Report on the Dagstuhl Seminar on Software Engineering for Self-Adaptive Systems

Rogério de Lemos[1], Holger Giese[2], Hausi A. Müller[3], and Mary Shaw[4]

[1] University of Kent, UK
r.delemos@kent.ac.uk
[2] Hasso Plattner Institute, University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de
[3] University of Victoria, Canada
hausi@cs.uvic.ca
[4] Carnegie Mellon University, USA
mary.shaw@cs.cmu.edu

Abstract. Softwares ability to adapt at run-time to changing user needs, system intrusions or faults, changing operational environment, and resource variability has been proposed as a means to cope with the complexity of todays software- intensive systems. Such self-adaptive systems can configure and reconfigure themselves, augment their functionality, continually optimise themselves, protect themselves, and re- cover themselves, while keeping most of their complexity hidden from the user and administrator. In this paper, we present research road map for software engineering of self- adaptive systems focusing on four views, which we identify as essential: design spaces, verification and validation, processes, and decentralisation.

Keywords: software engineering, self-adaptive systems, design spaces, verification and validation, processes, decentralization

1 Introduction

The simultaneous explosion of information and integration of technology together with the continuous evolution from software intensive systems to systems of systems to ultra-large-scale (ULS) systems requires new and innovative approaches for building, running and managing software systems [ULS 2006]. A consequence of this continuous evolution is that software systems are expected to become more versatile, flexible, resilient, dependable, robust, continuously available, energy-efficient, recoverable, customisable, self-healing, configurable, or self-optimising by adapting to changing requirements and contexts/environments [Roadmap 2008]. One of the most promising approaches to achieving such properties is to equip software systems with self-managing capabilities using self-adaptation mechanisms.

Research on the theory and practice of self-adaptation is highly interdisciplinary, and it draws ideas and solutions from many diverse fields, such as control

Dagstuhl Seminar Proceedings 10431
Software Engineering for Self-Adaptive Systems
http://drops.dagstuhl.de/opus/volltexte/2011/3088

engineering and dynamical systems, automation and instrumentation, machine learning and planning, fault-tolerance and reactive systems, and many others. The applications of self-adaptation also span a wide range: autonomic computing [Huebscher 2008], dependable computing, autonomic communications and networks [Dobson 2006], mobile ad hoc networks, sensor networks, ubiquitous computing, computing systems management [Hellerstein 2004], biologically-inspired computing, user-interface customisation, embedded computing, service-oriented architectures, web-service composition, embedded systems, mechatronics, mobile and autonomous robots, multi-agent systems, to financial systems.

The seminar focused on software engineering aspects of building self-adaptive and self-managing systems. The topic of self-adaptive systems has been studied independently within the different research areas of software engineering, including requirements engineering, modelling, architecture and middleware, event-based, component-based and knowledge-based systems, testing, verification and validation, as well as software maintenance and evolution [Dagstuhl 2008]. Recently several workshops have emerged to bring these independent efforts together by concentrating on the software engineering aspects of self-adaptive systems: WOSS (Workshop on Self-Healing Systems), WADS (Workshop on Architecting Dependable Systems), DEAS (Design and Evolution of Autonomic Application Software), SEAMS (Software Engineering for Self-Adaptive and Self-Managing Systems), and Dagstuhl Seminar 08031 on Software Engineering for Self-Adaptive Systems [Dagstuhl 2008].

The flexible nature of software provides an ideal platform for self-adaptation. However, the proper realisation of the self-adaptation functionality remains a formidable intellectual challenge. In the long run, we need to establish the foundations that enable the systematic development of future generations of self-adaptive systems. Therefore the current achievements have to be integrated into a more comprehensive overall research effort from which generic approaches should be devised. Building self-adaptive software systems cost-effectively and in a systematic and predictable manner is also a major engineering challenge.

The goal of this seminar was to bring together the leading software engineering experts and other distinguished experts from related fields on self-adaptive systems to discuss the fundamental principles, models, methods, techniques, mechanisms, state-of-the-art, and challenges for engineering self-adaptive software systems.

2 Topics of discussion

The aim of the Seminar was not so much to be comprehensive concerning the topics associated with software engineering for self-adaptive software systems, but to be focused on key and challenging topics.

