The Journal of Systems and Software xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Introduction to the special issue on state of the art in engineering self-adaptive systems

Danny Weyns[a,*], Sam Malek[b], Jesper Andersson[a], Bradley Schmerl[c]

[a] Linnaeus University, Sweden
[b] George Mason University, USA
[c] Carnegie Mellon University, USA

ARTICLE INFO

Article history:
Received 15 July 2012
Accepted 16 July 2012
Available online xxx

Keywords:
Self-adaptive systems
Autonomic systems
Self-organization

ABSTRACT

Researchers and engineers have been studying self-adaptation for over a decade, which has resulted in a vast body of knowledge. Nevertheless, as technology progresses and software systems are increasingly integrated, new challenges emerge. Among these challenges are the need for new theoretical models for self-adaptation, methods to verify and validate self-adaptive systems, and disciplined engineering approaches to support decentralization of control in self-adaptive systems. Tackling these challenges requires a cross-disciplinary approach. The goal of this special issue is to provide an overview of the state of the art in the field of self-adaptive software systems. From 61 submission, 13 papers were selected for publication. These papers demonstrate that the integration of different research fields that is required to tackle the challenges in engineering self-adaptation is underway. We offer the papers of this special issue as a benchmark on the current state of the art, and an exposition of key ideas and directions for further work.

1. Introduction

The house in which you live, the car you drive, the plane you fly, the TV you look at, are artifacts that have stable foundations. These artifacts, as many others we use in our daily lives, are physical: they are designed before they are built and used, and are rarely changed during their lifetime. Software is fundamentally different. Software is not a physical construct: it can easily be modified at any time, even while it is in use. This modification can occur in response to changes in requirements and operating conditions. However, the growing complexity of software systems makes modifying them, while maintaining functionality and quality of service, an increasingly difficult task to get right. The primary objective of self-adaptation is to enable software systems to deal with changes themselves, autonomously. The key challenge is then: how to engineer such self-adaptive systems in a disciplined manner?

Researchers and engineers have been studying principles of self-adaptive software systems for a long time. This has resulted in a vast body of knowledge. The ICSE 1998 paper of Oreizy and colleagues is an important milestone in understanding the underlying fundaments of self-adaptation (Oreizy et al., 1998). The authors pointed out that self-adaptation requires a runtime representation of the system that is kept in synchrony with the actual system. This runtime representation allows the system to reason about itself and adapt when needed. IBM's autonomic manager (Kephart and Chess, 2003) and Rainbow (Garlan et al., 2004) are two influential examples that realize this idea. Various researchers have argued that software architecture provides the right level of abstraction and generality to deal with self-adaptation (Oreizy et al., 1999; Garlan and Schmerl, 2002; Kramer and Magee, 2007).

As technology progresses and stakeholder requirements evolve, software systems are increasingly integrated, within and across ownership domains, posing new challenges for self-adaptation.

Recent community efforts have identified the need for new theoretical models for self-adaptation (Dobson et al., 2006; de Lemos et al., 2012), methods to verify and validate self-adaptive systems (Magee and Maibaum, 2006; Tamura et al., 2012), and disciplined engineering approaches to support decentralization of control in self-adaptive systems (Kramer and Magee, 2007; Weyns et al., 2012) as key challenges, among others. Tackling these challenges requires a cross-disciplinary approach (Cheng et al., 2009), where the know-how from architecture-based adaptation is synergistically merged with know-how from domains such as control theory, runtime verification, context-aware computing, self-organization, and multi-agent systems. The goal of this special issue is to provide an overview of the state of the art in the field of self-adaptive software systems. The call for papers was particularly successful

* Corresponding author.
E-mail address: Danny.Weyns@cs.kuleuven.be (D. Weyns).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.07.045

and attracted 61 submissions. All submitted papers went though a rigorous multiple-staged review process, and finally 13 high-quality papers were selected for publication.

2. Overview of the issue

The papers of the special issue cover a variety of topics in the space of engineering self-adaptive systems. We have grouped the papers in five categories based on key aspects considered in them: control loops, runtime models, decentralized control, formal foundations, and integrated perspectives. However, these categories are not disjunct and several of the papers could belong to more then one category.

