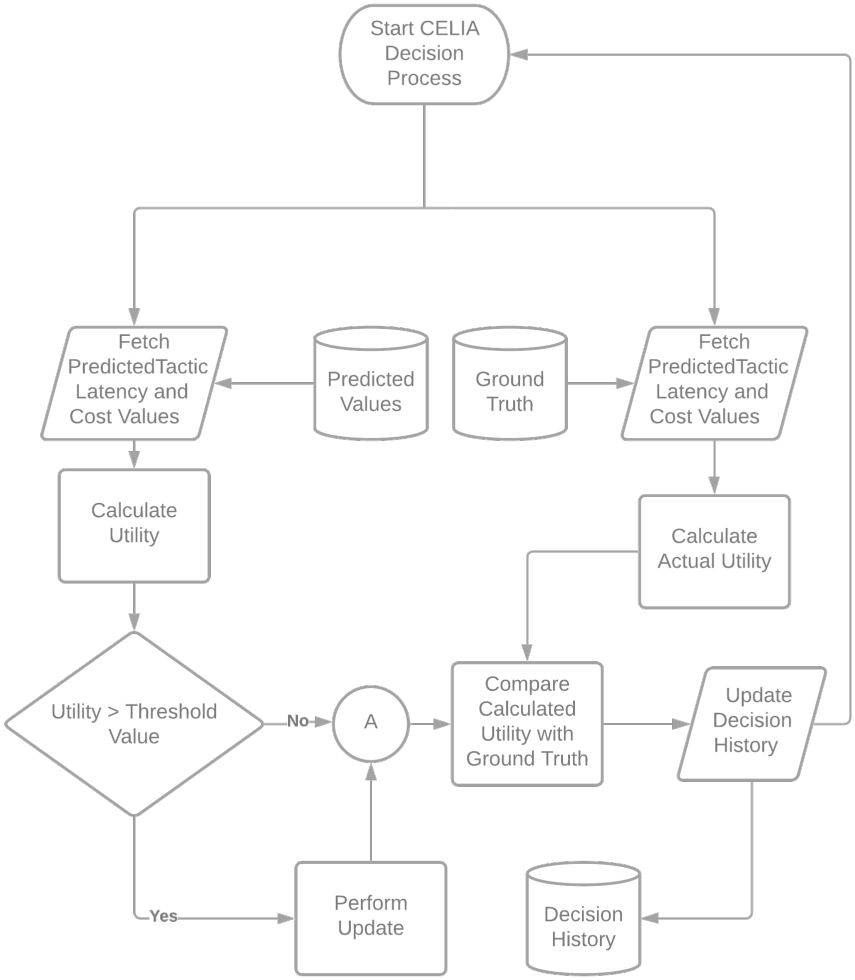

Figure 4.3: CELIA Simulation Component

If it is lower than the threshold value, it ignores the update.The *Reward* for performing an update increases as the time since the last update is performed increases. This feature has been implemented to emulate a time-based risk factor into CELIA's process that penalizes the system for making a bad decision.

4. **Assess evaluated TVA process:** The simulation component takes two files as input when assessing whether the system should undertake tactic operations:

   (a) Projected values for each tactic attribute (delay, cost), and

   (b) Observed tactic attributes (which serve as the ground truth).

With the help of the ground truth values, CELIA is able to provide several evaluation metrics to assess the ability of the evaluated tactic volatility aware process. These metrics add to the robustness of the evaluation by including metrics like achieved utility and frequency of correct decisions. This allows for a comparison of techniques in a more real-world setting rather than relying solely on statistical analysis. The following are the evaluation metrics:

(a) **Expected vs. achieved utility:** The system's ability to achieve the predicted utility level is recorded. Accurately anticipating the achieved utility is critical for systems in their decision-making processes when deciding on the best tactic(s) and adaptation strategies for the situation at hand [57, 56, 51].

(b) **Ability to make correct decisions:** Self-adaptive systems must be able to make a variety of decisions correctly. CELIA identifies when the system fails by comparing predicted vs. ground truth tactic attributes.:

i. Performed an action when the system should not have.

ii. Performed an action when the system should have.

iii. Didn't perform an action when it should not have.

iv. Didn't perform an action when it should have.

(c) **Expected utility not achieved:** Inaccurate tactic predictions can lead to situations where the system anticipates gaining a certain amount of utility. However, due to inaccurate tactic predictions, this utility will never be realized, resulting in a number of adverse repercussions, including an increase in general uncertainty and unexpected outcomes [55, 32].

(d) **Utility missed by taking incorrect action:** When systems fail to perform actions when they should, they may experience 'missed opportunity' utility. For example, if a system should have performed action X, which would have provided a utility of 5 points, but instead chose incorrectly not to do so, the system would have lost an opportunity to accrue a utility value of 5.

5. **Record output:** CELIA outputs the results of this evaluation in a .csv file.

6. **Perform next adaption decision-making iteration:** The system will then perform the next adaptation loop, and the adaptation and evaluation procedure will be repeated indefinitely until all provided data has been iterated.

4.3 Usage

CELIA can help researchers in a variety of self-adaptive research domains. The tool will help researchers evaluate tactic volatility aware processes and their capacity to operate effectively and resiliently. Adopters will be able to modify the tool due to its easily configurable nature, such as its utility equation, SLA, and adaption processes. This will allow researchers to test how changes to the self-adaptive decision-making process affect the system's capacity to respond appropriately to tactic volatility.

For this study, we used the different prediction mechanisms (described in Section 5.1) as the input for CELIA. For this analysis, we used values with both with and without uncertainty reduction tactics. The output from CELIA is later used to perform an analysis in this study.

Chapter 5

Evaluation

Our evaluation addresses the following research questions:

RQ1. *Is eRNN effective for predicting tactic volatility?* We demonstrate the effectiveness of eRNN for predicting tactic volatility. With the help of uncertainty reduction tactics, eRNN is able to out perform other prediction mechanisms when predicting latency by 6%(*MAE*) and cost by 46%(*MAE*).

RQ2. *Does eRNN effectively transfer the tactic volatility predictions to effective decisions compared with other baselines?* We demonstrate that eRNN is the best method for predicting volatility compared to other prediction techniques. We also observe that uncertainty reduction tactics provide useful information that can help the system to make more accurate decisions.

RQ3. *Are Uncertainty Reduction Tactics effective in helping to predict tactic volatility?* This work demonstrates the potential benefits of uncertainty reduction tactics in self-adaptive systems. Specifically, helping to make more accurate predictions regarding tactic latency and cost. While the potential benefits of *Uncertainty Reduction Tactics*(URT) are model-specific, eRNN is able to leverage it by improving its *Mean Squared Error*(MSE) by 50%. URTs act as regularization for model training and help make models more generalizable.

