A Framework for Managing Uncertainty in Self-Adaptive Software Systems

Naeem Esfahani
Department of Computer Science
George Mason University
Fairfax, Virginia, USA
nesfaha2@gmu.edu

***Abstract*—Self-adaptation endows a software system with the ability to satisfy certain objectives by automatically modifying its behavior. While many promising approaches for the construction of self-adaptive software systems have been developed, the majority of them ignore the uncertainty underlying the adaptation decisions. This has been one of the key inhibitors to widespread adoption of self-adaption techniques in risk-averse real-world settings. In this research abstract I outline my ongoing effort in the development of a framework for managing uncertainty in self-adaptation. This framework employs state-of-the-art mathematical approaches to model and assess uncertainty in adaptation decisions. Preliminary results show that knowledge about uncertainty allows self-adaptive software systems to make better decisions.**

Keywords- uncertainty; self-adaptation; software architecture

I. INTRODUCTION

Self-adaptation is an effective approach in dealing with the changing dynamics of many application domains, such as mobile and pervasive systems. In response to changes in the environment or requirements, a self-adaptive software system modifies itself to satisfy certain objectives [1], [2]. While the benefits of such systems are plenty, their development has shown to be more challenging than traditional software systems [1]. One key culprit is that self-adaptation is subject to *uncertainty* [1].

Uncertainty can be observed in every facet of adaptation, albeit at varying degrees. It follows from the fact that the system's user, adaptation logic, and business logic are loosely coupled, introducing numerous sources of uncertainty [3]. Consider that users often find it difficult to accurately express their quality preferences using complex utility functions, sensors employed for monitoring often have uncontrollable noise, analytical models used for assessing the system's quality attributes by definition make simplifying assumptions that may not hold at runtime, and so on. All of these factors challenge the confidence with which the adaptation decisions are made. A key observation is that while the level of uncertainty could vary, no self-adaptive software system is ever completely free of it.

This is precisely the challenge I have aimed to address in my PhD research. I highlight the overall intuitions behind my research as follows:

- Uncertainty in self-adaptive software systems is due to complexity and loose coupling between the components of such systems.
- Models need to make simplifying assumptions to allow abstraction. Therefore, they only provide an approximation of system's behavior.
- Analytical models use the information from the past to predict the behavior of the system in the future. Predictions are often prone to uncertainty.
- A range, which depicts the uncertainty, is a more representative approach for presenting the behavior of the system compared to a point estimate.

The research community has made great strides in tackling the complexity of constructing self-adaptive software systems [1], [2]. However, as corroborated by others [1], there is a dearth of applicable techniques for handling uncertainty in this setting. A few researchers have recently begun to address uncertainty issues in requirements specification [4], [5] and resource prediction [6], but no approach that I am aware of has tackled the challenge posed by uncertainty in making adaptation decisions. I believe this has been one of the primary obstacles to widespread adoption of self-adaptation in risk-averse domains.

This research abstract is organized as follows. Section II motivates the problem using a robotic software system. Section III outlines the hypotheses of this research. Section IV provides an insight into the approach. Section V outlines the evaluation framework. Section VI describes the current status of the research. The research abstract concludes with a discussion of remaining work and related work.

II. MOTIVATION

Similar to my recent publication [7], I use a robotic software system to motivate and describe this research. The robotic software is part of a distributed search and rescue system aimed at supporting the government agencies in dealing with emergency crises (e.g., fire, hurricane). Figure 1b provides an abridged view of the robotic system's architecture. The software components comprising the robotic system range from abstractions of the physical entities, such as software

