Bio-Inspired Techniques to Support High-Assurance Dynamically Adaptive Systems that Explicitly Handle Environmental Uncertainty

Betty H.C. Cheng and Philip K. McKinley
with A. J. Ramirez, A.C. Jensen, C. Byers, D. Knoester

February 2011

Workshop areas: Open Experimental/Challenge problems, Automotive Secure High Confidence Platforms, Safety-Critical Design Process

1 Overview

A dynamically adaptive system (DAS) must monitor itself and its execution environment to detect conditions warranting a reconfiguration. At run time, a DAS analyzes this monitoring information to determine when and how to *safely* reconfigure itself such that it continuously satisfies functional and non-functional requirements. Unfortunately, unanticipated environmental conditions may negatively impact the accuracy and reliability of monitoring information, and thus compromise the decision-making capabilities of a DAS. For example, unexpected sensory inputs, such as those produced by sensor noise or failure, may alter a DAS's self-assessment of requirements satisfaction. Moreover, it is impossible for a human to know and/or enumerate all possible combinations of system and environmental conditions that a DAS may encounter [1]. As a result, it is important to explore how system and environmental conditions impact the behavior of a DAS, preferably during the early stages of requirements engineering where there is greater flexibility for resolving obstacles that prevent the satisfaction of specific goals.

Over the past ten years, the authors have developed a number of techniques to address multiple dimensions of the challenges relating to the development of DAS that must operate safely even in the face of environmental uncertainty. Given the complexity and the large number of possible solutions, we have explored three key directions of research. First, we have developed a number of different techniques [2, 3, 4] to support the assurance of DASs that are designed to ensure that safe, acceptable behavior is delivered before, during, and after adaptation. Second we have developed a new requirements-specification language, RELAX, intended to explicitly address the need to support dynamic adaptation to handle functional and non-functional tradeoffs while responding to environmental conditions [1, 5]. Finally, we have developed a number of techniques to support run-time adaptation/reconfiguration with inspiration from biology that has clearly demonstrated robustness, self-healing, and adaptability in response to unexpected and adverse environmental conditions. Due to space constraints, we only describe work in the bio-inspired area as we believe that that is one of the most promising directions for tackling the challenges posed by DAS, and we have also been able to incorporate the results from the other two lines of research.

2 Bio-Inspired Techniques.

We have developed a number of techniques that harness evolutionary computation to address the challenges with developing a DAS that is able to support run-time monitoring and adaptation to handle environmental uncertainty. Our techniques include those that support adaptive run-time monitoring of system requirements, run-time generation of reconfigurations at the system architectural level, run-time generation of adaptive logic between configurations, and automatic generation of suites of design models for different configurations that satisfy safety and functional requirements [6, 7, 8], where each configuration makes different non-functional tradeoffs, such as performance, energy consumption, and security.

Adaptive Monitoring. Self-reconfiguration enables a dynamically adaptive system (DAS) to continuously satisfy its requirements even as system and environmental conditions change [9]. To verify that a DAS satisfies its requirements at run time, it must monitor itself and its execution environment by continuously probing components and sensors to obtain data. This monitoring data can then be analyzed to determine if a requirement has been violated or to detect conditions conducive to a requirement violation. Monitoring, however, is often computationally expensive, intrusive, and difficult to design. In addition, continuous monitoring involves tradeoffs between monitoring costs and accuracy, or the coverage and coherence of gathered data. For example, excessive monitoring may improve monitoring accuracy, yet it may also interfere with and alter the behavior of a DAS in unpredictable ways. Similarly, insufficient monitoring may conserve system resources, yet it may also fail to detect events leading to a requirement violation. As such, it is desirable to adapt the monitoring behavior of a DAS at run time in response to changing system and environmental conditions. Plato-RE [10] is an evolutionary computation-based approach that combines the concepts of requirements monitoring and adaptive sampling to enable a DAS to detect possible requirements violations while minimizing monitoring costs.