5.1 Experimental Design

5.1.1 Prediction Mechanism

We tested the following prediction mechanisms against our eRNN solution to see how effective it is in comparison to other options:

1. **Long Short-term Memory recurrent neural networks (LSTM)**: The Long Short-term Memory (LSTM) used was a fully connected model with one hidden layer. The hidden layer consisted of 1,000 LSTM nodes followed by an output layer. We used LSTM due to its gated memory cells that utilize information from previous network pass-throughs to make current predictions [24, 64]. It is an efficient and extremely powerful Recurrent Neural Network*(RNN)* for forecasting sequential time-series multivariate data.
2. **AutoRegressive Integrated Moving Average (ARIMA)**: ARIMA stands for Autoregressive(AR) Integrated(I) Moving Average(MA). It is also denoted by ARIMA(p, d, q) where p stands for the number of lag observations included in the model(AR), d represents the degree of differencing(I), and q represents the size, or order of the moving average window(MA). For our experiment, we used ARIMA with a differencing order of d = 1 as the time-series data was non-stationary. Autocorrelation plots were used to determine autoregressive (p=1) and moving terms (q = 0). We incorporated ARIMA in this work due to its use is previous studies [61] and used it as a baseline.
3. **Support Vector Regressor with a Radial Basis Function kernel (SVR-RBF)**: The Support Vector Regression *SVR* model was used with a radial bias function *RBF* and a linear kernel as standard machine learning model. For the kernel function, we used the default values of 1.0 and 0.1 for the regularization and other hyper-parameters. We used *(SVR)* in this study as it is a sophisticated model that performs well on time series data predictions [47, 59].
4. **One layer Multi-layer Perceptron (MLP)**: The Multi-Layer Perceptron *MLP* neural

network model was a fully connected model with one hidden layer of 100 nodes and an output layer that predicted cost and latency. Multi-layer Perceptron *(MLP)* are basic feed-forward neural networks that consist of fully connected hidden layers and do not utilize recurrent connections (*e.g.*data from previous passes through the network), and will serve as a baseline neural network model.

We compared these techniques to our eRNN-based prediction mechanism due to their prevalence in the research space and their prowess at time-series forecasting [6]. For this study, we used a total of 52,106 records. From this 36,472 records were used for model training, 7,819 were used for testing, and 7,815 were reserved for validation of the models. The EXAMM(eRNN) models we used for the study were very efficient. The largest one only had 44 nodes in total and 590 weights. All the models, including ARIMA were trained using the training dataset. The testing dataset was used to optimize the hyper-parameters and converge on the optimal model and stop the training process. The validation dataset was used to gauge the viability of the models and make sure they were generalized. It was not used in the training process.

5.1.2 Uncertainty Reduction Tactics

We evaluated the benefits of using uncertainty reduction tactics in conjunction with our eRNN-based methodology after proving the benefits of eRNN compared to other evaluated machine learning options. We only used eRNN to analyze URTs because it was found to be the most effective prediction mechanism that could use URTs. Moreover, ARIMA is a univariate model that can't take advantage of data from other sensors (URT # 1). Therefore, it was not considered in the URT analysis.

This evaluation was carried out using two uncertainty reduction tactics that were chosen for their real-world applicability [48, 72, 35, 8] and discussion in a prior paper [53]. In our evaluation, we employed the following two URTs:

URT #1: Reducing uncertainty due to model drift - Due to various forms of internal

and external volatility, the models utilized by a self-adaptive system may progressively become incompatible with the system's environment. Additional sensors or data collection operations can be used to address this uncertainty [8, 17, 75, 53]. We simulate this uncertainty reduction activity in our evaluation by pinging the remote server on a regular basis to obtain information about its availability and response time. This data is subsequently fed into the tactic prediction mechanism of the system.

URT #2: Changing the sampling rate of a parameter - A confidence interval for the value of the monitored parameter can be generated using the mean and variance of the observations. One way of controlling the width and uncertainty of the interval is to adjust the sampling rate [8, 35, 72, 48]. We emulated this uncertainty reduction tactic by incorporating every *n*th observation from the data into our prediction process. The *n*th value would be determined according to system-level agreement *SLA* and the system specifications. For this evaluation, we chose *n* as 5, 10, and 20. These values were chosen to represent the loss of information that would occur if the sampling rate was changed in relation to the size of the dataset. Due to the size of our data, values below 5 did not result in a significant reduction in sample size. We also chose a sampling rates of 10 and 20 by doubling the previous sampling rate. We didn't use values greater than 20 because our dataset became too sparse after doubling the sampling rate to 40. URT2 is a versatile framework that enables us to tailor the best URT technique to individual learning problems. For optimal prediction improvement, we use extensive cross-validation and use n=5 for latency prediction and n=20 for cost prediction in this experiment. Considering utility is only decided by two parameters (*i.e.*latency and cost), the benefit of such flexibility may not be significant in this study. However, in a more complicated simulation environment, where utility is determined by hundreds or thousands of factors, we can employ customized sample rates to reduce the uncertainty in each factor input, resulting in a significant impact on the utility score.

5.1.3 Evaluation Criteria

In our analysis, we employed the following evaluation criteria

1. **Ability to accurately predict tactic attributes:** The predicted features of a tactic have a substantial impact on the system's decision-making process since they can directly influence if and when a tactic is picked and implemented [51, 57, 55]. As a result, a system's ability to properly forecast the attributes of a tactic is critical for ensuring effective, efficient, and robust operation [61]. For our evaluation, we compared the predicted tactic latency and cost values against the observed ground truth values.
2. **Expected vs achieved utility:** We looked at the system's ability to deliver the anticipated amount of utility while operating in a dynamic environment by using variable data. Accurately anticipating the achieved utility is very critical for a system's adaption-decision process to choose the most appropriate tactic(s) [57, 54, 56, 38, 51]. For our evaluation, we compared the predicted utility with the actual utility (ground truth) of performing adaption decisions and tactics.
3. **Ability to make correct decisions:** The ability to make accurate and correct decisions is critical for self-adaptive systems and processes. Therefore, we collected data for the following system actions:

   (a) Performed an action when the system should not have.
   (b) Performed an action when the system should have.
   (c) Didn't perform an action when it should not have.
   (d) Didn't perform an action when it should have.

   For our evaluation, we compared the action that the system should have taken by using observed values (ground truth) against the actions recommended by the prediction system. We performed qualitative and quantitative studies of the impact of each prediction mechanism on a system's decision-making process.

5.1.4 Analysis Using Simulation Tool

Using the CELIA simulation tool and dataset (Section 4), we compared our TVA-E procedure to existing techniques. A simulation tool gives a more robust evaluation than a

statistical study, offering measures like frequency of correct decision and utility achieved from making decisions. To emulate variations in system design, and reflect more realistic scenarios and environments, we ran simulations against varied threshold values (Section 4).

For this study, we used the 'download' tactic described in Section 4. This process involves repeatedly downloading the Apache installation file [1]. We chose this technique to evaluate the ability of the methods under consideration because it featured the most real-world volatility, emulating the behaviors that a file transfer activity in a self-adaptive system would resemble. Due to a variety of reasons, such as variability in network traffic and server load, this tactic has constantly exhibited real-world volatility. Additional grounds for focusing on this technique were: I) the server's latency and cost variations were sufficient for a proper evaluation, and II) to simplify the analytic output/examination. Overall, we used a total of 52,106 tactic operations in our evaluation and measured the latency and cost of performing this action.

We used three different server locations (Switzerland, Canada, USA) from across the world that host the Apache installation file to collect the tactic operation's cost and latency values. We chose these particular servers due to their prevalence and their location. The physical distance was bound to introduce variability in a myriad of factors (*e.g.*network traffic, server resource availability, *etc.*) and experience real-world volatility. We are making this dataset available for use by anyone to promote further studies in this domain.

5.2 Experimental Results

RQ1: Is eRNN effective for predicting tactic volatility?