– **Design spaces**: The development of complex software involves models at different levels of granularity. Also all approaches to self-adaptation which plan the effect of changes ahead of their application require models as a

means for prediction. Similarly, approaches to estimate the quality of a solution at development-time also needs models to evaluate the quality of the adaptation. Following different trends these models may be oriented towards control theory featuring related analysis capabilities [Hellerstein 2004] or support a more architecture centric view [Kramer 1996], [Kramer 2007]. Other relevant aspects are their partial or global character as well as the capability to adjust the models to observations of the system and environment to also address un-anticipated dynamic changes. Therefore, the question which models are required for the different development steps of self-adaptive systems is of paramount importance. Design spaces represent the set of design alternatives as decisions to be made, along with the alternatives for each decision and criteria that guide the choice. The decisions are often organised hierarchically.

- **Verification and validation**: The vision of self-adaptive systems is promising but also risky. Giving systems the freedom to self-adapt at run-time obviously weakens our capabilities to assure their proper operation. Therefore, an important question is how much run-time assurance is needed in addition and above design-time assurance. Moreover, it is crucial to identify properties which are only assessable at run-time (e.g., stability) and thus irrelevant in the contact of traditional static systems. What are the validation requirements associated with each type of adaptation? How can these obligations become a visible part of the development process?
- **Processes**: Activities such as selecting a single execution strategy which are traditionally handled at development-time may be deferred to run-time in a self-adaptive system to gain more flexibility and the capability to adapt to the current situation at run-time. It seems important to identify which activities are often deferred to run-time as well as for which activities this seem still not possible respectively useful.
- **Decentralisation**: Self-adaptive systems can be viewed from two perspectives, either individual systems or cooperative systems. Individual self-adaptive systems evaluate their own global behaviour and change it when the evaluation indicates that they are not accomplishing what they were intended to do, or when better functionality or performance is possible realising self-adaptation. In a centralised fashion. such systems typically operate with an explicit internal representation of themselves and their global goals. On the other hand, self-adaptive systems can be composed of a large number of components that interact according to local and typically simple rules in a decentralised manner. The global behaviour of the system emerges from these local interactions, and it is difficult to deduce properties of the global system by studying only the local properties of its parts. Such systems do not necessarily use internal representations of global properties or goals. These two perspectives of self-adaptive system represent two extremes going from full centralized to a fully decentralized. In practice, there is clearly a continuum of designs and implementations where both system types are intertwined to achieve their behavioural goal.

3 Outcomes

The two concrete outcomes from this Seminar will be a new roadmap paper and a new book. The roadmap paper, which will follow the same format of the previous paper will be structured according to the new topics to be identified during the seminar, namely: design spaces, processes, verification and validation, and decentralisation. For each topic, the objective is to summarise the current state-of-the-art, discuss its limitations, and identify future challenges for the field.

The book will contain state-of-the-art contributions from participants of the seminar and some invited contributions. In addition to these contributions, the roadmap paper will be the introductory chapter of the book, which should be followed by four chapters containing extended versions of the topics discussed in the roadmap paper. The book will be published by Springer as Lecture Notes in Computer Science volume on their State-of-the-Art series.

References

[Dagstuhl 2008] Schloss Dagstuhl Seminar 08031. Software Engineering for Self-Adaptive Systems, Wadern, Germany (2008) http://www.dagstuhl.de/08031/

[Dobson 2006] Dobson, S., Denazis, S., Fernndez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A Survey of Autonomic Communications. ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1(2):223-259 (2006)

[Hellerstein 2004] Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. John Wiley & Sons (2004)

[Huebscher 2008] Huebscher, M.C., McCann, J.A.: A Survey of Autonomic Computing-Degrees, Models, and Applications. ACM Computing Surveys, 40 (3):7:1-28 (2008)

[Kramer 1996] Kramer, J., Magee, J.: Dynamic Structure in Software Architectures. ACM SIGSOFT Software Engineering Notes 21(6):3-14 (1996)

[Kramer 2007] Kramer, J., Magee, J.: Self-managed Systems: An Architectural Challenge. In: Future of Software Engineering (FoSE 2007), pp. 259-268, IEEE Computer Society, Washington, DC, USA (2007)

[Roadmap 2008] Cheng, B.H.C., et al: A Research Roadmap: Software Engineering for Self-Adaptive Systems. Schloss Dagstuhl Seminar 08031 Report on Software Engineering for Self-Adaptive Systems, Wadern, Germany, 12 pages (2008)

[ULS 2006] * Northrop, L., Feiler, P., Gabriel, R., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., Wallnau, K.: Ultra-Large-Scale SystemsThe Software Challenge of the Future. Technical Report, Software Engineering Institute, Carnegie Mellon University, 134 pages (2006) http://www.sei.cmu.edu/uls