ISSN 2186-7437

NII Shonan Meeting Report

No. 2013-7

Engineering Adaptive Software Systems (EASSy)

Shinichi Honiden, National Institute of Informatics, Japan
Zhenjiang Hu, National Institute of Informatics, Japan
Hausi Müller, University of Victoria, Canada
John Mylopoulos, University of Trento, Italy
Yijun Yu, The Open University, United Kingdom

September 9-12, 2013

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Engineering Adaptive Software Systems (EASSy)

Organizers:
Shinichi Honiden, National Institute of Informatics, Japan
mailto:honiden@nii.ac.jp
Zhenjiang Hu, National Institute of Informatics, Japan
mailto:hu@nii.ac.jp
Hausi Müller, University of Victoria, Canada
mailto:hausimuller@gmail.com
John Mylopoulos, University of Trento, Italy
mailto:john.mylopoulos@gmail.com
Yijun Yu, The Open University, United Kingdom
mailto:y.yu@open.ac.uk

September 9-12, 2013

Objectives

As software-intensive systems continue to invade all aspects of personal, business and social life, they are required to operate in ever more open and dynamic environments where the one constant is uncertainty. Coping with such uncertainty calls for systems that monitor their environment and adapt so that they can continue to fulfill their requirements. The problem of engineering such systems is being addressed in a number of research communities, including Software Engineering, Systems, Ubiquitous Computing, Service-Oriented Computing, Multi-Agent Systems, Robotics and more.

As a result of research efforts within these communities, there have been many proposals on how to engineer such adaptive systems. Some are policy- / requirements-based, others biologically-inspired, still others focus on awareness as the key facility that leads to adaptivity. The main objective of the proposed workshop is to bring together some of the authors of these proposals so that they can compare and contrast their respective approaches. In the process, we hope that participants in the workshop will go away with a better understanding of what ideas work, and under what circumstances. Some of the more specific issues to be discussed at the meeting include:

1. How do we engineer adaptive software systems? What are the concepts, tools and techniques that can support requirements elicitation, architectural design and implementation of such systems?

2. How do we reengineer legacy software systems in order to turn them into adaptive ones?

3. Comparative review of adaptation mechanisms in Robotics, Multi-Agent Systems, Software Engineering, Socio-Technical Systems, Ubiquitous Computing, etc.

4. Usability issues for adaptive software systems. How do we ensure effective human interaction with complex software systems that have adaptive components?

5. Evolution of adaptive software systems. How do deployed adaptive systems evolve? How can we ensure convergence and stability for such systems?

6. Evolution and control of systems-of-systems, where each component system has its own requirements and its own adaptation mechanism. How do we ensure convergence, stability and coherence for such systems-of-systems?

7. Runtime models: most of the approaches to adaptivity are model-based in the sense that they deploy models of system requirements and the domain to support monitoring, diagnosis and compensation in the case of failure. How are such runtime models different from their design counterparts? How do we reason with runtime models to support adaptation functions, i. e., monitoring, diagnosis and compensation?

8. How do we prevent failures for such systems through run-time reasoning mechanisms? Since such mechanisms are inherently intractable, how can we support incremental run-time reasoning that predicts and/or prevents failures? Workshop format

This four-day EASSy workshop has been held at the Shonan Village Center near Tokyo. The format of the workshop consisted of presentations and discussions. Senior presenters delivered position statements on comprehensive approaches to adaptive software system engineering, other presentations described on-going work through group presentations.

Meetings schedule and participants

```
Schedule, participants, and introductory videos
Arrival Day--Sep 8
19:00--21:30  Welcome Reception

Day 1--Mon, Sep 9
7:30--8:30  Breakfast

8:30--9:00  Session 1

NII (National Institute of Informatics) by Shinichi Honiden
Shonan Meetings by Zhenjiang Hu
Welcome and Overview
9:00--10:30  Session 2
```

Hausi Müller (Organizer), University of Victoria, Canada
Norha Villegas, Icesi University, Colombia
Gabriel Tamura, Icesi University, Colombia
10:30--11:00 Break

11:00--12:00 Session 3

Hoh Peter In, Korea University, Korea
Jeong-Dong Kim, Korea University, Korea
12:00--13:15 Lunch

13:15--14:30 Session 4

Bashar Nuseibeh, The Open University, UK & Lero, Ireland
Mazeiar Salehie, Lero, Ireland
Liliana Pasquale, Lero, Ireland
14:30--15:00 Break

15:00--17:00 Session 5

Introductory Video
Yijun Yu (Organizer), The Open University, UK
Arosha K. Bandara, The Open University, UK
Pierre Akiki, The Open University, UK
Lionel Montrieux, The Open University, UK
18:00--19:30 Dinner

Day 2--Tue, Sep 10
6:00--7:30 Hike to Lookout Tower

7:30--8:45 Breakfast

9:00--10:30 Session 6

John Mylopoulos (Organizer), University of Trento, Italy
Ivan Jureta, University of Namur, Belgium
Vítor E. Silva Souza, Federal University of Esprito Santo (Ufes), Brazil
10:30--11:00 Break

11:00--12:00 Session 7

Patrick Martin, Queen's University, Canada
Marin Litoiu, York University, Canada
12:00--13:15 Lunch

13:15--15:00 Session 8

Shinichi Honiden (Organizer), National Institute of Informatics, Japan
Zhenjiang Hu, National Institute of Informatics, Japan
Soichiro Hidaka, National Institute of Informatics, Japan

Kenji Tei, National Institute of Informatics, Japan
Fuyuki Ishikawa, National Institute of Informatics, Japan
15:00--15:30 Break

15:30--17:00 Session 9

Bihuan Chen, Fudan University, China
Xin Peng, Fudan University, China
Guiling Wang, North China University of Technology, China
Lin Liu, Tsinghua University, China
18:00--19:30 Dinner

19:15-21:15 Sing along

Day 3--Wed, Sep 11
7:30--8:45 Breakfast

9:00--10:30 Session 10

Announcements
Rogério de Lemos, University of Kent, UK
Danny Weyns, Linnaeus University Sweden
Muhammad Usman Iftikhar, Linnaeus University, Sweden
10:30--11:00 Break

11:00--12:00 Session 11

Tetsuo Tamai, Hosei University, Japan
Scott Hissam, SEI, USA
12:00--13:30 Lunch

13:30--19:00 Excursion (afternoon)

19:00--21:30 Banquet in Kamakura

Day 4--Thu, Sep 12
6:00--7:30 Hike to Lookout Tower

7:30--8:45 Breakfast

9:00--10:30 Session 12

Giordano Tamburrelli, Università della Svizzera Italiana, Switzerland
Valerio Panzica La Manna, Politecnico di Milano, Italy

10:30--11:00 Break

11:00--12:00 Working Session to Design Workshop Book

12:00--13:00 Lunch

13:00   Departure

Presentation abstracts

Here is a list of abstracts of the talks as delivered.

Engineering Adaptive Privacy

Arosha Bandara
Department of Computing and Communications
The Open University
United Kingdom

As mobile computing applications have become commonplace, it is increasingly important for them to address end-users' privacy requirements. Mobile privacy requirements depend on a number of contextual socio-cultural factors to which mobility adds another level of contextual variation. However, traditional requirements elicitation methods do not sufficiently account for contextual factors and therefore cannot be used effectively to represent and analyse the privacy requirements of mobile end users. On the other hand, methods that investigate contextual factors tend to produce data that does not lend itself to the process of requirements extraction. To address this problem we have developed a Privacy Requirements Distillation approach that employs a problem analysis model to extract and refine privacy requirements for mobile applications from raw data gathered through empirical studies involving end users. We demonstrate our approach using qualitative data from an empirical study of a mobile social networking application.