In this study, we demonstrate eRNN's capacity in general prediction processes and those processes that benefit from accounting for tactic volatility. Predicting the volatility of the tactic is a key aspect of decision-making in a Tactic Volatility Aware processes [61]. A good prediction model can have a huge impact on the system's effectiveness, resiliency,

[1] https://downloads.apache.org/httpd/httpd-2.4.43.tar.gz

and ability to complete system and mission-critical operations [61]. For this study, we used eRNN with our TVA-E process.

We used the data from the CELIA data creation tool to train various models along with eRNN. The dataset was split into three different sets: training, testing, and validation. After the training and testing of these models, we evaluated their performance using the validation dataset.

We used the mean squared error (*MSE*) to evaluate the predicted tactic cost and latency against the ground truth. MSE is a well-used loss function for regression tasks. We also used Mean Absolute Error (*MAE*) as an alternate error measurement because MSE can be sensitive to the scale of data and outliers. This will allow us to comprehensively study and compare the results. *MSE* and *MAE* are defined as following:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - \hat{t}_i)^2 \qquad MAE = \frac{1}{N}\sum_{i=1}^{N}|t_i - \hat{t}_i|$$

Where N is the total number of data points, t is the ground truth value (*e.g.*cost or latency), and $\hat{t}_i$ is the predicted value. We tested each model on the three different sets of data from the Apache download servers described in Section 4, and reported the performance of each model. Note that ARIMA utilizes uni-variate time-series data to make predictions, and therefore data from uncertainty reduction tactics can not be incorporated into the model. Consequently, the results for that are not reported.

After taking into consideration Table 5.1, 5.2 & Figure 5.3, we came to the following conclusions:

- Table 5.1 and 5.2 show that eRNN has the best performance in terms of predicting latency and cost. The uncertainty reduction tactic (URT) does not benefit eRNN in improving its predictive power in terms of MSE, but it does so in terms of MAE.
- The predicted latency has a large number of outliers in the observed values, as shown in Figure 5.2 (ground truth). We, therefore, report the MAE for each compared model to better evaluate the model in this case because it is less sensitive to outliers. From

Table 5.1: Demonstration of the capabilities of eRNN in predicting tactic latency in relation to compared methods. $\overline{MSE}$ of eRNN with URTs is around 7% than without URTs applied. $\overline{MAE}$ of eRNN with URTs is around 6% than without URTs applied. It also consistently matches or outperforms other predictive mechanisms at predicting tactic latency.

URT Tactic	Model	$\overline{MSE}$	$\overline{MAE}$
noURT	**eRNN**	**0.00061**	**0.016**
	ARIMA	0.00069	0.017
	LSTM	0.00083	0.016
	MLP	0.00086	0.016
	SVR_Linear	0.0087	0.035
	SVR_rbf	0.009	0.036
URT_Combine(Sample rate=5%)	**eRNN**	**0.00057(7%↑)**	**0.0015(6%↑)**

Table 5.2: Demonstration of the capabilities of eRNN in predicting tactic cost in relation to compared methods. $\overline{MSE}$ of eRNN with URTs is around 65% than without URTs applied. $\overline{MAE}$ of eRNN with URTs is around 46% than without URTs applied. It also consistently matches or outperforms other predictive mechanisms at predicting tactic cost.

URT Tactic	Model	$\overline{MSE}$	$\overline{MAE}$
noURT	**eRNN**	**0.00032**	**0.013**
	ARIMA	0.00027	0.013
	LSTM	0.00051	0.014
	MLP	0.00085	0.014
	SVR_Linear	0.0028	0.052
	SVR_rbf	0.0022	0.052
URT_Combine(Sample rate=20%)	**eRNN**	**0.00011(65%↑)**	**0.007(46%↑)**

a more aggregate level, a small percentage of outliers would likely have little impact on the system's functionality (e.g., it doesn't matter if the prediction that leads to a non-optimal action is slightly wrong or very wrong; the incorrect action was still taken). RQ2 expands on our claim that eRNN's higher MAE does not prevent it from making effective decisions.

- The added uncertainty reduction tactic information has a major impact on eRNN's cost prediction. When uncertainty reduction tactic information is used to train the eRNN, the MSE is lowered by around half.

- Using the uncertainty reduction tactic information during training also serves as regularization for the models. Figure 5.3 demonstrates that eRNN often provides more predictions that are close to the outliers without uncertainty reduction tactic information (which are likely to systematic noises). Using uncertainty reduction tactics allows us to address overfitting issues which can cause a problem with the generalizability of models.
- eRNN also generates networks that are significantly more efficient in terms of the number of nodes. EXAAM is not only able to compete and beat the other models in terms of predictions, but it also uses an order of magnitude lesser number of nodes. This highlights the inefficiency of using traditional fixed neural network model architectures. Due to the varying limitations of embedded systems like power, storage, and size, it can be highly beneficial to use eRNNs in place of other models as they can be easier to integrate with such systems [44, 71, 11].

Outcome: *Our findings demonstrate that eRNN is effective at both making predictions, and specifically at predicting tactic cost and latency volatility.*

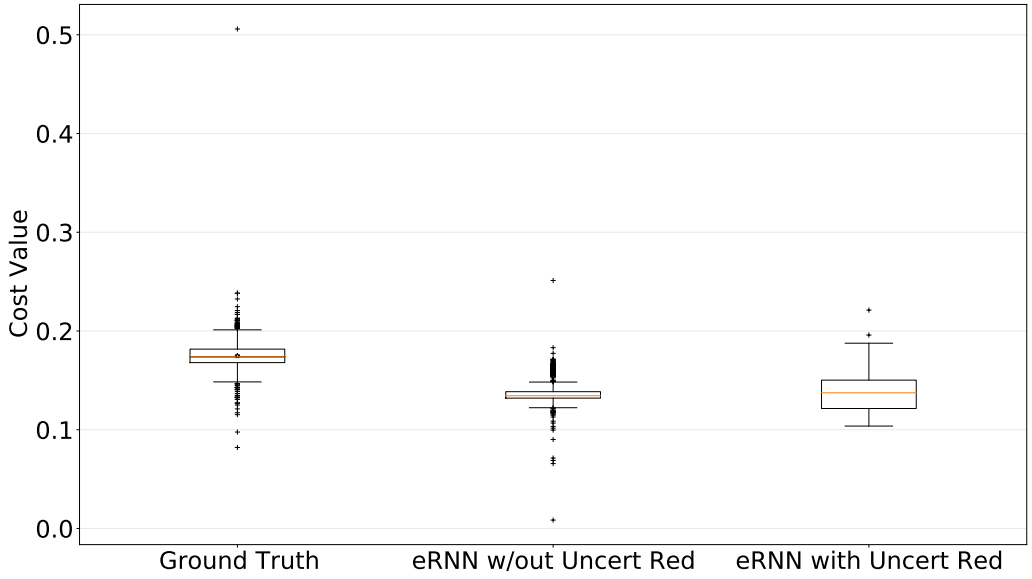

Figure 5.1: Predicted tactic cost(on server1)