Run-time Generation of Reconfigurations. Plato-MDE [11, 12, 13] is an evolutionary computation-based approach for generating target system models at *run-time* in response to changing requirements and environmental conditions. Each target system model represents a potential system configuration that may be reached through a sequence of reconfiguration steps. Plato-MDE evaluates each generated target system model to determine its suitability given current system conditions. In addition, Plato-MDE leverages current system models to constrain the degree of change in the generated target models. As a result, Plato-MDE enables an adaptive system to implicitly control the complexity and novelty of the reconfiguration itself at run time. Moreover, rather than prescribing explicit reconfiguration plans at design time in anticipation of possible reconfiguration scenarios, developers need only specify the relative importance of each functional and non-functional concern to apply Plato-MDE.

Run-time Generation of Safe Adaptation Paths. An *adaptation path* comprises a series of reconfiguration steps. To prevent loss of state or introduction of erroneous results during a reconfiguration, a *safe* adaptation path preserves dependency relationships and ensures component communications are not interrupted [14, 15, 2]. Although multiple safe adaptation paths may exist for a given situation, the identification and selection process is non-trivial, as different solutions may represent tradeoffs between reconfiguration costs, performance, and reliability. Hermes is an evolutionary computation-based approach for automatically generating adaptation paths that safely transition an executing system from its current configuration to its desired target configuration. Hermes harnesses the process of evolution to efficiently explore parts of a vast solution space comprising all possible adaptation paths. Moreover, instead of focusing on a single criterion when generating adaptation paths, Hermes evolves solutions that balance competing objectives between functional and non-functional requirements, such as minimizing reconfiguration costs while maximizing reconfiguration performance and reliability. Additionally, Hermes can be applied at design time to generate alternative adaptation paths, and at run time to generate safe adaptation paths that handle changing system and environmental conditions.

Strategies for Mitigating Negative Impacts of Environmental Uncertainty. Early system requirements and domain assumptions are often ambiguous and idealized, thus leading to inconsistencies between a system specification and its behavior at run time [16, 17]. To augment goal-oriented models with more comprehensive and realistic requirements, van Lamsweerde and Letier proposed systematic techniques, heuristics, and formal techniques for reasoning about obstacles nd partial goal satisfaction. However, as DASs increase in complexity and become intertwined with the physical elements, including the environment, it becomes increasingly impractical for a human to exhaustively explore the system and environmental conditions that may adversely impact a DAS [1, 5].

We have developed LOKI, [1], an automated technique for identifying combinations of system and environmental conditions that obstruct design goals. LOKI can be leveraged to generate suites of test cases as well as suggest refinements to a goal model. LOKI, a domain-independent evolutionary computation-based approach, is designed to explore how uncertainty may obstruct goals in a DAS, where uncertainty refers to the unknown effects of system and environmental conditions upon goal satisfaction. Instead of searching for specific instances of goal obstructions, however, LOKI generates a diverse set of behaviors in a DAS that may lead to goal obstructions. To achieve this objective, LOKI leverages the concept of novelty search [18] to generalize, or collapse, vast collections of possible behaviors into fewer subsets of *different* behaviors in response to system and environmental conditions. Searching for novelty enables LOKI to discover both latent behaviors and requirements violations. While a requirements violation clearly obstructs a specific set of system goals, latent behaviors manage to satisfy requirements through unexpected and potentially undesirable behaviors. Furthermore, those unwanted behaviors may mean that requirements need to be modified to explicitly disallow the unwanted behavior. In addition, the set of system and environmental conditions that cause such behaviors can be reused to guide the testing process of implemented systems We have applied LOKI to an autonomous intelligent vehicle system (IVS) that performs adaptive cruise control and lane keeping while avoiding collisions. To this end, we leverage a set of IVS goal-oriented requirements models to implement an IVS prototype in the Webots simulation platform [19]. Experimental results show LOKI is able to discover combinations of system and environmental conditions that lead to latent behaviors and requirements violations. Furthermore, the set of behaviors discovered by LOKI may be analyzed to identify elements in the goal model that may require new or augmented obstacle mitigations, as well as additional constraints to disallow undesirable latent behaviors.

[1]In Norse mythology, Loki is the god of mischief and trickery.

References

[1] J. Whittle, P. Sawyer, N. Bencomo, Betty H.C. Cheng, and J.-M. Bruel, “RELAX: Incorporating uncertainty into the specification of self-adaptive systems,” in *In the Proceedings of the 17th International Requirements Engineering Conference (RE ’09)*, (Atlanta, Georgia, USA), pp. 79–88, IEEE Computer Society, September 2009.