Model-Based Self-Adaptation from Requirements to Architectures: A Decision-Making Process

Bihuan Chen
School of Computer Science
Fudan University
China

Model-based self-adaptation has been proposed as one way to dynamically tune parameters, change structures and behaviors of software systems in response to the changing environments. Specifically, requirements and architecture models have been used for self-adaptation. However, requirements-driven approaches usually assume that requirements elements can be directly mapped to architectural elements, and architecture-based approaches usually assume that requirements are well-understood at design time and unchanged at runtime. In this talk, we propose to combine requirements-driven and architecture-based self-adaptations. Requirements adaptations capture requirements as goal models to reason about the best plan within the problem space, and architecture-based adaptations capture architectural design decisions as decision trees to

search for the best design for the desired requirements. Following these adaptations, component-based architecture models are reconfigured using incremental and generative model transformations.

References

[1] G. S. Blair, N. Bencomo, and R. B. France. Models@run.time. Computer, 42(10):22-27, 2009.

[2] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste. Rainbow: Architecture-based self-adaptation with reusable infrastructure. Computer, 37(10):46-54, 2004.

[3] Y. Wang and J. Mylopoulos. Self-repair through reconfiguration: A requirements engineering approach. In ASE, pages 257-268, 2009.

[4] B. Chen, X. Peng, Y. Yu, and W. Zhao. Are your sites down? Requirements-driven self-tuning for the survivability of Web systems. In RE, pages 219-228, 2011.

[5] X. Peng, B. Chen, Y. Yu, and W. Zhao. Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop. J. Syst. Softw., 85(12):2707-2719, 2012.

Exploration of Adaptation Space - Linking with Efforts in Service-Oriented Computing

Fuyuki Ishikawa
National Institute of Informatics
Japan

There have been a lot of efforts on self-adaptation in the area of service-oriented computing. Systems are composed by combining existing services, and adapt typically by selecting and switching from alternative services. General problems of service composition select a service for each involved task for given global goals (functions, quality (non-functional) constraints, and optimization criteria).

Our work extends this setting to select a set of candidate services for each task, to avoid greedy non-optimal adaptation and to reduce runtime overhead for adaptation. As a technical approach, our work uses graphs to model slightly-different functions of similar services. This approach allows to efficiently identify alternative services as well as to see the potential trade-offs between strength of goals and availability of alternatives. Although this work targeted the communities of world wide web and service-oriented computing, it shows an instance study about how to identify and explore adaptation space (inside solution space) at development-time and at runtime.

Self-Adaptive Software Systems: Properties and Assessment

Gabriel Tamura

Icesi University
Colombia

In recent years the dynamic capabilities of self-adaptive software-intensive systems have proliferated and improved significantly. However, these dynamic capabilities must be systematically guaranteed in their migration to the industry before reaching the society. To advance the field of self-adaptive and self-managing systems and leverage their benefits it is necessary to develop methods and tools to assess and possibly certify adaptation properties of self-adaptive systems, not only at design time but also, and especially, at run-time. In previous works we first proposed a framework for evaluating quality-driven self-adaptive software systems [9]. This framework is based on a survey of self-adaptive system papers and a set of adaptation properties derived from control theory properties. Moreover, these properties are mapped to software quality attributes, and thus, corresponding software quality metrics can be used to assess adaptation properties.

Second, in [5] we proposed several ways of integrating software validation and verification (V&V) measures in the MAPE-K loop to ensure that adaptation mechanisms produce software adaptations that satisfy user requirements and expected quality attributes throughout their lifecycle. Given that high levels of adaptation and autonomy provide new ways for software systems to operate in highly dynamic environments, developing certifiable V&V methods for guaranteeing the achievement of self-adaptive software goals is still one of the major challenges facing the entire research field. Therefore, in this paper we identify highlight fundamental challenges and concerns for the development of V&V methods and techniques that provide certifiable trust in self-adaptive and self-managing systems; and present a proposal for evaluating properties inherent to self-adaptive software systems.

References

[1] N. Villegas, H. Müller, G. Tamura, et al., "A Framework for Evaluating Quality-driven Self-Adaptive Software Systems", in *Proceedings 6th International Symposium ICSE (SEAMS)*, 2011, pp. 80–89, ACM.

[2] G. Tamura, N. Villegas, H. Müller, et al., "Towards Practical Runtime Verification and Validation of Self-Adaptive Software Systems", *LNCS*, 2013, vol. 7475, pp. 108–132, Springer.

Managing Non-Functional Uncertainty via Model-Driven Adaptivity

Giordano Tamburrelli
Universitá della Svizzera Italiana
Switzerland

Modern software systems are often characterized by uncertainty and changes in the environment in which they are embedded.

Hence, they must be designed as adaptive systems. In this talk we discuss a framework that supports adaptation to non-functional manifestations of uncertainty. The proposed framework allows engineers to derive, from an initial model of the system, a finite state automaton augmented with probabilities.

The system is then executed by an interpreter that navigates the automaton and invokes the component implementations associated to the states it traverses. The interpreter adapts the execution by choosing among alternative possible paths of the automaton in order to maximize the system's ability to meet its non-functional requirements. We also discuss the implementation of the proposed solution and its application to an adaptive application inspired by an existing worldwide distributed mobile application.

User-Driven Situational Service Mashups

Guilin Wang
North China University of Technology
China

Situational applications are created for a narrow group of users with a unique set of needs to deal with situational and ad-hoc problems. For such kind of situational applications, the requirements are in variety, can't be determined in advance and may change over time. As a typical approach to build situational applications, service mashups come into our attention.

Though there are some existing service mashup tools, it is still challenging for those developers with no or little programming skills to develop service mashups to solve the transient and ad-hoc business problems. This talk begins with an interesting motivating scenario and discusses some of the research issues including the data service modeling, end-user service programming. Some research results and an on-going work on service recommendation are briefly introduced. The initial work results have been implemented in a prototype called "data service space(DSS)", which is briefly introduced in this talk.

Software Engineering at Runtime – Situation-Aware Smart Applications

Hausi A. Müller
Department of Computer Science
Faculty of Engineering
University of Victoria

With the rise of the Industrial Internet [1] the world entered a new era of innovation [2]. At the heart of this new industrial revolution is the convergence of the global industrial system with computing power, low-cost sensing, big data, predictive analytics, and ubiquitous connectivity. The growing proliferation of smart devices and applications is accelerating the convergence of the physical and the digital worlds [5, 6, 7, 8]. Smart apps allow users, with the help of sensors and networks, to do a great variety of things [4], from tracking their friends to controlling remote devices and machines [3].

At the core of such smart systems are self-adaptive systems that optimize their own behaviour according to high-level objectives and constraints to address changes in functional and nonfunctional requirements as well as environmental conditions [9]. Self-adaptive systems are implemented using four key technologies: runtime models, context management, feedback control theory [10], and run-time verification and validation [5, 11].

The proliferation of highly dynamic and smart applications challenges the software engineering community in re-thinking the boundary between development time and run time and developing techniques for adapting systems at run time. The key challenge is to automate traditional software engineering, maintenance and evolution techniques to adapt and evolve systems at run time with minimal or no human interference [12, 15].

Hitherto, most developers did not instrument their software with sensors and effectors to observe whether requirements are satisfied in an evolving environment at run time. One way to break out of this mold is to make the four key technologies readily accessible at run time [9].

References

[1] Financial Post, "The Industrial Internet Economy: Canada 2025", May 2013. http://business.financialpost.com/category/industrial-internet/?__lsa=d7be-94a3

[2] P.C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines", *GE Technical Report*, http://www.ge.com/sites/default/files/Industrial_Internet.pdf, Nov. 2012.