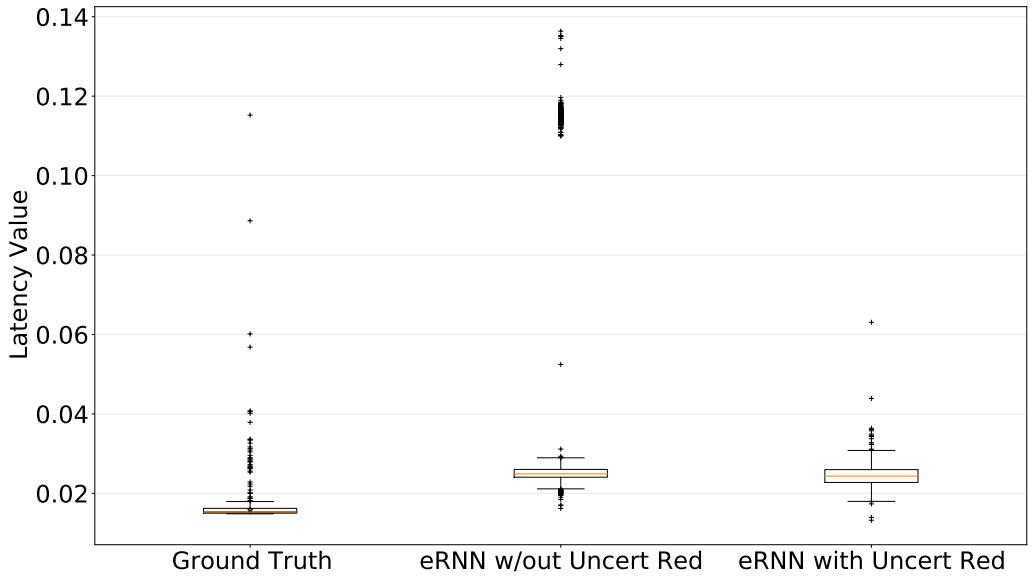

Figure 5.2: Predicted tactic latency(on server1)

Figure 5.3: Demonstration of the ability of uncertainty reduction tactics to make the prediction distribution similar to the ground truth with sparse outliers.

RQ2: Does eRNN effectively transfer the tactic volatility predictions to effective decisions compared with other baselines?

The predicted utility's value is not well scaled, unlike tactic cost and latency, which have values that are commonly dispersed around 1 (range from 0 to 16,441.9). As a result, we evaluate the utility prediction through using a scale-independent measurement rather than MSE: mean absolute percentage error (MAPE). The MAPE formula is as follows:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}|\frac{t_i - \hat{t}_i}{t_i}| \tag{5.1}$$

As shown in Table 5.3 the MAPE exceeding 100%, suggests that the model is likely to overestimate the utility, particularly when the actual utility is minimal (close to zero). This is typically seen as a major prediction error. Large prediction failures have a high chance of leading to inaccurate adaptation decisions in self-adaptive systems. We discovered that eRNN performs comparably to state-of-the-art algorithms like ARIMA, LSTM, and MLP and that its performance can be improved further using URT operations.

For systems to choose the most relevant tactic(s) and adaption techniques for the observed scenario [57, 54, 56, 38, 51], it is critical to accurately predict utility. The benefits can be improved even more by using the uncertainty reduction data during the training phase, particularly for eRNN, where the URT improves by 24

final decision-making procedure can be thought of as a binary classification exercise. As a result, the system can be evaluated as a binary classifier. We'll start by defining a few terms. The number of correct 'update' decisions made by the system is represented by the true positive (TP). The true negative (TN) represents the system's correct 'not update' decisions. The number of inaccurate 'update' decisions is the false positive (FP), and the number of incorrect 'not update' decisions is the false negative (FN). We can assess the classification accuracy of the system output with the help of these metrics. This accuracy calculation indicates the percentage of correct decisions in the system output. We use false positive rate ($FPR = \frac{FP}{TP+FP}$) for wrong decisions and false negative rate ($FNR = \frac{FN}{FN+TN}$) for evaluating the frequency of wrong 'update' and 'not update' decisions. Using

Table 5.3: Evaluation of utility prediction, demonstrating the how predicted utility deviates from the true utility. This utility calculation is from the Simulation tool which uses the Utility equation. The utility equation is a product of two functions. Therefore, for analyses of the differences in utility among the prediction mechanisms, we utilized MAPE to scale it proportionally.

Model	$\overline{MAPE}$
eRNN	21%
ARIMA	13%
LSTM	16%
MLP	14%
SVR_linear	57%
SVR_rbf	57%
eRNN+URT_Combine (Sample rate=5)	16%

Table 5.4: This table compares the decisions made by the CELIA simulation tool using the evaluated prediction mechanisms. The CELIA simulation had a reward threshold of 1000 in the SLA. This table breaks down the decisions according to different evaluation criteria. This table shows how eRNN with URT has the lowest False Positive Rate.

Model	$\overline{FPR}$	$\overline{FNR}$	$\overline{U\_gain}$	$\overline{U\_loss}$	$\overline{Acc}$
eRNN	15%	7%	2.10E+07	1.40E+05	88%
ARIMA	6.6%	5.6%	2.1E+07	9.3E04	94%
LSTM	7%	11%	2.00E+07	1.90E+05	91%
MLP	9.20%	6%	2.00E+07	9.00E+04	92%
SVR_linear	0%	54%	2.00E+07	1.80E+06	59%
SVR_rbf	0%	56%	2.00E+07	1.90E+06	58%
eRNN+URT_Combine (Sample rate=5)	0.53%	17%	2.30E+06	3.40E+02	99%

Table 5.4, we can conclude:

- In our eRNN-based TVA-E approach, the uncertainty reduction information helps improve accuracy by 13%, resulting in the highest accuracy.
- The FPR of the eRNN-based process is the lowest. This is in line with the prior observation about the utility prediction's small MAE. Our eRNN-based approach is less likely

to deliver a false update tactic action since it rarely overestimates small utility values.

- When compared to other baselines, the eRNN-based system has a moderate FPR and FNR but low prediction accuracy. We see that URTs are able to rebalance the two types of errors (FPR and FNR) in such a way that the overall benefit (accuracy) is maximized. If the amount of utility missed by incorrect 'not update' decisions is small, URT chooses to tolerate an increase in FNR in the rebalanced decision-making procedure because this type of mistake will only have a minor negative impact. Utility loss can be used to quantify the negative impact. The utility loss is the sum of all incorrect 'not update' decisions' ground truth utility scores. The utility loss is an intuitive measure of how much utility a system could have gained if the system made the right 'update' decision. It can be argues that eRNN has a relatively higher FPR. However, despite the higher FNR value of eRNN+URT, it lost the least amount of utility compared to others. It is able to access the risk of not updating and "plays it safe" while not losing a lot of utility.
- The eRNN-based system's high FNR rate has a minor negative impact on final decision-making. When compared to other methods, the false-negative decisions made by an eRNN-based system have the least utility loss.
- Our eRNN-based TVA-E process can be used with uncertainty reduction information to reduce utility loss to 0.1%. This means URTs helps eRNN to minimize the expected loss during the decision-making process. Similar to utility loss, we can use utility gain to quantify the positive impacts made by correct decisions. The utility gain is the sum of the true utility scores for all correct 'update' decisions.

Our general observations are that:

- For latency and cost predictions, eRNN is the best candidate. The model's accuracy and robustness are both excellent.
- The strength of eRNN cost and latency prediction can be well transformed to utility prediction.
- Our eRNN-based decision-making process not only makes the best decisions, but it also maximizes utility gain and minimizes utility loss.