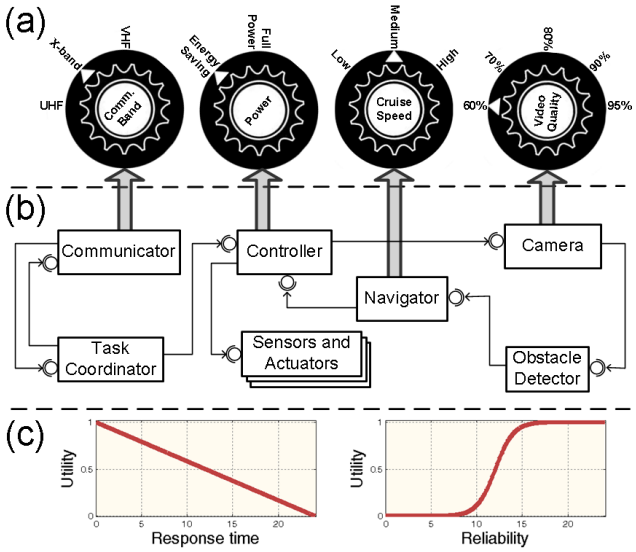

Figure 1. A subset of the robotic software: (a) configuration dimensions and alternatives for components of the robot, (b) software architecture, and (c) utility functions defined in terms of quality attributes.

controlled sensors and actuators on board the robot, to purely logical functionalities, such as image detection and navigation.

The software components comprising this system are *customizable,* meaning that they can be configured to operate in different modes of operation. Figure 1a shows some of the available configuration dimensions. For instance, *Power* is a configuration dimension for the *Controller* component. A *Controller* could operate in either *Energy Saving* or *Full Power* mode. A component may have many configuration dimensions.

The configuration of a software component determines its quality attributes (e.g., response time) and resource usage (e.g., memory), which could also impact the properties of the entire system. For instance, given the resource-constrained nature of the mobile robots, the configuration decisions of each component have a significant impact on the system's performance as well as its battery life. Such decisions can only be effectively made at runtime, since the system properties (e.g., available bandwidth) are often not known at design-time and may change at runtime.

As shown in Figure 1c, for making runtime decisions, utility functions capturing the user's satisfaction with different levels of quality attribute (e.g., availability) are used. The adaptation logic uses analytical models to estimate the effect of configuration decision on the system's quality attributes, and in turn the resulting utility. For example, given the configuration of the robot's components, an analytical model, such as Queueing Network model [8], may be used to quantify the response time of a particular scenario. The objective is to find a configuration that achieves the maximum overall utility.

The above approach is rather myopic, since it does not consider the uncertainty of information used in making adaptation decisions. Consider that almost every facet of the approach outlined above faces some form of uncertainty:

- *Uncertainty in System Parameters*: The monitoring data obtained from a running system rarely corresponds to a single value, but rather a distribution of values obtained over the observation period. For instance, a sensor monitoring the available network bandwidth may return a slightly different number every time a sample is collected. This variation could be either due to actual changes in the bandwidth or the error (noise) in the employed probes.
- *Uncertainty in Analytical Models*: Analytical models often make simplifying assumptions, and thus provide only estimates of the system's behavior. For instance, an analytical model quantifying the system's response time may account for the dominant factors, such as execution time of components, and ignore others, such as the transmission delay difference between TCP and UDP. Response time estimates provisioned by such a formulation are not only error-prone, but also the magnitude of error varies depending on the circumstances.
- *Uncertainty in User Preferences*: Eliciting user's preferences in terms of utility functions, such as those depicted in Figure 1c, is a well-known challenge [1]. Often users have difficulty expressing their preferences and thus the overall accuracy of the utility functions remains subjective, making the analysis based on them prone to uncertainty.

The uncertainty in these elements challenges the system's ability in making decisions that bring about the intended effects.

III. Research Hypotheses

My PhD research can be described based on the following hypotheses:

Hypothesis 1: The uncertainty in the knowledge and information used to make adaptation decisions is inevitable. There are several mathematical approaches for dealing with uncertainty by considering a range of behavior instead of point estimates. These approaches have been successfully used in other fields (e.g., control theory, economy, and statistics). I hypothesize that *if the adaptation decisions are made based on ranges of uncertainty instead of point estimates, the trade-offs will be more accurate and as a result the system will be more effective in achieving its objectives*.

Hypothesis 2: Uncertainty in making adaptation decisions is neglected partially due to the misconception that dealing with uncertainty is computationally extensive. However, there are modern mathematical approximation techniques that can deal with uncertainty efficiently. I hypothesize that *it is possible to incorporate uncertainty in the decision-making process in an efficient and timely manner for execution at run-time*.