[3] IBM Corp., "How to compete in the era of *smart*", http://www.ibm.com/smarterplanet/, July 2013.

[4] IBM Corp.: The Internet of Things, YouTube Video, March 2010. http://www.youtube.com/watch?v=sfEbMV295Kk.

[5] M.A. Feki, F. Kawsar, M. Boussard, and L. Trappeniers (Bell Labs), "The Internet of Things: The Next Technological Revolution, Guest Editor's Introduction", *IEEE Computer*, vol.46, no.2, pp. 24–25, Feb. 2013.

[6] G. Kortuem, A.K. Bandara, N. Smith, M. Richards, and M. Petre, "Educating the Internet-of-Things Generation", *IEEE Computer*, vol.46, no.2, pp. 53–61, Feb. 2013.

[7] S. Balasubramaniam and J. Kangasharju, "Realizing the Internet of Nano Things: Challenges, Solutions, and Applications", *IEEE Computer*, vol.46, no.2, pp. 62–68, Feb. 2013.

[8] M. Singhal, S. Chandrasekhar, T. Ge, R.S. Sandhu, R. Krishnan, G.J. Ahn, and E. Bertino, "Collaboration in Multicloud Computing Environments: Framework and Security Issues", *IEEE Computer*, vol.46, no.2, pp. 76–84, Feb. 2013.

[9] M. Litoiu and J. Mylopoulos (Eds.), "Procs. 8th ACM/IEEE International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2013)", 2013.

[10] R.M. Murray, "Control in an information rich world: Report of the panel on future directions in control, dynamics, and systems", *SIAM*, 2003.

[11] R. de Lemos, H. Giese, H.A. Müller and M. Shaw (Eds.), "Software Engineering for Self-Adaptive Systems", *LNCS 7475*, Number 10431 in Dagstuhl Seminar. Springer, Feb. 2013.

[12] H.M. Müller and N.M. Villegas, "Evolving highly adaptive software systems", in *Evolving Software Systems*, T. Mens, A. Serebrenik, and A. Cleve (eds.), Springer, 33 pages, July 2013. In press.

[13] G. Tamura, N.M. Villegas, H.A. Müller, J.P. Sousa, B. Becker, B., G. Karsai, S. Mankovskii, M. Pezzé, W. Schäfer, L. Tahvildari and K. Wong, "Towards practical runtime verification and validation of self-adaptive software systems", in *LNCS*, 2013, vol. 7475, pp. 108–132, Springer.

[14] N.M. Villegas, H.A. Müller, G. Tamura, L. Duchien, and R. Casallas, "A Framework for Evaluating Quality-driven Self-Adaptive Software Systems", in *Proceedings 6th International Symposium ICSE (SEAMS)*, 2011, pp. 80–89, ACM.

[15] S. Balasubramanian, R. Desmarais, H.A. Müller, U. Stege, and S. Venkatesh, "Characterizing problems for realizing policies in self-adaptive and self-managing systems", in *Proc. 6th Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, May 2011, pp. 70–79.

Dynamic Self-adaptive Software Technology in System of Systems

Hoh Peter In
Korea University
Korea

The smartphone revolution has made great impact not only on people's lifestyle, but also on software-intensive system design. Stand-alone systems have been transformed and connected into an eco-system to survive in highly-open, dynamic-changing, and unpredictable IT environment. Moreover, a single eco-system begins to connect with other ones (called "mega-ecosystems") to provide seamless smart services. However, the few research community has focused on self-adaptiveness of mega-ecosystems software.

In this talk, we will pay attention on importance of "self-adaptiveness" in mega-ecosystems and investigate the research issues of self-adaptive system of systems (SASoS), which would be a key component of mega-ecosystems. How to define self-adaptiveness of system of systems? What are the key components and their architecture? How to collaborate one with another in SASoS? Given a new component of SASoS, how does it join/register in an existing SASoS? How to share knowledge (collective intelligence) of self-adaptiveness among SASoS components? Since most stand-alone systems are non-adaptive ones, how to transform them into SASoS? We expect this workshop would be a good starting point of exploring these issues among the research community.

Requirements problem and solution concepts for adaptive systems

Ivan Jureta
University of Namur
Belgium

When we do Requirements Engineering (RE) for adaptive systems, are we doing RE as usual? Do adaptive systems pose new challenges to RE? If yes, what kinds of challenges are these? The talk gives preliminary answers to these questions.

The main argument of the talk, is that the requirements problem and solution concepts for adaptive systems are different from the de facto standard problem and solution concepts in RE, introduced by Pamela Zave and Michael Jackson in their seminal 1997 ACM Transactions on Software Engineering and Methodology paper, "Four dark corners of requirements engineering". In the talk, I argue that the solution concept in adaptive systems RE is not a single specification as in Zave and Jackson's requirements problem, but more specifications along with evolution requirements that impose constraints on which specifications we can switch between at runtime, whereby switching occurs when we observe that the system has failed to satisfy requirements to the desired extent. I relate this to key ideas in RE for adaptive systems, including monitoring, feedback loops, probabilistic relaxation, fuzzy relaxation, and evolution requirements.

Engineering Adaptive Software Systems: A Requirements Engineering Perspective

John Mylopoulos
University of Trento
Italy

Our work on software adaptation is founded on the premise that software adapts when it fails to meet its requirements or when its requirements say so. Our framework assumes that requirements are (stakeholder) goals and generally have many solutions, each consisting of a specification (of system behaviour). A system can't be adaptive unless it operationalizes several of these behaviours. Adaptation takes place when one or more requirements fail. Adaptation amounts to having the system switch from one behaviour to another, with an expectation that the new behaviour will work better. Adaptation is driven by an *adaptation mechanism* that takes into account the failures-at-hand (divergence from requirements), and picks an alternative behaviour among those available by exploiting qualitative relations between alternative behaviours and requirements.

There are two classes of special requirements that determine what monitoring, analysis, planning and execution (the MAPE loop of autonomic architectures) will be realized in the adaptation mechanism. The first class consists of *awareness requirements* that define limits on the amount of runtime failure that can be tolerated. For example, if R = "Schedule meeting", then R' = "R will not fail more than 4% of the time during any one month period" is an awareness requirement. The second class consists of evolution requirements that define changes over time to other requirements. For instance, “If R fails, try R^- instead”, where $R^- =$“Schedule conference call” is an example of an evolution requirement. Notice that evolution requirements specify what has to change, awareness requirements do not.

It is important to distinguish between adaptation and evolution of software systems, just as biology has done for biological systems. In biology, individuals adapt and species evolve,both to survive. Adaptation involves switching to an alternative behaviour, evolution involves changing the genetic makeup/blueprint and physical structure of a species. By analogy, we say that a particular software system (e.g., the MacOS running on my machine) adapts if it switches at run-time to an alternative behaviour. A software system (the MacOS sold by Apple in September 2013) constitutes a species and evolves if its requirements change and the system architecture, implementation, etc. change in accordance. Software individuals adapt, software species evolve. Evolution entails requirements changes, adaptation entails no changes to requirements. Evolution mechanisms are very different from adaptation ones, and are much harder to come by.

The results of our work on adaptive software systems can be found in three PhD theses: Vítor Souza’s work [7] focused on the design of feedback loops and proposed many of the concepts mentioned above. Yiqiao Wang’s thesis [2] concentrated on monitoring and analysis (diagnosis), while Fabiano Dalpiaz’s work [5] looked at adaptive systems as multi-agent systems. Two other theses have explored the notion of variability in goal-oriented requirements models and served as useful starting points for our work on adaptation mechanisms: Sotirios Liaskos [1] and Alexei Lapouchnian [4]. Finally, the notion of requirements evolution is explored in the thesis of Neil Ernst [6]. For each of these theses there have been several publications.