- The uncertainty reduction information can be used to regularize model prediction, and reduce model output uncertainty.
- Uncertainty reduction information can be used to reduce utility loss.

Outcome: *In comparison to existing prediction techniques, our analysis shows that eRNN is the best method for predicting volatility. We also notice that uncertainty reduction tactics provide useful information that can aid the system in making more accurate decisions.*

RQ3: Are Uncertainty Reduction Tactics effective in helping to predict tactic volatility?

While the potential benefits of uncertainty reduction tactics have been explored previously [53], no documented efforts have been made to demonstrate or evaluate their potential benefits in simulated self-adaptive systems. As discussed earlier and demonstrated in RQ1 & RQ2, we discovered that uncertainty reduction tactics helped eRNN deliver improved tactic latency and cost predictions.

- The effectiveness of URT varies by the prediction model. While all of the prediction models improve when URT is added, eRNN takes advantage of the uncertainty information and improves its MSE by 50%.
- Using uncertainty reduction tactic information when training model also serves as regularization. As demonstrated in figure 5.3, eRNN predictions tend to be relatively more inaccurate and closer to the outliers without uncertainty reduction tactic information. Using uncertainty reduction tactic information allows eRNN to address this issue and perform a very well.

Outcome: *This study highlights the potential benefits of uncertainty reduction tactic, particularly in terms of making more reliable predictions regarding tactic volatility.*

5.3 Discussion

General application of TVA-E

Self-adaptive systems will be required to work efficiently, effectively, and resiliently in volatile environments, as they become more ubiquitous. This research highlights the importance of self-adaptive systems in properly accounting for tactic volatility. Accounting for tactic volatility has been shown to reduce uncertainty and enhance a system's ability to make more optimal decisions. Self-adaptive systems with an adaptation control loop will be required to routinely make predictions regarding the attributes of a tactic. These predictions will allow the system to understand its current and future environment and, therefore, will impact the current and future state of the system. TVA-E decently complements a MAPE-K adaptation control loop. It is able to account for tactic cost and latency volatility and help make better decisions regarding tactic adaptation and deployment. It does so by using data-driven prediction and uncertainty reduction tactics. The nature of TVA-E also allows system administrators to integrate additional tactic attributes besides cost and latency.

Processes such as those targeting the reduction of large adaptation spaces [73] may be able to benefit from and integrate with our findings. Our work can help in this area by providing systems more accurate tactic information that can be used to both eliminate improbable adaptation options (e.g., those that are impracticable due to cost/latency) and improve the utility equations that are used to choose the best adaptation options.

eRNN

Predicting the volatility of the tactic is a key aspect of a decision-making process in a Tactic Volatility Aware processes [61]. A good prediction model can have a huge impact on the system's effectiveness, resiliency, and ability to complete system and mission-critical operations [61]. Recurrent Neural networks (RNNs) are generally superior to traditional statistical methods. On data that is highly non-linear, acyclic, and not seasonal, RNNs outperform models such as the ARIMA family of models. Furthermore, RNNs are also more

suited to time series forecasting, which incorporates multiple correlated time series of input data. Classical statistical models usually struggle with such kinds of data. These features of RNNs make them a much better choice for providing predictions for self-adaptive systems.

However, building efficient and effective Recurrent Neural networks structures is a time-consuming, expensive, and relatively difficult process. Neuro-evolution algorithms allow for selecting, connecting, and combining potential architectural components instead of simply evaluating standard RNN architectures. This results in a more comprehensive and thorough search for available model architectures and automates the design, training, and testing processes.

For this study, we used The Evolutionary eXploration of Augmenting Memory Models (EXAMM) [60] or eRNN (as referenced throughout this study) with our TVA-E process and compared its performance against other methods. This work shows the value of eRNN in anticipating tactic volatility in self-adaptive systems, as well as the general capabilities of our TVA-E method based on eRNN. This study exhibits eRNN's capabilities and lays the groundwork for its future use in other fields of machine learning. As a result, the findings of this study promote research not just in self-adaptive systems but also in a wide variety of other fields where machine learning may be used.

URTs

While the advantages of uncertainty reduction tactics have been explored from a theoretical standpoint [53], this is the first attempt to test them in a simulated environment. This study lays the groundwork for future research, such as I) investigating uncertainty reduction tactics in more simulated and physical locations and settings, II) further evaluating them, and III) developing new uncertainty reduction tactics for a wide variety of applications. The benefits of the supplemental information offered to eRNN through uncertainty reduction strategies have been established, demonstrating the value of this additional information to this model. Uncertainty in information is widely acknowledged as being harmful

to decision-making processes in systems [53, 16, 34], and this work shows the value of uncertainty reduction tactics in reducing the amount of uncertainty encountered by a system.

CELIA

This experiment also highlighted the usefulness of CELIA's simulation tool and data creation tool. The demand for tactic volatility-aware processes will grow in tandem with the growing recognition of the need for systems to function well in volatile environments. Researchers, academics, and practitioners will be able to use the provided dataset and simulation tool to create and evaluate their own tactic volatility aware processes and contribute to this research space. CELIA's modifiable nature and ability to accept any utility equation allowed us to integrate the tactic volatility data. The output from the simulation tool allowed us to thoroughly test and evaluate TVA-E and the dataset.

Chapter 6

Related Works

Several works have demonstrated the importance of accounting for tactic volatility [51, 55]. A great majority of state-of-art decision-making processes don't adequately account for tactic volatility. These processes usually assume that tactics have static and unchanging attributes [61, 51]. This constraint has the ability to have a considerable negative impact on the system's efficacy, efficiency, and robustness [54, 55]. The need for processes such as our proposed TVA-E technique that account for tactic volatility has been discussed previously [54, 55]. Although Proactive Latency Aware (PLA)-based [54] techniques such as PLA-PMC [46], SB-PLA [51] and PLA-SDP [56] account for tactic volatility, existing PLA-based processes consider latency to be a static value [61, 51]. This work extends and advances existing proactive-focused processes by providing more accurate information that can be used to make higher-quality decisions. Another process, Model Predictive Control (MPC) [13] choose control inputs to optimize forecasts of process behavior [66, 51]. Our volatility-aware process can help techniques like MPC through a more informed and accurate control input.

Palmerino *et al.*[61] used ARIMA time series forecasting to account for tactic volatility in self-adaptive systems. The TVA-E method we propose improves on existing work in that: I) TVA-E utilizes eRNN, which has been demonstrated to be superior to ATIMA for predicting tactic volatility in our evaluations, II) This study performs a much more robust and detailed analysis and comparison of methods for predicting tactic volatility, and III) TVA-E also incorporates and assesses the use of uncertainty reduction tactics in decision-making processes.

Machine learning has previously been used to help determine the most efficient configurations for self-adaptive systems and to aid adaptation planning. Quin *et al.*[65] improved the MAPE-K [21] feedback loop by using a learning model. This model selects subsets of adaption options, allowing the system to make better decisions. Jamshidi *et al.*[40] used machine learning to identify Pareto-optimal configurations to avoid having to explore every possible configuration. This work also limited the search space to make planning more manageable. However, these works differ from ours in the sense that they do not address tactic volatility and its challenges.