Hypothesis 3: Precisely quantifying uncertainty is a difficult task. For example, it is not very easy to exactly quantify the uncertainty of network bandwidth in the robotic system. I hypothesize that *incorporating partial information about the uncertainty in the analysis allows for better*

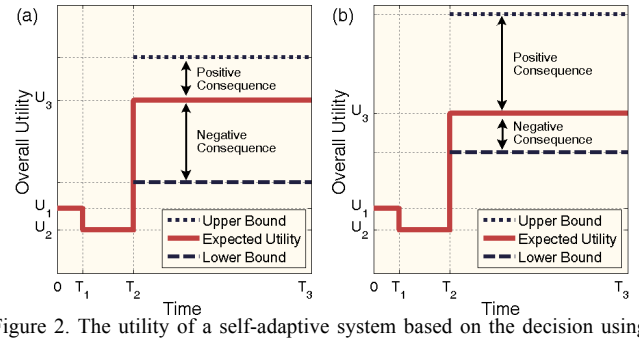

Figure 2. The utility of a self-adaptive system based on the decision using: (a) traditional approach, where the uncertainty is not considered and (b) my approach, which considers uncertainty.

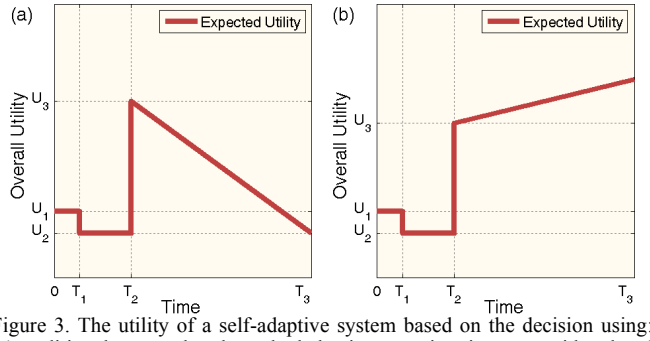

Figure 3. The utility of a self-adaptive system based on the decision using: (a) traditional approach, where the behavior over time is not considered and (b) my approach, which considers the behavior over time.

decisions compared to having no information at all (i.e., considering only point estimates).

Hypothesis 4: The uncertainty in the environment is not a fixed value and may change over time. There are mathematical approaches, such as time series analysis, for anticipating trajectories for values of interest over time. I hypothesize that *by considering the impact of adaptation decisions over time, the quality and stability of the software systems can be improved.*

IV. Approach

Figure 2a shows the typical behavior of a self-adaptive system that does not incorporate uncertainty in its analysis. We abstractly refer to this as the *traditional approach*. The system is initially executing with utility U_1 prior to time T_1. At time T_1, due to either an internal or external change, the system's utility drops to U_2. By time T_2, the self-adaptation logic detects this drop in utility, finds and effects an optimal configuration, which is conventionally defined as the one achieving the maximum utility. As shown in Figure 2a, this corresponds to U_3, which represents the *expected* utility of the best configuration for the system. In practice, however, the *actual* utility of the system may vary between the two dashed lines, representing the likely positive and negative consequences of uncertainty during T_3. By not accounting for uncertainty in the analysis, the approach is vulnerable to gross over- or under-estimation of the utility.

The centerpiece of my research, which is detailed in [7], is the reconceptualization of what is traditionally considered as the optimal solution, such that the uncertainty is incorporated into the analysis. I illustrate the insights using Figure 2b. Similar to the scenario of Figure 2a, a new configuration is effected at time T_2, except the strategy is to select the configuration that concurrently satisfies the following three objectives: (1) maximizes U_3, which represents the most likely utility for the system under uncertainty; (2) maximizes the *positive consequence* of uncertainty, which represents the likelihood of the solution being better than U_3; and (3) minimizes the *negative consequence* of uncertainty, which represents the likelihood of the solution being worse than U_3.

As depicted in Figure 2, concurrent satisfaction of the three objectives may result in a smaller value of *expected* utility (i.e., U_3) compared to that of the traditional approach. But since the information used to estimate the expected utility is uncertain, *expected* utility is not guaranteed to occur in practice. I argue that the range of possible utility determines the true quality of a solution.