References

[1] Sotirios Liaskos, ”Acquiring and Reasoning about Variability in Goal Models”, PhD dissertation, Department of Computer Science, University of Toronto, June 2008.

[2] Yiqiao Wang, ”Monitoring and Diagnosis for Autonomic Systems: A Requirements Engineering Approach”, PhD dissertation, Department of Computer Science, University of Toronto, November 2009.

[3] Lei Jiang, ”Data Quality By Design: A Goal-Oriented Approach”, PhD dissertation, Department of Computer Science, University of Toronto, December 2009.

[4] Alexei Lapouchnian, ”Exploiting Requirements Variability for Software Customization and Adaptation”, PhD dissertation, Department of Computer Science, University of Toronto, November 2010.

[5] Fabiano Dalpiaz, ”Exploiting Contextual and Social Variability for Software Adaptation”, PhD dissertation, Department of Information Engineering and Computer Science (DISI), University of Trento, January 2011.

[6] Neil Alexander Ernst, "Software Evolution: A Requirements Engineering Approach", PhD dissertation, Department of Computer Science, University of Toronto, December 2011.

[7] Vítor E. Silva Souza, "Requirements-based Software System Adaptation", PhD dissertation, Department of Information Engineering and Computer Science (DISI), University of Trento, June 2012.

Composition-based Interaction Design for Adaptable Distributed Software Systems

Kenji Tei
National Institute of Informatics
Japan

Distributed systems embedded in physical word will face changes in network or physical world, and such changes will affect quality of the system. To satisfy its requirements, the distributed system should be adapted in response to these changes in order to satisfy its requirements.Modern approach for designing self-adaptive software takes external approach, where self-adaptive system consists of adaptable software and adaptation engine. To enable self-adaptation, the adaptable software should support one or more solutions tightly coupled with requirements, and provide adaptation APIs to change solutions even at runtime.

Interaction is high level behavior design of distributed system. Interactions of self-adaptive distributed system should be traceable from requirements and be changeable to enable adaptation. However traceability link between requirements and interactions are unclear. It is hard to localize changes in interactions to satisfy one failed requirement because one interaction is related to one or more requirements, and one requirements is related to one or more interaction. To mitigate this problem, we propose composition-based interaction design approach [1]. In this approach, a piece of interaction is designed to be related to a requirement, and complete interaction is achieved by composing one or more pieces. This approach can maintain clear traceability link between requirement and interaction so that changes to satisfy one failed requirement can be localized in the corresponding piece of interaction.

References

[1] R. Takahashi, F. Ishikawa, K. Tei, and Y. Fukazawa, "Intention based Automated Composition Approach for Coordination Protocol", in *The IEEE 20th International Conference on Web Services (ICWS)*, 2013.

Engineering Adaptive Digital Investigations using Forensic Requirements

Liliana Pasquale
Lero
Ireland

A digital forensic investigation aims to collect and analyse the evidence necessary to demonstrate a potential hypothesis of a digital crime. Despite the availability of several digital forensics tools, investigators still approach each crime case from scratch, postulating potential hypotheses and analysing large volumes of data. This talk proposes to explicitly model forensic requirements in order to engineer software systems that are forensic-ready and guide the activities of a digital investigation. Forensic requirements relate some speculative hypotheses of a crime to the evidence that should be collected and analysed in a crime scene. In contrast to existing approaches, we propose to perform proactive activities to preserve important - potentially ephemeral - evidence, depending on the risk of a crime to take place. Once an investigation starts, the evidence collected proactively is analysed to assess if some of the speculative hypotheses of a crime hold and what further evidence is necessary to support them. For each hypothesis that is satisfied, a structured argument is generated to demonstrate how the evidence collected supports that hypothesis. Obtained results suggest that our approach provides correct investigative findings and reduces significantly the amount of evidence to be collected and the hypotheses to be analysed.

Software Systems Adaptation by Composition

Lin Liu
Tsinghua University
China

Software Systems running in the Internet are facing an ever-changing environment, which requires constant monitoring and adaptation supporting facilities need to be in place to ensure that user's needs are satisfied on-demand. In order to understand the requirements of adaptive systems and discuss requirements engineering strategies and techniques in response to the needs of engineering adaptive systems, I would set out from example adaptation mechanism in biological systems and control systems to give my observation on the general model of adaptation by composition.

In particular, a theoretical typology of different levels of adaptation capability is proposed, and their correspondence with the key characteristics of adaptive systems is concretized, namely, the static, reactive, adaptive and collaborative systems. The typology is built on a common architecture using rule base to store knowledge for run-time use. The accompanying development method includes a series of steps to allow run-time adaptation, and a goal-oriented modeling method and notation to analyze possible requirements changes at run-time. A general process to support for run-time adaptation of service-oriented system will also be mentioned to support the accompanying development method.

Runtime Adaptation for Reliability

Lionel Montrieux
Department of Computing and Communications
The Open University
United Kingdom

Some complex software systems, such as Eclipse, are made of several components that can be installed, run, stopped, and updated independently. This provides several advantages over distributing the software as a single components: updates to specific components can be distributed without having to re-package the entire software; several organisations can provide components and features independently from the core of the software; users can choose to only install and use the features they want or need; dependencies between components allow for a wide variety of working configurations; etc.

However, the ability to deploy a custom configuration, as well as varying levels of quality depending on the vendors that release components, can sometimes lead to bugs or crashes in components. In systems with lots of components and complex dependencies, reverting a component to a previous version can be tedious if done manually.

In this talk, we propose to automate the downgrade of faulty components, using a 3-steps approach:

- identify the faulty component from an error stack-trace;
- compute, given a suitable proximity function, the closest configuration that allows for the faulty component to be downgraded, while keeping dependencies between components satisfied, and minimising the changes to be made to the configuration;
- updating the configuration so the user can continue using the software, almost without interruption.

We also present a case study we are currently running, using Eclipse and OSGi, which involves hundreds of components. We also describe another case study we would like to look at, using the Gentoo Linux distribution, which involves over 15,000 packages.

Adaptive Security: A Requirements-Driven Approach

Mazeiar Salehie
Lero
Ireland

Security is primarily concerned with protecting valuable assets from harms. Identifying and evaluating assets are therefore key activities in any security engineering process - from modeling threats and attacks, discovering existing vulnerabilities, to selecting appropriate countermeasures. However, despite their crucial role, assets are often neglected during the development of secure software systems. Indeed, many systems are designed with fixed security boundaries and assumptions, and without the possibility to adapt when assets change unexpectedly, new threats arise, or undiscovered vulnerabilities are revealed. To handle such changes, systems must be capable of dynamically enabling different security countermeasures. The proposed requirements-driven framework in this talk promotes assets as first-class entities in engineering secure software systems. An asset model is related to requirements, expressed through a goal model, and the objectives of an attacker, expressed through a threat model. These models are then used as input to build a causal network to analyze system security in different situations, and to enable, when necessary, a set of countermeasures to mitigate security threats. The causal network is conceived as a runtime entity that tracks relevant changes that may arise at runtime, and enables a new set of countermeasures. The framework is illustrated with the aid of two domain applications, mobile phone security and physical access control.