Esfahani's *et al.*[31] work proposed a process that seeks to improve the self-adaptive process by using machine learning. This work incorporated online learning to assist in the decision-making process. Elkhodary [26] conducted a preliminary study to combine learning, dynamic optimization, and feature-orientation techniques to develop and propose a new class of self-adaptive systems that could update their adaptation logic on run-time. Our study differs in the sense that it is more proactive by employing learning to predict tactic volatility at run-time.

Kinneer *et al.*[43] devised a novel method for utilizing prior planning information to help self-adaptive systems adjust to new and unexpected scenarios. This study accounted for tactic failures and also argued for tactic latency and its pitfalls. Although this method is useful for overall system design, it lacks a mechanism to assist the system with learning and predicting tactic latency and cost value. Our TVA-E approach, on the other hand, is very good with learning and predicting tactic latency and cost values.

There have been several datasets utilized in prior self-adaptive systems research. The FIFA 98 dataset [10, 5] is a collection of requests made to the FIFA website over the course of four months. This dataset has been widely utilized in self-adaptive research for a number of purposes, including mimicking the need for more servers owing to increased network traffic [23, 20, 76, 18]. However, this dataset is not suitable for research for tactic volatility problems as it does not have the concept of latency or cost in it. Our study provides a dataset along with a simulation tool to create and evaluate tactic volatility aware processes.

Palmerino *et al.*[61] demonstrated the importance of accounting for tactic volatility and developed called *VolAtiLity EmulaTor* (VALET). Our proposed CELIA tool provides several improvements, variations, and advantages over VALET. Some of these advantages include easier and more detailed user customization and additional evaluation outputs (*e.g.*frequency of correct decisions, utility difference, expected utility not achieved, missed utility). The ongoing research in these areas demonstrates the need for a robust dataset and evaluation tool to support the research. CELIA also integrates the notion of performance penalties and propagation. This is where any wrong actions can have repercussions in the near future.

This research space has a few tools that have been used in the evaluation and analysis of self-adaptive decision-making processes. The *'ZNN.com'* [19] and *Rice University Bidding System* (RUBiS) [3] are multi-tiered web applications that have been used in the evaluation of self-adaptive processes as target systems. However, these tools are not simulation tools and require actual hardware and software to emulate web servers such as Apache HTTP. This makes use of these tools increasingly challenging and time-consuming. Moreover, they don't have a self-adaptive system component and require an additional layer of modifications to be used for research purposes [51]. DARTSim [50] is another tool that could be used as it provides a high-level simulation of a team of unmanned aerial vehicles (UAVs) conducting a reconnaissance mission in a hazardous and unknown environment. *Simulator for Web Infrastructure and Management* (SWIM) [58] simulates a web application such as Znn or RUBiS. Although DARTSim and SWIM have a concept of tactic latency, it is not volatile. CELIA differs from them as it is able to readily account for tactic volatility. CELIA is also easier to implement and can be easily customized to support a range of evaluations and configurations (datasets, utility equations, SLA).

Existing artifacts and model problems (exemplars) [4] for self-adaptive systems contain artifacts that act on real-world data and/or maybe accessible remotely for experimentation. However, no existing exemplars are known to have datasets with considerable degrees of tactic cost or latency volatility.

Chapter 7

Threats and Future Work

In many systems, the tactic cost may be a fuzzy and difficult-to-define metric. This inability to assess cost precisely could limit the quality and quantity of observed input values into our prediction process, limiting the adoption of our process. Furthermore, the cost might be a relative concept, and we consider it a quantitative value in our TVA-E approach. One could argue that the 'cost' of executing a tactic is the general wear and tear on a device's hardware. In most cases, such a cost is difficult to estimate. As a result, while utilizing TVA-E, the concept of cost must be confined to a value that is easily quantifiable.

We have showcased how our proposed TVA-E works both in a simulation tool and analyze its performance with statistical analysis. Despite demonstrating the benefits of our TVA-E approach with real-world experimental data, implementing processes in real-world scenarios and hardware can introduce its own sets of unforeseen unique challenges. Our adaptation approach will be integrated into physical equipment such as Internet of Things (IoT) devices, small unmanned aerial vehicles (UAVs) and drones, and self-adaptive online services in the future.

Although TVA-E allows for the monitoring of requirements indicated in a *Service Level Agreement* (SLA) for self-adaptive systems, time-series analysis does not always allow for the measurement of all needs. For example, a system might be required to be available for X percent of the time. This is not something that can be quantified or predicted using time-series analysis. Therefore, using this TVA-E process will require SLA requirements to be in a time-series format for this process to perform adequately.

Real-world uncertainty reduction tactics may involve trade-offs such as cost or risk,

which we did not take into account. However, we do not believe this is an issue because most uncertainty reduction techniques, including the one utilized in this study, are meant to be low-cost and low-risk by nature. As a result, we do not believe that incorporating cost or risk into our analysis would have significantly altered the conclusion of our findings. Furthermore, one of the goals of this project was to illustrate the benefits of uncertainty reduction tactics from a theoretical standpoint. In the evaluation of our work, we used the uncertainty reduction tactic of querying a decentralized resource. Further strategies for reducing uncertainty should be included in future studies.

This study has demonstrated TVA-E's ability to account for tactic volatility. However, there are limitations to this work. TVA-E may not be a significant improvement over current processes in the early stages of implementation. The performance of machine learning models depends heavily on the existence of data. As a result, due to a lack of prior data to consider, tactic attributes may still need to be predetermined. Furthermore, because of the need for historical data for machine learning models, observing tactic volatility might be impractical for any tactics that are sparsely deployed. This means that traditional predefined processes will still be necessary.

The goal of this project is to show the fundamental capabilities and benefits of uncertainty reduction techniques. This study did not have the goal of implementing and evaluating a variety of uncertainty reduction tactics because the primary goal was to demonstrate the foundational benefits of uncertainty reduction tactics. Moreover, uncertainty reduction tactics employed by a system will be domain-specific. However, further research is needed to determine the impact and potential benefits of additional uncertainty reduction strategies for a system.

Machine Learning algorithms, including EXAAM, require and perform offline training, testing, and validation. They typically require an extensive amount of data to generate models that can operate at an acceptable level. These machine learning algorithms also require an extensive amount of time on high-performance computing systems that result in excessive energy consumption. Algorithms such as LSTM and MLP also require a large

number of nodes in order to generate good working models. This can further slow down the training and run-times of these models. After the training, testing, and validation processes of neural network models are completed, the best model architectures are used, but they cannot be updated or improved further. However, EXAAM has recently been shown to be capable of allowing transfer learning to different architectural structures [28, 30]. EXAAM is also a lot more efficient once the model architectures are finalized, as it results in a lot fewer nodes compared to other algorithms such as LSTM and MLP. Future work with EXAAM and TVA-E will allow the online evolution of the algorithms. This is where the evolutionary process, learning, and optimization of the models can continue in an online fashion as the model is exposed to new information and learns from its actions.

The eRNNs trained by the EXAAM algorithm, that were used for the purpose of this research, was more prone to the effect of outliers compared to other algorithms. This was shown by its higher Mean Squared Error than ARIMA compared to the Mean Absolute Error. This needs to be taken into consideration while designing systems where data outliers can have high ramifications. It may be that in some scenarios, this characteristic of eRNN might not suit the case at hand. However, EXAAM is in its early stages of its development. As it sees more adoption, it will be fine-tuned to fit more use-cases. This will be very beneficial due to its efficient architecture. Moreover, future works with EXAAM, TVA-E and online learning should allow us to overcome this problem.