As depicted in Figure 3, the other aspect of my research is the anticipation of the behavior of the selected solution over time. Figure 3a depicts a configuration picked by traditional approach in which the behavior over time is neglected. On the other hand, my approach, which is depicted in Figure 3b, considers the behavior of the selected configuration over time, i.e., selects a configuration that has lower utility at the moment in which the decision is made with expectation for improvement in the future. Since the system is expected to do better in the future, my approach decreases the number of adaptations compared to traditional approach. Note that in Figure 3 the behavior over time is depicted linearly, but in general the behavior over time may have different shapes.

V. Evaluation Framework

For the experiments I have set up a controlled environment to allow me to create and measure the effect of uncertainty in the system. For that purpose, I used XTEAM [9], an architectural modeling, analysis, and simulation environment

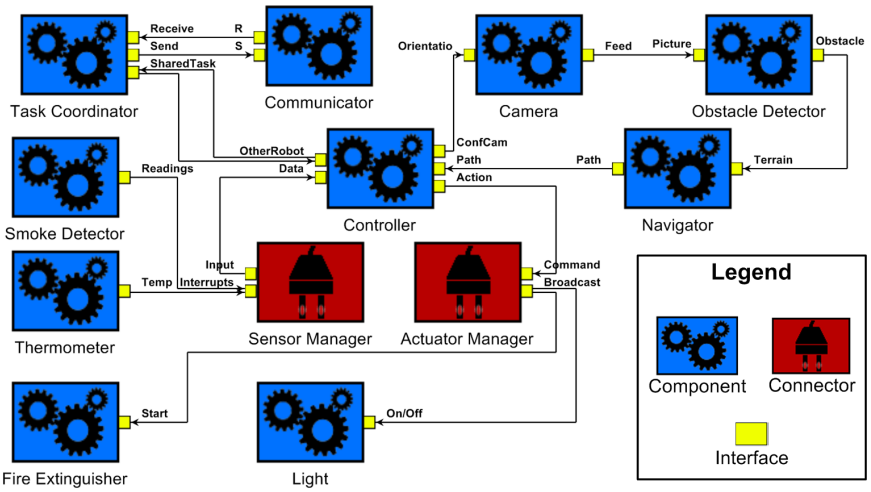

Figure 4. The high-level XTEAM model of the robotic software system.

that is integrated with Prism-MW [10], which is a middleware platform with extensive support for runtime monitoring and adaptation.. Through this integration, the XTEAM models are kept in sync with the software running on top of Prism-MW, and vice versa. Moreover, XTEAM can also be used to control the execution of the software running on Prism-MW, including the ability to fix the workload, and configure the software and hardware properties. I use XTEAM to simulate uncertainty by controlling the extent of random changes in the system parameters (e.g., available network bandwidth, and memory consumption of configuration alternatives). However, neither the system under study nor my algorithms will be controlled by XTEAM, which will allow them to behave as they would in practice. For the purpose of this research I will mainly use the robotic software to evaluate my ideas and contributions. Figure 4 depicts the high-level XTEAM model of the robotic software system.

VI. CURRENT STATUS

I have developed a general quantitative approach for tackling the complexity of automatically making adaptation decisions under uncertainty [7]. In this part of framework estimates of uncertainty in the elements comprising a self-adaptation problem are incorporated in *possibilistic analysis* of the adaptation choices. Possibilistic analysis is founded on the principles of fuzzy mathematics [11], which provides a sound basis for representing uncertainty, as well as dealing with its negative and positive consequences on the adaptation choices. As I mentioned before, my framework redefines the conventional definition of optimal adaptation decision to one that has the best range of behavior. In turn, the selected solution has the highest likelihood of satisfying the system's quality objectives, even if due to uncertainty, properties expected of the system are not borne out in practice.

I demonstrated my framework by applying it to the problem of improving the robotic software system's quality of service via runtime reconfiguration of its customizable software components. I evaluated the developed subset of my framework under numerous circumstances and using a prototype of a robotic software system described in Section II. The results demonstrated my framework's ability to deal with uncertainty by making adaptation decisions that are superior to those of the conventional approach. For instance, Figure 5 shows the preliminary results of comparing the quality of solutions selected by my framework with the traditional approach in 10 different experiments. For each experiment, I applied both approaches on the same adaptation problem. I then executed the software system in the selected configuration, and observed the actual quality of the solution. For a fair comparison, in each experiment, I used XTEAM to fix the application workload, as well as the range of uncertainty in the execution context (e.g., network bandwidth, memory usage). "Fixing the range of uncertainty" means controlling the range of random behavior within each source of uncertainty. Thus, different executions still resulted in different observed behaviors. The results show that in comparison to the traditional approach, my framework is more likely to select a solution with better overall utility.