ActivFORMS: Active Formal Models for Self-Adaptive Systems

Muhammad Usman Iftikhar and Danny Weyns
Linnaeus University
Sweden

An important challenge for self-adaptive systems is to provide evidence that the system goals are satisfied during operation, regarding the uncertainty of changes that may affect the system, its environment or its goals. To provide such evidence, state of the art approaches propose to equip the feedback loop with formal models of the managed system, the context in which it executes and its goals (or parts of these). These models are updated at runtime to handle uncertainties and support decision making for adaptations. However, existing approaches do not provide guarantees about the behavior of the adaptation functions themselves, which may ruin the adaptation capabilities. Furthermore, the focus is mainly on parametric uncertainty (e.g., likelihood of faults, service availability), but not on structural uncertainty (e.g., adding new goals), which refer to unanticipated changes that typically require dynamic updates, including updates of the feedback loop. In this talk, we introduce Active FORmal Model for Self-adaptation (ActivFORMS) that uses a formal model of the complete feedback loop. This model is directly executed at runtime by a virtual machine to realize self-adaptation. ActivFORMS assures that the verified system goals are guaranteed at runtime, and enables runtime updates of the formal model to support unanticipated changes. We show how we have used ActivFORMS for different adaptation scenarios in a small-scale robotic system.

Dynamic Context Management and Reference Models for Dynamic Self-Adaptation

Norha M. Villegas
Icesi University
Colombia

Our society is increasingly demanding situation-aware smarter software (SASS) systems, whose goals change over time and depend on context situations. A system with such properties must sense their dynamic environment and respond to changes quickly, accurately, and reliably, that is, to be context-aware and self-adaptive. A relevant problem addressed in our research work is the dynamic management of context information, with the goal of improving the relevance of SASS systems' context-aware capabilities with respect to changes in their requirements and execution environment. Therefore, one of our research motivations is the investigation of dynamic context management and self-adaptivity to: (i) improve context-awareness and exploit context information to enhance quality of user experience in SASS systems [9], and (ii) improve the dynamic capabilities of self-adaptivity in SASS systems [10].

Context-awareness and self-adaptivity pose significant challenges for the engineering of SASS systems. Regarding context-awareness, a first challenge is the impossibility of fully specifying environmental entities and the corresponding monitoring requirements at design-time. A second challenge arises from the continuous evolution of monitoring requirements due to changes in the system caused by self-adaptation. As a result, context monitoring strategies must be modeled and managed in such a way that they support the addition and deletion of context types and monitoring conditions at runtime. For this, the user must be integrated into the dynamic context management process. Concerning self-adaptivity, a third challenge is to control the dynamicity of adaptation goals, adaptation mechanisms, and monitoring infrastructures, and the way they affect each other in the adaptation process. This is to preserve the effectiveness of context monitoring requirements and thus self-adaptation.

To address these challenges we have proposed several contributions [1]. First, the personal context sphere concept to empower users to control the life cycle of personal context information in user-centric SASS systems [3, 4]. Second, the SmarterContext ontology that supports the modeling of context information and its monitoring requirements while supporting changes in these models at runtime [5, 2]. Third, an efficient context processing engine that discovers implicit contextual facts from context information specified in changing context models [7, 8]. Fourth, a reference model for designing highly dynamic self-adaptive systems [10], for which the continuous pertinence between monitoring mechanisms and both changing system goals and context situations is a major concern.

References

[1] Norha Villegas, "Context management and self-adaptivity for situation-aware smart software systems", *Dissertation University of Victoria*, 2013, http://dspace.library.uvic.ca:8080/handle/1828/4476.

[2] G. Tamura, N. Villegas, H. Müller, L. Duchien, and L. Seinturier, "Improving Context-Awareness in Self-Adaptation using the DYNAMICO Reference Model", in *Proceedings 8th International I Symposium SEAMS*, 2013, pp. 153–162, ACM.

[3] N. Villegas and H. Müller, "The SmarterContext Ontology and its Application to the Smart Internet: A Smarter Commerce Case Study", *LNCS*, 2013, vol. 7855, pp. 151–184, Springer.

[4] N. Villegas, H. Müller, et al. "DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems", *LNCS*, 2013, vol. 7475, pp. 265–293, Springer.

[5] G. Tamura, N. Villegas, H. Müller, et al., "Towards Practical Runtime Verification and Validation of Self-Adaptive Software Systems", *LNCS*, 2013, vol. 7475, pp. 108–132, Springer.

[6] N. Villegas, H. Müller, "Managing Dynamic Context to Optimize Smart Interactions and Services", in *Springer-Verlag, Berlin, Heidelberg*, 2010, pp. 289–318.

[7] S. Ebrahimi, N. Villegas, H. Müller, and A. Thomo, "SmarterDeals: A Context-aware Deal Recommendation System based on the SmarterContext Engine", in *CASCON*, 2012, pp. 116–130, ACM.

[8] J. Muñoz, G. Tamura, N. Villegas, and H. Müller, "Surprise: User-controlled Granular Privacy and Security for Personal Data in SmarterContext", in *CASCON*, 2012, pp. 131–145, ACM.

[9] N. Villegas, H. Müller, et al., "A Framework for Evaluating Quality-driven Self-Adaptive Software Systems", in *Proceedings 6th International Symposium ICSE (SEAMS)*, 2011, pp. 80–89, ACM.

[10] N. Villegas, H. Müller, et al., "A Dynamic Context Management Infrastructure for Supporting User-driven Web Integration in the Personal Web", in *CASCON*, 2011, pp. 200–214, ACM.

Exploiting Big Data in Engineering Adaptive Cloud Services

Paul Martin
Queen's University
Canada

Our research into adaptive cloud services has shown that adaptation can require the storage and analysis of potentially large amounts of data [1, 2]. The position I argue in my talk is that big data has an important role to play in the engineering of adaptive software systems, in general, and adaptive cloud services, in particular.

I use our own research to support my claim. In our research into QoS-aware management for cloud services big data analytics are required in a number of the component services, specifically cloud provider recommendation, workload forecasting, performance prediction and monitoring. In each of these component services big data is used to generate models to facilitate the necessary decision-making. We see that the big challenge in all these cases is the need to adapt the models when the workload and/or environment changes. We also observe that, with the creation and management of these models, model management will be an important addition to the adaptive middleware.

References

[1] P. Martin, S. Soltani, W. Powley and M. Hassannezhad, "QoS-Aware Cloud Application Management", in *Advances in Parallel Computing: Clouds, Big Data and Data-Intensive Computing*, February 2013.

[2] R. Mian, P. Martin, F. Zulkernine and J. Vazquez-Poletti, "Towards Building Performance Models for Data-intensive Workloads in Public Clouds", in *Proc of 4th International Conference on Performance Engineering*, Prague, Czech Republic, April 2013, pp. 259–270.

Engineering Adaptive User Interfaces for Enterprise Applications

Pierre Akiki
Department of Computing and Communications
The Open University
United Kingdom

Enterprise applications (e.g., enterprise resource planning) help in managing an enterprise's functional business areas such as: Accounting, marketing, etc. However, existing research (e.g., [10]) has shown that these applications suffer from usability problems. One of the main causes behind this problem is their inability to cater for the variety in the end-users' needs. Adaptive behavior has been suggested as a means for enhancing usability [5] and some works particularly suggested applying it to enterprise application UIs [9].

The primary objective of this research is devising a general-purpose platform for building adaptive enterprise application UIs based on a runtime model-driven approach. To achieve this objective we presented an architecture (CEDAR) [1], a UI adaptation mechanism (RBUIS) [2], and a supporting tool (Cedar Studio) [3].

The CEDAR architecture [1] serves as a reference for devising adaptive model-driven enterprise application UIs. This architecture is based on the (1) Three Layer Architecture [8] (Adaptive System Layering), (2) CAMELEON reference framework [6] (UI Abstraction), and (3) Model-View-Controller paradigm (Implementation). CEDAR promotes the use of interpreted runtime models instead of code generation for providing more flexibility in performing advanced UI adaptations at runtime.