This work realizes that any uncertainty reduction operations utilized inside a system will be domain-specific; hence no work may presume that it will be able to implement and assess every type of uncertainty reduction operation. In our analysis, we factored in every fifth uncertainty reduction value. Changes to this sampling rate value could be made and evaluated in the future.

The data used in this study was generated by the CELIA tool we developed in conjunction with this study. The empirical results reported in this study should be considered in light of the limitations of the CELIA data generation tool. The tool gathers the data by downloading the same file from three different locations across the world. The tool was

run over a span of months. While the data gathering algorithms were thoroughly vetted as proof-of-concept and the data was cleaned according to industry standards, it may still be limited by its nature. Future studies can use independent hardware nodes spread across a network to capture higher volatility.

The evaluations performed in this study were done using statistical analysis on the results of the CELIA simulation tool we developed with this paper. This simulation tool was developed in light of the absence of other tools that could take into regard tactic volatility. Nonetheless, these results from this evaluation must be interpreted with caution. CELIA is a configurable simulation tool emulating a self-adaptive system that provides researchers with a platform to evaluate their own tactic volatility aware processes. It needs an external source of time-series data that it can iterate through to run the simulation. The quality of the evaluation is dependant on the data itself. Future studies can develop a tactic volatility aware solution such as DARTSim [50] that is a sandbox simulator that simulates the physical environment.

Chapter 8

Conclusion

We studied and proposed a new TVA-E process, which utilizes uncertainty reduction tactics. We also tested our TVA-E process with multiple learning methods including *ARIMA*, *LSTM*, *MLP* and *evolved Recurrent Neural Networks*. We came to a conclusion that eRNN is the most suitable for our TVA-E process because of its ability and general computing requirements. Our process in combination with eRNN (TVA-E) can easily integrate into mainstream adaptation processes, allowing it to have a positive impact on a wide range of self-adaptive systems. Our simulations and analysis, which used 52,106 records, show that:

1. TVA-E can account for tactic volatility adequately,
2. eRNN demonstrates its effectiveness compared with other leading machine learning alternatives. eRNN with URT is more than 65% more accurate than the next best model at predicting tactic cost in terms of $\overline{MSE}$. It is also 7% better than the next best model at predicting latency in terms of $\overline{MSE}$.
3. Uncertainty reduction tactics can be very useful, particularly in terms of making more reliable predictions regarding tactic volatility. eRNN takes advantage of the uncertainty information the best and improves its MSE by 50%

This study also contributes the CELIA data generation and simulation tool. CELIA can help researchers in a variety of self-adaptive research domains. The tool will help researchers and practitioners evaluate their tactic volatility aware processes as well as their

capacity to operate effectively and resiliently. This project makes all of the results, evaluation software, and other information available to the public at *https://github.com/valet-tool/valet-tool*.

Bibliography

[1] The internet traffic archive. http://ita.ee.lbl.gov/.

[2] Merriam-webster. (n.d.). utility. in merriam-webster.com dictionary. retrieved november 19, 2021. https://www.merriam-webster.com/dictionary/utility.

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

[4] Seams exemplars. https://www.hpi.uni-potsdam.de/giese/public/selfadapt/category/exemplar/, 2019.

[5] Worldcup98, n.d.

[6] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010.

[7] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Autonomic vertical elasticity of docker containers with elasticdocker. In *2017 IEEE 10th international conference on cloud computing (CLOUD)*, pages 472–479. IEEE, 2017.

[8] Cesare Alippi, Giuseppe Anastasi, Mario Di Francesco, and Manuel Roveri. Energy management in wireless sensor networks with energy-hungry sensors. *IEEE Instrumentation & Measurement Magazine*, 12(2):16–23, 2009.

[9] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. Modeling and analyzing mape-k feedback loops for self-adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23. IEEE, 2015.

[10] Martin Arlitt and Tai Jin. A workload characterization study of the 1998 world cup web site. *IEEE network*, 14(3):30–37, 2000.

[11] Peter E Bailey, David K Lowenthal, Vignesh Ravi, Barry Rountree, Martin Schulz, and Bronis R De Supinski. Adaptive configuration selection for power-constrained heterogeneous systems. In *2014 43rd International Conference on Parallel Processing*, pages 371–380. IEEE, 2014.

[12] Luiz André Barroso and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108, 2009.

[13] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[14] Javier Cámara, Gabriel A Moreno, and David 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.

[15] Javier Cámara, Gabriel A Moreno, David Garlan, and Bradley Schmerl. Analyzing latency-aware self-adaptation using stochastic games and simulations. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 10(4):23, 2016.

[16] Javier Cámara, Wenxin Peng, David Garlan, and Bradley 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.

[17] Paulo Casanova, David Garlan, Bradley Schmerl, and Rui Abreu. Diagnosing unobserved components in self-adaptive systems. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 75–84, 2014.

[18] T. Chen and R. Bahsoon. Self-adaptive and online qos modeling for cloud-based software services. *IEEE Transactions on Software Engineering*, 43(05):453–475, may 2017.

[19] Shang-Wen Cheng, David Garlan, and Bradley Schmerl. Evaluating the effectiveness of the rainbow self-adaptive system. In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '09, page 132–141, USA, 2009. IEEE Computer Society.

[20] Salva Daneshgadeh, Thomas Kemmerich, Tarem Ahmed, and Nazife Baykal. An empirical investigation of ddos and flash event detection using shannon entropy, koad and svm combined. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 658–662. IEEE, 2019.

[21] Rogério De Lemos, Holger Giese, Hausi A Müller, and Mary Shaw. Software engineering for self-adaptive systems ii. 2010.

[22] Travis Desell. Accelerating the evolution of convolutional neural networks with node-level mutations and epigenetic weight initialization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 157–158. ACM, 2018.

[23] A Dhanapal and P Nithyanandam. An effective mechanism to regenerate http flooding ddos attack using real time data set. In *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 570–575. IEEE, 2017.

[24] Shuyang Du, Manish Pandey, and Cuiqun Xing. Modeling approaches for time series forecasting and anomaly detection. 2017.

[25] Ibrahim Elgendi, Md Farhad Hossain, Abbas Jamalipour, and Kumudu S Munasinghe. Protecting cyber physical systems using a learned mape-k model. *IEEE Access*, 7:90954–90963, 2019.

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

[27] AbdElRahman ElSaid, Steven Benson, Shuchita Patwardhan, David Stadem, and Desell Travis. Evolving recurrent neural networks for time series data prediction of coal plant parameters. In *The 22nd International Conference on the Applications of Evolutionary Computation*, Leipzig, Germany, April 2019.

[28] AbdElRahman ElSaid, Joshua Karnas, Zimeng Lyu, Daniel Krutz, Alexander G Ororbia, and Travis Desell. Neuro-evolutionary transfer learning through structural adaptation. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 610–625. Springer, 2020.

[29] AbdElRahman ElSaid, Joshua Karns, Alexander Ororbia II, Daniel Krutz, Zimeng Lyu, and Travis Desell. Neuroevolutionary transfer learning of deep recurrent neural networks through network-aware adaptation, 2020.