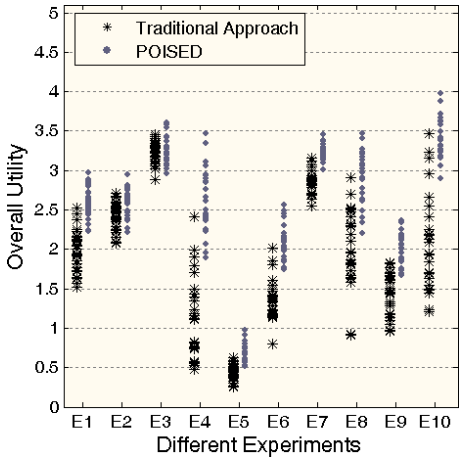

Figure 5. Comparison of my framework with traditional approach in 10 different experiments where 30 observations are collected for each experiment.

The experiments in Figure 5 follow the description of Figure 2: The worst-case quality of configurations selected by my framework is comparable with the best-case quality of configurations selected by traditional approach. In other words, the overall range of behavior in my framework is always better. This is expected, since traditional approach aims to maximize the mean behavior of the system, while my approach aims to maximize the range of behavior.

VII. REMAINING WORK

So far, my focus has been on uncertainty in monitoring and user preferences, which are two examples of internal uncertainties. I distinguish between the external and internal uncertainty. *External uncertainty* arises from the environment or domain in which the software is deployed. For example, external uncertainty for a software system deployed in an unmanned vehicle may include the likelihood of certain weather conditions occurring. Software self-adaptation is one approach in dealing with the effects of external uncertainty, e.g., in a snowstorm the vehicle's navigator component may be replaced with a more conservative navigator to avoid a collision. On the other hand, *internal uncertainty* is rooted in the difficulty of determining the impact of adaptation on the system's quality objectives, e.g., determining the impact of replacing a software component on the system's responsiveness, battery usage, etc.

Another venue of future work is investigation of the applicability of my research ideas to external uncertainty. In fact, learning for self-adaption, which was studied in FUSION framework [12], is a way of mitigating the external uncertainty.

My framework does not currently consider the behavior over time in decision-making. I am currently studying different approaches (e.g., time series analysis) for incorporating such effects. I will be able to benefit from the literature in control theory for this aspect of my research. Moreover, I am investigating different approaches for merging the two aspects of my research together. In other words, I am studying the interactions of anticipation (which as depicted in Figure 3 captures the uncertainty over time) with ranges (which as depicted in Figure 2 capture the uncertainty in a given time).

Finally, up to this point, I have used fuzzy logic for decision-making in the face of uncertainty because of its efficiency (recall hypothesis 3). I plan to study other efficient approaches for incorporating uncertainty in analysis, such as probability theory and Bayesian networks.

VIII. Related Work

The literature in the area of self-adaption is extensive. In lieu of enumerating all of the related studies, I refer the reader to [1], [2] for a comprehensive analysis of the state-of-the-art in self-adaptation. I focus my discussion here to those works that are of utmost relevance. The challenge posed by uncertainty in the construction of dependable self-adaptive software system is an established concept [1], [13]. A few works [3-6], [12] have aimed to tackle the different facets of this challenge as follows.

Whittle et al. [5] introduced RELAX, a formal requirements specification language that relies on Fuzzy Branching Temporal Logic to specify the uncertain requirements in self-adaptive systems. In a subsequent publication [4], Cheng et al. extended RELAX with goal modeling to specify the uncertainty in the objectives. This research is complementary to their work, as RELAX targets requirements specification phase, while my framework targets decision-making phase at runtime.

Dynamic configuration of resource-aware services was studied by Poladian et al. [18], where they showed how to select an appropriate set of services to carry out a user task, and allocate resources among those services at runtime. Subsequently, the work was extended to make anticipatory decisions [6], and considered the inaccuracy of future resource usage predictions. Unlike this research, their approach neither aims to satisfy a utility range, nor employs analysis techniques to incorporate the effect of uncertainty in decisions.