The Role-Based UI Simplification (RBUIS) mechanism [2] is based on CEDAR and combines role-based access control [7] with adaptive behavior for simplifying UIs. In RBUIS, roles are divided into groups representing the aspects based on which the UI will be simplified such as computer literacy, job title, etc. RBUIS supports feature-set minimization by assigning roles to task models for providing users with a minimal feature-set based on the context-of-use. The assignment could be done by I.T. personnel but there is also a potential for engaging end-users in this process [4]. Layout optimization is supported by assigning roles to workflows that represent adaptive UI behavior visually and through code, and can be executed on concrete UI models. Furthermore, RBUIS supports user feedback for refining the adaptation operations.

The Cedar Studio IDE [3] provides tool support for building enterprise applications based on the CEDAR architecture. Cedar Studio allows developers and I.T. personnel to apply RBUIS using a set of visual design and code editing tools that support the creation of UI models and adaptive behavior. Automatic generation between the UI levels of abstraction is supported with the possibility to make manual changes at any level.

References

[1] Akiki, P.A., Bandara, A.K., and Yu, Y. Using Interpreted Runtime Models for Devising Adaptive User Interfaces of Enterprise Applications. Pro-

ceedings of the 14th International Conference on Enterprise Information Systems, SciTePress (2012), pp. 72–77.

[2] Akiki, P.A., Bandara, A.K., and Yu, Y. RBUIS: Simplifying Enterprise Application User Interfaces through Engineering Role-Based Adaptive Behavior. Proceedings of the fifth ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM (2013), pp. 3–12.

[3] Akiki, P.A., Bandara, A.K., and Yu, Y. Cedar Studio: An IDE Supporting Adaptive Model-Driven User Interfaces for Enterprise Applications. Proceedings of the fifth ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM (2013), pp. 139–144.

[4] Akiki, P.A., Bandara, A.K., and Yu, Y. Crowdsourcing User Interface Adaptations for Minimizing the Bloat in Enterprise Applications. Proceedings of the fifth ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM (2013), pp. 121–126.

[5] Benyon, D. Adaptive systems: A solution to usability problems. User Modeling and User-Adapted Interaction 3, 1 (1993), pp. 65–87.

[6] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, (2003), pp. 289–308.

[7] Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., and Chandramouli, R. Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security 4, 3 (2001), pp. 224–274.

[8] Kramer, J. and Magee, J. Self-Managed Systems: an Architectural Challenge. Proceedings of the Workshop on the Future of Software Engineering, IEEE (2007), pp. 259–268.

[9] Singh, A. and Wesson, J. Evaluation criteria for assessing the usability of ERP systems. Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, ACM (2009), pp. 87–95.

[10] Topi, H., Lucas, W.T., and Babaian, T. Identifying Usability Issues with an ERP Implementation. Proceedings of the International Conference on Enterprise Information Systems, SciTePress (2005), pp. 128–133.

Architecting Resilience: Handling Malicious and Accidental Threats

Rogério de Lemos
University of Kent, UK
Universidade de Coimbra, Portugal

Resilience is the persistence of service delivery that can justifiably be trusted, when facing changes. While architecting is the art and science of creating and building complex systems, and which covers the following basic activities: scope, structure and certification. One important aspect of resilience is the provision of assurances, and these are obtained by building arguments about system resilience. However in order to build arguments, one needs collect, structure and analyse evidence in which in self-adaptive systems can be obtained either at development-time or run-time time.

This talk has covered three contributions in which architecting resilience can be effectively employed in the handling of accidental and malicious threats. In the first contribution, we have described how for self-adaptive software systems integration testing can be performed at run-time. On itself this activity should be implemented as a feedback control loop, which should be associated with the analysis activity of the autonomic MAPE-K [1].

The second contribution described a stepwise progress for the provision of assurances about the resilience of self-adaptive software systems, and it covered the following topics: (i) resilience evaluation based on environmental stimuli in which probabilistic model-checking is used for obtaining levels of confidence [2], (ii) resilience evaluation by comparing adaptation mechanisms of self-adaptive software systems [3], (iii) robustness evaluation of controllers by injecting faults into the probes of Rainbow [4], (iv) effectiveness of architecture-based self-adaptation by evaluating the effort of evolving industrial middleware into an architectural-based self-adaptive software system [3], finally (v) robustness-driven resilience evaluation of self-adaptive software systems in which system properties are evaluated by injecting faults [6].

The third contribution described an approach based on self-adaptation as a means to improve the management of malicious behaviour, by adapting authorization policies and access rights [5]. The goal is to adapt to mitigate malicious behaviour, and prevent future attacks.

References

[1] J. Camara, R. de Lemos, M. Vieira, R. Almeida, and R. Ventura, "Architecture-Based Resilience Evaluation for Self-Adaptive Systems", *Computing Journal (Special "Software Architecture for Code Testing and Analysis")*, 2013, vol.95, no.8, pp. 689-722.

[2] J. Camara and R. de Lemos, "Evaluation of resilience in self-adaptive systems using probabilistic model-checking", in *Proceedings of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2012)*, Zurich, Switzerland. June 2012. pp. 53-62.

[3] J. Camara, P. Correia, R. de Lemos, D. Garlan, P. Gomes, B. Schmerl, and R. Ventura, "Evolving an Adaptive Industrial Software System to Use Architecture-Based Self-Adaptation", in *Proceedings of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2013)*, San Francisco, CA, USA. May 2013. pp. 13-22.

[4] J. Camara, R. de Lemos, N. Laranjeiro, R. Ventura, and M. Vieira, "Robustness Evaluation in Self-Adaptive Software Systems", in *Latin American Symposium on Dependable Computing (LADC 2013)*. Rio de Janeiro, RJ, Brazil. April 2013. pp. 1-10.

[5] C. Bailey, D. W. Chadwick, R. de Lemos, and K. W. S. Sui, "Enabling the Autonomic Management of Federated Identity Providers", in *7th In-*

ternational Conference on. Autonomous Infrastructure, Management and Security (AIMS 2013), June 2013, UPC Barcelona, Spain. 2013. pp. 100-111.

[6] C. E. da Silva and R. de Lemos, “Dynamic plans for integration testing of self-adaptive software systems”, in *Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011)*, Honolulu, HI, USA. May 2011. pp. 148-157.

Coordinating Architecture-Based Self-Protecting Systems

Scott Hissam
SEI
USA

Industry and governments are pushing for reuse and commonality in software-reliant systems to achieve economies of scale, and better integration between systems. However, this also provides economies of scale to cyber-attackers, who can devise an attack and reuse it on several systems that share common artifacts. Techniques employed to thwart these attacks, such as moving target (MT) defense, persistently interfere with other quality attributes, and can make it more costly to produce, and maintain these systems. The aim of this project is to enable systems that use common architectures and components to securely share information about their threat environment such that the individual systems can proactively adapt their behavior and structure in time to deny the possibility of reusing the attack, without the impact on other quality attributes that other techniques impose. Thus, this project will turn the advantage that reuse and commonality give to attackers into a defense advantage.

References

[1] Cheng, Shang-Wen; Poladian, Vahe; Garlan, David; and Schmerl, Bradley. “Improving architecture-based self-adaptation through resource prediction.” In *Software Engineering for Self-Adaptive Systems*, pp. 71-88. Springer, 2009.

[2] De Lemos, R.; Giese, H.; Müller, H.; Shaw, M.; Andersson, J.; Litoiu, M.; Schmerl, B. et al. “Software engineering for self-adaptive systems: A second research roadmap.” In *Software Engineering for Self-Adaptive Systems II*, Springer, 2013.