[30] AbdElRahman ElSaid, Joshua Karns, Zimeng Lyu, Daniel Krutz, Alexander Ororbia, and Travis Desell. Improving neuroevolutionary transfer learning of deep recurrent neural networks through network-aware adaptation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 315–323, New York, NY, USA, 2020. Association for Computing Machinery. Best paper nominee.

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

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

[33] Aidin Ferdowsi, Samad Ali, Walid Saad, and Narayan B Mandayam. Cyber-physical security and safety of autonomous connected vehicles: Optimal control meets multi-armed bandit learning. *IEEE Transactions on Communications*, 67(10):7228–7244, 2019.

[34] Erik 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.

[35] Mohamed Medhat Gaber and Philip S Yu. A framework for resource-aware knowledge discovery in data streams: a holistic approach with its application to clustering. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 649–656, 2006.

[36] David Garlan, S-W Cheng, A-C Huang, Bradley Schmerl, and Peter Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54, 2004.

[37] David Garlan, Bradley Schmerl, and Shang-Wen Cheng. *Software Architecture-Based Self-Adaptation*, pages 31–55. Springer US, Boston, MA, 2009.

[38] Thomas J. Glazier, Bradley Schmerl, Javier Cámara, and David Garlan. Utility theory for self-adaptive systems. Technical Report CMU-ISR-17-119, Carnegie Mellon University Institute for Software Research, December 2017.

[39] Marcus Paul Gutierrez and Christopher Kiekintveld. Bandits for cybersecurity: Adaptive intrusion detection using honeypots. In *AAAI Workshop: Artificial Intelligence for Cyber Security*, 2016.

[40] Pooyan Jamshidi, Javier Cámara, Bradley Schmerl, Christian Kästner, and David Garlan. Machine learning meets quantitative planning: Enabling self-adaptation in autonomous robots. In *Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '19, pages 39–50, Piscataway, NJ, USA, 2019. IEEE Press.

[41] Gueyoung Jung, Kaustubh R. Joshi, Matti A. Hiltunen, Richard D. Schlichting, and Calton Pu. A cost-sensitive adaptation engine for server consolidation of multitier applications. Middleware'09, page 163–183, Berlin, Heidelberg, 2009. Springer-Verlag.

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

[43] Cody Kinneer, David Garlan, and Claire Le Goues. Information reuse and stochastic search: Managing uncertainty in self-* systems. 2019.

[44] Matt Knudson and Kagan Tumer. Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems*, 59(6):410–420, 2011.

[45] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In *Future of Software Engineering (FOSE '07)*, pages 259–268, May 2007.

[46] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with prism: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.

[47] Kunhui Lin, Qiang Lin, Changle Zhou, and Junfeng Yao. Time series prediction based on linear regression and svr. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 1, pages 688–691. IEEE, 2007.

[48] Wenhong Ma, Changcheng Huang, and James Yan. Adaptive sampling for network performance measurement under voice traffic. In *2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577)*, volume 2, pages 1129–1134. IEEE, 2004.

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

[50] Gabriel Moreno, Cody Kinneer, Ashutosh Pandey, and David Garlan. Dartsim: an exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 181–187. IEEE, 2019.

[51] Gabriel A Moreno. *Adaptation Timing in Self-Adaptive Systems*. PhD thesis, Carnegie Mellon University, 2017.

[52] Gabriel A. Moreno. Adaptation timing in self-adaptive systems, Jun 2018.

[53] Gabriel A. Moreno, Javier Cámara, David Garlan, and Mark Klein. Uncertainty reduction in self-adaptive systems. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '18, page 51–57, New York, NY, USA, 2018. Association for Computing Machinery.

[54] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley 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.

[55] Gabriel A Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. Efficient decision-making under uncertainty for proactive self-adaptation. In *Autonomic Computing (ICAC), 2016 IEEE International Conference on*, pages 147–156. IEEE, 2016.

[56] Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. Flexible and efficient decision-making for proactive latency-aware self-adaptation. *ACM Trans. Auton. Adapt. Syst.*, 13(1):3:1–3:36, April 2018.

[57] Gabriel A. Moreno, Alessandro V. Papadopoulos, Konstantinos Angelopoulos, Javier Cámara, and Bradley 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.

[58] Gabriel A. Moreno, Bradley Schmerl, and David Garlan. Swim: An exemplar for evaluation and comparison of self-adaptation approaches for web applications. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '18, pages 137–143, New York, NY, USA, 2018. ACM.

[59] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*, pages 999–1004. Springer, 1997.

[60] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. Investigating recurrent neural network memory structures using neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 446–455, New York, NY, USA, 2019. ACM.

[61] J. Palmerino, Q. Yu, T. Desell, and D. Krutz. Improving the decision-making process of self-adaptive systems by accounting for tactic volatility. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 949–961, Nov 2019.

[62] Jeffrey Palmerino, Qi Yu, Travis Desell, and Daniel E. Krutz. Improving the decision-making process of self-adaptive systems by accounting for tactic volatility, 2020.

[63] José D'Abruzzo Pereira, Rui Silva, Nuno Antunes, Jorge LM Silva, Breno de França, Regina Moraes, and Marco Vieira. A platform to enable self-adaptive cloud applications using trustworthiness properties. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 71–77, 2020.

[64] Gábor Petneházi. Recurrent neural networks for time series forecasting, 2019.

[65] Federico Quin, Danny Weyns, Thomas Bamelis, Sarpreet Singh Buttar, and Sam Michiels. Efficient analysis of large adaptation spaces in self-adaptive systems using machine learning. In *Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '19, pages 1–12, Piscataway, NJ, USA, 2019. IEEE Press.

[66] James B Rawlings. Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52, 2000.

[67] Kandala Saikiran and MSV Sashi Kumar. An adaptive authorization in openstack cloud platform using mape-k. 2019.

[68] Dale E. Seborg, Duncan A. Mellichamp, and Thomas F. Edgar. *Process Dynamics and Control*. Wylie Series in Chemical Engineering. John Wiley & Sons, third edition, 2011.

[69] Matthew David Smith and Marie-Elisabeth Paté-Cornell. Cyber risk analysis for a smart grid: How smart is smart enough? a multi-armed bandit approach. In *SG-CRC*, pages 37–56, 2017.

[70] Kenneth Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[71] Fei Sun and Tong Zhang. Low-power state-parallel relaxed adaptive viterbi decoder. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(5):1060–1068, 2007.

[72] Niels LM Van Adrichem, Christian Doerr, and Fernando A Kuipers. Opennetmon: Network monitoring in openflow software-defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8. IEEE, 2014.

[73] Jeroen Van Der Donckt, Danny Weyns, Federico Quin, Jonas Van Der Donckt, and Sam Michiels. Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals.

[74] Tao Wang, Jiwei Xu, Wenbo Zhang, Zeyu Gu, and Hua Zhong. Self-adaptive cloud monitoring with online anomaly detection. *Future Generation Computer Systems*, 80:89–101, 2018.

[75] David L Wells and Paul Pazandak. Taming cyber incognito: Tools for surveying dynamic/reconfigurable software landscapes. In *Working Conference on Complex and Dynamic Systems Architectures, Brisbane, Australia*, pages 13–24, 2001.

[76] Zhao Zhibin, Yao Lan, Yang Xiaochun, Li Binyang, and Yu Ge. A filter-based uniform algorithm for optimizing top-k query in distributed networks. *Wuhan University Journal of Natural Sciences*, 11(5):1383–1388, 2006.