Cheng and Garlan [3] described three specific sources of uncertainty (*problem-state identification*, *strategy selection*, and *strategy outcome*) in self-adaptation and provided high-level guidelines for mitigating them in Rainbow [14]. In this research, I am proposing a novel approach for tackling the challenge of *strategy outcome*, i.e., the impact of uncertainty on the selected solution, and techniques to deal with it.

Finally, in a recent work from our group, we presented FUSION [12], a learning based approach to engineering self-adaptive systems. Instead of relying on static analytical models that are subject to wrong assumptions, FUSION uses machine learning to self-tune the adaptive behavior of the system to unanticipated changes, but does not address making adaptation decisions under uncertainty.

IX. Conclusion

In this research abstract I have described the overview of my PhD research. I also presented a subset of my framework that has been the latest outcome of my research. This subset uses possibility theory and fuzzy logic to deal with uncertainty in monitoring and user preferences. This research is being expanded in three directions: (1) I am studying different approaches for dealing with uncertainty efficiently; (2) I am studying possible extensions to the approach so I can include other kinds of uncertainties (both internal and external) in my analysis; (3) I am studying possible techniques for anticipating utility over time and their integration with the depiction of behavior as a range.

Acknowledgement

I would like to thank my advisor, Dr. Sam Malek, for his support and advice during this research. This work is partially supported by grant CCF-0820060 from the NSF.

References

[1] B. Cheng et al., "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems, LNCS Hot Topics*, 2009, pp. 1-26.

[2] J. Kramer and J. Magee, "Self-Managed Systems: an Architectural Challenge," in *Int'l Conf. on Software Engineering*, Minneapolis, Minnesota, 2007, pp. 259-268.

[3] S. W. Cheng and D. Garlan, "Handling uncertainty in autonomic systems," in *Int'l Wrkshp. on Living with Uncertainty*, Atlanta, Georgia, 2007.

[4] B. H. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty," in *Int'l Conf. on Model Driven Engineering Languages and Systems*, Denver, Colorado, 2009.

[5] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J.-M. Bruel, "RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems," in *Int'l Requirements Engineering Conf.*, Atlanta, Georgia, 2009, pp. 79-88.

[6] V. Poladian, D. Garlan, M. Shaw, M. Satyanarayanan, B. Schmerl, and J. Sousa, "Leveraging Resource Prediction for Anticipatory Dynamic Configuration," in *Int'l Conf. on Self-Adaptive and Self-Organizing Systems*, Boston, Massachusetts, 2007, pp. 214-223.

[7] N. Esfahani, E. Kouroshfar, and S. Malek, "Taming Uncertainty in Self-Adaptive Software," in *The joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Szeged, Hungary, 2011.

[8] D. A. Menasce, L. W. Dowdy, and V. A. F. Almeida, *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, 2004.

[9] G. Edwards, S. Malek, and N. Medvidovic, "Scenario-Driven Dynamic Analysis of Distributed Architectures," in *Int'l Conf. on Fundamental Approaches to Software Engineering*, Braga, Portugal, 2007, vol. 4422, pp. 125-139.

[10] S. Malek, M. Mikic-Rakic, and N. Medvidovic, "A Style-Aware Architectural Middleware for Resource-Constrained, Distributed Systems," *IEEE Trans. Softw. Eng.*, vol. 31, no. 3, pp. 256-272, 2005.

[11] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.

[12] A. Elkhodary, N. Esfahani, and S. Malek, "FUSION: A Framework for Engineering Self-Tuning Self-Adaptive Software Systems.," in *Int'l Symp. on the Foundations of Software Engineering*, Santa Fe, New Mexico, 2010, pp. 7-16.

[13] D. Garlan, "Software Engineering in an Uncertain World," in *FSE/SDP Wrkshp. on the Future of Software Engineering Research*, Santa Fe, New Mexico, 2010.

[14] D. Garlan, S. W. Cheng, A. C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," *IEEE Computer*, vol. 37, no. 10, pp. 46-54, Oct. 2004.