[3] Evans, D.; Nguyen-Tuong, A.; and Knight, J. “Effectiveness of moving target defenses.” In *Moving Target Defense*, pp. 29-48. Springer, 2011.

[4] Garlan, D.; Cheng, S-W; Huang, A-C.; Schmerl, B; and Steenkiste, P., “Rainbow: Architecture-based self-adaptation with reusable infrastructure.”, *Computer*, vol.37, IEEE, 2004.

[5] Garlan, D.; Schmerl, B.; and Cheng, S-W. ”Software architecture-based self-adaptation.” In *Autonomic computing and networking*, pp. 31–55. Springer, 2009.

[6] Poladian, V.; Garlan, D.; Shaw, M.; Satyanarayanan, M.; Schmerl, B.; Sousa, J., "Leveraging Resource Prediction for Anticipatory Dynamic Configuration." In *Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, 2007.

[7] Schmidt, D., "Towards Common Operating Platform Environments", *Software Engineering Institute*, 2012.

[8] Yuan, Eric; and Malek, Sam, "A taxonomy and survey of self-protecting software systems." In *Proceedings of Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, IEEE, 2012..

[9] Yuan, E.; Malek, S.; Schmerl, B.; Garlan, D.; and Gennari, J., "Architecture-based self-protecting software systems." In *Proceedings of the 9th international Conference on Quality of Software Architectures (QoSA'13).* ACM, 2013.

Designing Self-adaptive System using Control Loops

Shinichi Honiden
National Institute of Informatics
Japan

Self-adaptive System consists of Adaptation Engine and Adaptable Software. Adaptable Software is implemented by a set of control loops.

We define a process of elaboration for the goal model that extracts a set of control loops from the requirements descriptions as components that constitute extensible systems [1]. A control loop is a cyclic activity flow based on the process control model, which is a model for defining system behaviors. A controller in the model forms a (control) loop consisting of four key activities: collect, analyze, decide, and act. This cycle starts with a collection of relevant data that reflects the current state of the system. The system analyzes the collected data and adjusts its behavior to achieve its goals. After that, the system acts in a way that reflects decisions.

We regard such control loops to be independent components that prevent the impact of a change from spreading outside them. When additional requirements appear, new control loops are easy to embed into a system constructed with multiple control loops. Even if the requirements change, the idea of compositional control loops can easily accommodate these changes simply by updating the relevant control loops. We devised a goal model compiler [2] to enable partial design models to be extracted for each evolution and built a programming framework [3] that enables execution of multiple control loops. The goal model compiler identifies multiple control loops from goal-oriented requirements descriptions.

References

[1] H. Nakagawa, A. Ohsuga, and S. Honiden, "A goal model elaboration for localizing changes in software evolution", in *Proc. of 21st IEEE International Requirements Engineering Conference (RE'13).* IEEE, 2013.

[2] H. Nakagawa, A. Ohsuga, and S. Honiden, "gocc: A configuration compiler for self-adaptive systems using goal-oriented requirements description", in *Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'11)*, pp. 40–49. ACM, 2011.

[3] H. Nakagawa, A. Ohsuga, and S. Honiden, "Towards dynamic evolution of self-adaptive systems based on dynamic updating of control loops", in *Proc. of IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems (SASO'12)*, pp. 59–68. IEEE, 2012.

Bidirectional Graph Transformation Infrastructure and Its Applications

Soichiro Hidaka
National Institute of Informatics
Japan

Bidirectional transformation is a mechanism that maintains consistency between two artifacts while adapting to changes in one artifact by propagating them to the other artifact. Round-trip properties ensure the stability in the propagation.

GRoundTram (Graph Roundtrip Transformation for Models) has been developed as a bidirectional model transformation framework with the round-trip property, equipped with validation, debugging and traceability. Transformation language is based on Buneman et al.'s graph query language, and backward change propagation has been realized by traceability based on bulk semantics (edge-wise transformation) of the structural recursion of the language, enabling decomposition of changes into small changes for the edge-wise transformation.

Several recent attempts to apply GRoundTram have been introduced, including co-evolution of model and code, collaborative development of bio-models, and feedback of formal verification results.

References

[1] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Nakano: GRoundTram: An Integrated Framework for Developing Well-Behaved Bidirectional Model Transformations, Progress in Informatics, No. 10, Apr 2013

[2] S. Hidaka, Z. Hu, K. Inaba, H. Kato and K. Nakano, GRoundTram: An Integrated Framework for Developing Well-Behaved Bidirectional Model Transformations (short paper), 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011) pp.480-483 Nov 2011

[3] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, K. Nakano, Bidirectionalizing Graph Transformations, 15th ACM SIGPLAN International Conference on Functional Programming, pp.205-216 Sep 2010

[4] K. Inaba, S. Hidaka, Z. Hu, H. Kato, K. Nakano, Graph- Transformation Verification using Monadic Second-Order Logic, 13th International ACM

SIGPLAN Symposium on Principles and Practice of Declarative Programming, pp.17-28 Jul 2011

[5] S. Hidaka, Z. Hu, K. Inaba, H. Kato, K. Matsuda, K. Nakano and I. Sasano, Marker-directed optimization of UnCAL graph transformations, 21st International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2011) LNCS vol. 7225, pp.123-138 Jul 2011

[6] Y. Yu, Y. Lin, Z. Hu, S. Hidaka, H. Kato, L. Montrieux, Maintaining Invariant Traceability through Bidirectional Transformations, 34th International Conference on Software Engineering (ICSE 2012) pp.540-550 Jun 2012

[7] J. Wilson-Kanamori and S. Hidaka, A Bidirectional Collaboration Framework for Bio-Model Development, Second International Workshop on Bidirectional Transformations (BX 2013), Mar 2013, Rome, Italy, colocated with ETAPS 2013

[8] I. Sasano, Z. Hu, S. Hidaka, K. Inaba, H. Kato, K. Nakano, Toward bidirectionalization of ATL with GRoundTram, Proc. of the 4th International Conference

[9] S. Hidaka, K. Asada, Z. Hu, H. Kato, K. Nakano, Structural Recursion for Querying Ordered Graphs, 18th ACM SIGPLAN International Conference on Functional Programming, to appear, Sep 2013 [10] K. Asada, S. Hidaka, H. Kato, Z. Hu, K. Nakano, A Parameterized Graph Transformation Calculus for Finite Graphs with Monadic Branches,15th International Symposium on Principles and Practice of Declarative Programming, to appear, Sep 2013

[10] K. Nakano, S. Hidaka, Z. Hu, K. Inaba, H. Kato, View Updatability Checking with Simulation-based Graph Schema,, JSSST Computer Software 29(2) pp.174-192 Apr 2012

[11] S. Hidaka, Z. Hu, H. Kato, K. Nakano, Towards a Compositional Approach to Model Transformation for Software Development, ACM symposium on Applied Computing pp.468-475 Mar 2009

Role-Based Adaptive Modeling Framework 'Epsilon' and a Case Study

Tetsuo Tamai
Hosei University, Japan
collaborated with
Supasit Monpratarnchai (Fujitsu)

We propose an adaptive system design framework based on our "Epsilon" modeling concept and language.

The core of Epsilon modeling is the notion of roles and how roles interact each other in a defined context. In Epsilon, an environment is defined as a field of collaboration between roles and an object adapts to the environment assuming one of the roles. Objects can freely enter or leave environments and belong to multiple environments at a time so that dynamic adaptation or evolution of objects is realized. Environments and roles are the first class constructs at runtime as well as at model description time so that separation of concerns is not only materialized as a static structure but also observed as behaviors. Environments encapsulating collaboration are independent reuse components to be deployed separately from objects.