2.1. Control loop

Patikirikorala et al. investigate a control engineering approach focusing on performance management of software systems. The authors present different types of control schemas and demonstrate their usefulness in a range of experiments that were performed under different operating conditions. The paper by Eracar et al. uses a feedback control loop to control the behavior of a satisfaction problem solving algorithm. It demonstrates significant performance gain for two different NP-hard constraint satisfaction problems. Peng et al. present a control-based method for self-tuning of different quality properties. The authors employ goal models that represent runtime requirements. These models are used to make tradeoff decisions by a preference-based goal reasoning mechanism. The decisions are mapped to optimal architectural reconfigurations for the actual operating conditions.

2.2. Runtime models

In their paper, Amoui et al. introduce a model-centric approach to support fine-grained adaptations of software systems. Central to the approach are graph-based models of the software that are interpreted at runtime to manage system adaptations. Wu et al. present a non-intrusive online monitoring approach to dynamically analyze data-centric properties for multi-participant service-based applications. The approach not only considers constraints on the sequence of exchanged messages, but also exploits the content of messages. In this way, the approach extends existing monitoring patterns in the area of service-based systems. Abebe and Ryan study dynamic offloading of computational tasks in pervasive environments. The authors present an approach where each device maintains a graph of components running in local memory, combined with an abstraction of components running on remote devices. Evaluation of computationally heavy applications shows a significant improvement in communication costs, memory needs, power consumption, and efficiency of adaptations.

2.3. Decentralized control

Khakpour et al. present a framework to model large-scale ecosystems that integrates central control with self-organization. The formal foundation allows combined used of model checking at design time with runtime verification to verify structural and behavioral adaptation properties. This combined use of formal methods at design time and runtime is necessary to manage the complexity of the hybrid approach. Pruteanu and Dulman present LossEstimate, a fully decentralized approach that allows online estimation of communication failures in large-scale wireless networks. The estimates provide dynamic approximation of message loss in the network. This gossip-like approach exhibits small communication overhead and fast convergence time for different types of network topologies.

2.4. Formal foundations

Perz-Palacin et al. propose an adaptation framework that can be used to reduce power consumption in a computing infrastructure by tuning the number of servers and their operating frequency. The framework employs stochastic Petri Nets to guarantee a proper balance between energy consumption and system performance. In their paper, Xu et al. propose ADAM, an approach to identify defects in context-aware adaptive systems. The approach monitors runtime errors and relates the errors to responsible defects in the application. ADAM relies on formally defined adaptation semantics that are exploited by assert checkers to detect errors. Analyzing what to monitor and when to adapt in order to guarantee requirements in complex domains is a difficult problem, so Salifu et al. introduce an approach that encodes the monitoring and adaptation problem as propositional logic constraints. A SAT solver is then used to decide between monitoring and adaptation options.

2.5. Integrated perspective

Hallsteinsen et al. present an integrated methodology and development framework for adaptive software systems, focusing on ubiquitous and dynamic computing environments. The approach follows the model-driven paradigm and is supported by a middleware that facilitates dynamic adaptations at runtime. Cheng and Garlan introduce the Stitch language to express repair strategies that map to business objectives. These strategies allow an architecture-based self-adaptation framework to select a strategy for adaptation with optimal utility in a given context.

This summary demonstrates that the integration of different research fields that is required to tackle the challenges in engineering self-adaptation is underway. Nevertheless, there remains plenty of room for researchers to contribute to this undertaking, and push the field forward. We offer the papers of this special issue as a benchmark on the current state of the art, and an exposition of key ideas and directions for further work.

Acknowledgments

We thank all the authors for submitting their work to the special issue. We are grateful to the reviewers for their excellent work. Finally, we thank the editor in chief for his support throughout the process of preparing this special issue, and JSS for hosting the special issue.

References

Cheng, B., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Serugendo, G.M., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G., Kienle, H., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D., Whittle, J., 2009. Software engineering for self-adaptive systems: a research roadmap. In: Software Engineering for Self-Adaptive Systems, vol. 5525. Lecture Notes in Computer Science. Springer.

de Lemos, R., Giese, H., Müller, H., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N., Vogel, T., Weyns, D., Baresi, L., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Desmarais, R., Dustdar, S., Engels, G., Geihs, K., Goeschka, K., Gorla, A., Grassi, V., Inverardi, P., Karsai, G., Kramer, J., Lopes, A., Magee, J., Malek, S., Mankovskii, S., Mirandola, R., Mylopoulos, J., Nierstrasz, O., Pezzè, M., Prehofer, C., Schäfer, W., Schlichting, R., Smith, D.B., Sousa, J., Tahvildari, L., Wong, K., Wuttke, J., 2012. Software engineering for self-adaptive systems: a second research roadmap. In: Software Engineering for Self-Adaptive Systems II, vol. 7475, Lecture Notes in Computer Science. Springer.

Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F., 2006. A survey of autonomic

Please cite this article in press as: Weyns, D., et al., Introduction to the special issue on state of the art in engineering self-adaptive systems. J. Syst. Software (2012), http://dx.doi.org/10.1016/j.jss.2012.07.045

communications. ACM Transactions on Autonomous and Adaptive Systems 1 (December (2)), 223–259, http://doi.acm.org/10.1145/1186778.1186782.
Garlan, D., Cheng, S., Huang, A.C., Schmerl, B., Steenkiste, P., 2004. Rainbow: architecture-based self-adaptation with reusable infrastructure. IEEE Computer 37, 46–54.
Garlan, D., Schmerl, B., 2002. Model-based adaptation for self-healing systems. In: ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02), Charleston, SC, 18–19 November.
Kephart, J., Chess, D., 2003. The vision of autonomic computing. Computer 36 (1).
Kramer, J., Magee, J., 2007. Self-managed systems: an architectural challenge. Future of Software Engineering.
Magee, J., Maibaum, T., 2006. Towards specification, modelling and analysis of fault tolerance in self managed systems. In: Software Engineering for Adaptive and Self-Managing Systems.
Oreizy, P., Gorlick, M.M., Taylor, R.N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D.S., Wolf, A.L., 1999. An architecture-based approach to self-adaptive software. IEEE Intelligent Systems 14 (May (3)), 54–62 http://dx.doi.org/10.1109/5254.769885.
Oreizy, P., Medvidovic, N., Taylor, R., 1998. Architecture-based runtime software evolution. In: International Conference on Software Engineering.
Tamura, G., Villegas, N., Muller, H., Sousa, J., Becker, B., Karsai, G., Mankovskii, S., Pezze, M., Schafer, W., Tahvildari, L., Wong, K., 2012. Towards practical runtime verification and validation of self-adaptive software systems. In: Software Engineering for Self-Adaptive Systems II, vol. 7475. Lecture Notes in Computer Science. Springer.
Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Goeschka, K., 2012. On patterns for decentralized control in self-adaptive systems. In: Software Engineering for Self-Adaptive Systems II, vol. 7475, Lecture Notes in Computer Science. Springer.

Danny Weyns is an Associate Professor in the Computer Science Department at Linnaeus University. He received a Ph.D. from the Katholieke Universiteit Leuven, Belgium in 2006 and was a postdoc at the same university between 2006 and 2011. His research interests are in software architecture, self-adaptive systems, multi-agent systems, and middleware for decentralized systems. The current focus of Danny's research is on a formally founded approach for engineering decentralized self-adaptive software systems.

Sam Malek is an Assistant Professor in the Computer Science Department at George Mason University. His research interests are in software architecture, autonomic computing, and software dependability. Malek received his Ph.D. and M.S. degrees in Computer Science from the University of Southern California, and his B.S. degree in Information and Computer Science from the University of California Irvine. He is a member of IEEE, ACM, and ACM SIGSOFT.

Jesper Andersson is an Associate Professor in the Computer Science Department at Linnaeus University. He received a Ph.D. from Linköping University in 2007 on dynamic software architectures. His research interests are in software architecture, self-adaptive systems, software engineering processes, and software product lines.

Bradley Schmerl is a Senior Systems Scientist in the School of Computer Science at Carnegie Mellon University. He received his Ph.D. from Flinders University in Adelaide, South Australia in 1997 and was an Assistant Professor at Clemson University between 1998 and 2000. His interests include software architecture, self-adaptive systems, pervasive computing systems and software development environments.

Please cite this article in press as: Weyns, D., et al., Introduction to the special issue on state of the art in engineering self-adaptive systems. J. Syst. Software (2012), http://dx.doi.org/10.1016/j.jss.2012.07.045