[2] J. Zhang, Betty H.C. Cheng, Z. Yang, and P. K. McKinley, *Enabling safe dynamic component-based software adaptation*, vol. 3549 of *Lecture Notes in Computer Science*, pp. 194–211. Springer Berlin / Heidelberg, 2005.

[3] J. Zhang and Betty H.C. Cheng, “Model-based development of dynamically adaptive software,” in *Proceedings of the 28th International Conference on Software Engineering*, (New York, NY, USA), pp. 371–380, ACM (Distinguished Paper Award), 2006.

[4] J. Zhang, H. J. Goldsby, and Betty H.C. Cheng, “Modular verification of dynamically adaptive systems,” in *Proceedings of the Eighth International Conference on Aspect-Oriented Software Development*, 2009.

[5] Betty H.C. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, “A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty,” in *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS’09)*, Lecture Notes in Computer Science, (Denver, Colorado, USA), pp. 468–483, Springer-Verlag, October 2009.

[6] H. J. Goldsby and Betty H.C. Cheng, “Automatically generating behavioral models of adaptive systems to address uncertainty,” in *Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems*, (Berlin, Heidelberg), pp. 568–583 (Distinguised Paper Award), Springer-Verlag, 2008.

[7] H. J. Goldsby, Betty H.C. Cheng, P. K. McKinley, D. B. Knoester, and C. A. Ofria, “Digital evolution of behavioral models for autonomic systems,” in *Proceedings of the Fifth IEEE International Conference on Autonomic Computing*, (Chicago, Illinois), pp. 87–96 (Best Paper Award), IEEE Computer Society, 2008.

[8] H. J. Goldsby and Betty H.C. Cheng, “Automatically discovering properties that specify the latent behavior of uml models,” in *Proceedings of the 13th International Conference on Model Driven Engineering Languages and Systems*, (Oslo, Norway), pp. 316–330, Springer-Verlag, October 2010.

[9] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and Betty H.C. Cheng, “Composing adaptive software,” *Computer*, vol. 37, no. 7, pp. 56–64, 2004.

[10] A. J. Ramirez and Betty H.C. Cheng, “Adaptive monitoring of software requirements,” in *To Appear in the Proceedings of the 2010 Workshop on Requirements at Run Time*, (Sydney, Australia), 2010.

[11] A. J. Ramirez, D. B. Knoester, Betty H.C. Cheng, and P. K. McKinley, “Applying genetic algorithms to decision making in autonomic computing systems,” in *Proceedings of the Sixth*

International Conference on Autonomic Computing, (Barcelona, Spain), pp. 97–106 (Best Paper Award), June 2009.

[12] A. J. Ramirez, D. B. Knoester, Betty H.C. Cheng, and P. K. McKinley, "Plato: A genetic algorithm approach to run-time reconfiguration of autonomic computing systems," *To Appear in the Cluster Computing*, 2010.

[13] A. J. Ramirez and Betty H.C. Cheng, "Evolving models at run time to address functional and non-functional adaptation requirements," in *Proceedings of the Fourth Workshop on Models at Run Time*, vol. 509, (Denver, Colorado, USA), pp. 31–40, ACM, October 2009.

[14] J. Kramer and J. Magee, "The evolving philosophers problem: Dynamic change management," *IEEE Trans. on Soft. Eng.*, vol. 16, no. 11, pp. 1293–1306, 1990.

[15] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *Future of Software Engineering 2007*, (Minneapolis, Minnesota), pp. 259–268, IEEE Computer Society, May 2007.

[16] A. van Lamsweerde and E. Letier, "Integrating obstacles in goal-driven requirements engineering," in *Proceedings of the 20th International Conference on Software Engineering*, (Kyoto, Japan), pp. 53–62, IEEE Computer Society, 1998.

[17] A. van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Transactions on Software Engineering*, vol. 26, pp. 978–1005, October 2000.

[18] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," in *Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI)*, (Cambridge, MA, USA), MIT Press, 2008.

[19] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.