The Epsilon framework starts with the requirements phase, where the i* methodology is used for constructing RE models targeting Epsilon. Then, the model is represented in UML, extended with stereotypes: "context" and "role". The model is transformed into an executable code in EpsilonJ/Java, using the model transformation language ATL.

A case study was conducted, taking a problem from the multi-agent community, the "Traffic Jam Monitoring" problem. It requires context-driven dynamic collaboration between traffic monitoring devices.

Lastly, we briefly discuss the innate difficulty of constructing self-adaptive systems.

Dynamic Update and Self-Adaptation: can we fill the gap?

Valerio Panzica La Manna
Politecnico di Milano
Italy

Self-adaptive systems are currently designed to automatically react to changes that are defined at design time. However, modern software systems are also subject to continuous and unanticipated changes. Unanticipated changes may come from the environment and context in which the system is operating, or from the requirements, where new functionality needs to be added or existing functionality needs to be modified. Some of these systems, from financial transaction processing to safety critical applications, must also operate continuously and cannot be stopped. Research in Dynamic Software Updates has the goal of engineering systems that evolve at runtime, without the need of being stopped and restarted.

In this talk I'll provide an overview of a specification-oriented perspective of dynamic updates [1, 2]. We define under which condition a dynamic update is safe, and we provide an approach to automatically synthesize a dynamically updating system from changes in the specification. With the vision of having self-adaptive systems that are able to deal with unanticipated changes, we propose future directions for integrating this technique to the different steps of self-adaptation: from the goal model to the automatic generation of self-adaptive systems.

References

[1] Ghezzi, C.; Greenyer, J.; Panzica La Manna, V., "Synthesizing dynamically updating controllers from changes in scenario-based specifications," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 145–154, 4-5 June 2012.

[2] V.P. La Manna, J. Greenyer, C. Ghezzi, and C. Brenner, "Formalizing correctness criteria of dynamic updates derived from specification changes",

in *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'13)*, IEEE Press, Piscataway, NJ, USA, pp. 63-72.

Requirements-based Software System Adaptation in Practice: the Zanshin Framework

Vítor E. Silva Souza
Federal University of Espírito Santo
Brazil

In this talk I presented an overview of the Zanshin approach and framework for the design and development of adaptive systems. Zanshin is a Requirements Engineering-based framework for the design of adaptive systems that exploits concepts of Control Theory in order to make the elements of the feedback loops that provide adaptivity first class citizens in the requirements models. It has been developed during my PhD [1] in the University of Trento, Italy, with professor John Mylopoulos.

The Zanshin approach can be divided in three main activities that are concerned with the elicitation/modeling of: (i) Awareness Requirements [2] as indicators to be monitored; (ii) system parameters and how they affect indicators (for reconfiguration); and (iii) Evolution Requirements [3] as specific adaptation strategies (to be enacted when indicators 'fail'). At runtime, the Zanshin framework reads the specification produced in this process and is able to send adaptation instructions to the managed system.

Examples presented during the talk can be seen in detail in [1] (chapter 7) and the system is also available for download (https://github.com/sefms-disi-unitn/Zanshin), so others can repeat the same experiments or create their own.

References

[1] V.E.S. Souza, "Requirements-based Software System Adaptation", *Ph.D Thesis, University of Trento, Italy*, 2012.

[2] V.E.S. Souza, A. Lapouchnian, W.N. Robinson, and J. Mylopoulos, "Awareness Requirements", in *Software Engineering for Self-Adaptive Systems II*, vol. 7475, R. Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds. Springer, 2013, pp. 133–161.

[3] V.E.S. Souza, A. Lapouchnian, K. Angelopoulos, and J. Mylopoulos, "Requirements-driven software evolution", in *Computer Science - Research and Development*, pp. 1–19, 2012.

Human Factors in Self-Adaptive Socio-Technical Systems

Xin Peng
School of Computer Science
Fudan University
China

Most of the software systems today can be treated as the so-called socio-technical systems, in which human, organization, hardware and software components work in tandem to fulfill stakeholder requirements. These systems operate under uncertainty as components fail, humans act in unpredictable ways, and the environment of the system changes. Therefore, self-adaptation has been an essential capability of socio-technical systems and human factors play an important role in it [2].

Human factors in self-adaptive socio-technical systems can be considered from different perspectives.

By considering human as expert of business, design and IT infrastructure, we need to capture human expertise at different layers as knowledge bases of runtime adaptation and establish some kinds of multi-layered control and feedback loops at runtime [1, 3].

By considering human as system user, a socio-technical system needs to monitor and learn the changing and personalized quality requirements of individual users and adapt based on the learned user quality requirements.

By considering human as component, we can consider human as a part of human architecture, which interacts with software architecture and can also be adapted at runtime.

By considering human as agent, we need to support the multi-agent nature of socio-technical systems and consider to adapt each agent and its social collaborations to better achieve its goals by its own capabilities and social collaborations with others.

Human factors from the above perspectives and others need to be considered together with software techniques such as reconfigurable software architectures to better support self-adaptation in socio-technical systems.

References

[1] X. Peng, B. Chen, Y. Yu, and W. Zhao, "Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2707–2719, 2012.

[2] L. Fu, X. Peng, Y. Yu, J. Mylopoulos, and W. Zhao, "Stateful requirements monitoring for self-repairing socio-technical systems," in *20th IEEE International Requirements Engineering Conference*, 2012, pp. 121–130.

[3] B. Chen, X. Peng, Y. Yu, and W. Zhao, "Are your sites down? Requirements-driven self-tuning for the survivability of Web systems", in *RE*, 2011, pp. 219–228.

Runtime Traceability for an Adaptive Time Machine

Yijun Yu
Department of Computing and Communications
The Open University
United Kingdom

Time travel is the dream of archaeologists to verify their hypotheses of our history, also the subject of science fiction writers to propose their speculations of our destiny. Featured by change management and backup file systems, even imperfect, travelling to the past of a project is no longer a dream.

However, travelling to the future software world remains largely so. On basis of various advances in the theory and practice of requirements engineering [3, 6, 5], especially the progress in the active maintenance of runtime traceability relationships [4, 1] between problems and solutions, we speculate that travelling to the future may be enabled by (a collection of) carefully designed software systems [2].

References

[1] Y. Yu, Y. Lin, Z. Hu, S. Hidaka, H. Kato, and L. Montrieux, "Maintaining invariant traceability through bidirectional transformations", in *ICSE*, 2012, pp. 540–550.

[2] M. Wermelinger, Y. Yu, A. Lozano, and A. Capiluppi, "Assessing architectural evolution: a case study", *Empirical Software Engineering*, vol. 16, no. 5, pp. 623–666, 2011.

[3] M. Salifu, Y. Yu, A. K. Bandara, and B. Nuseibeh, "Analysing monitoring and switching problems for adaptive systems", *Journal of Systems and Software*, vol. 85, no. 12, pp. 2829–2839, 2012.

[4] Y. Yu, T. Tun, and B. Nuseibeh, "Specifying and detecting meaningful changes in programs", in *ASE*, 2011, pp. 273–282.

[5] Y. Yu, Y. Wang, J. Mylopoulos, S. Liaskos, A. Lapouchnian, and J.C.S.P. Leite, "Reverse Engineering Goal Models from Legacy Code," in *RE*, 2005, pp. 363–372.

[6] Y. Wang, S. A. McIlraith, Y. Yu, and J. Mylopoulos, "An automated approach to monitoring and diagnosing requirements," *Automated Software Engineering Journal*, vol. 16, no. 1, pp. 3–35, 2009.

Can bidirectional transformation be used for implementing Adaptive Software Systems?

Zhenjiang Hu
National Institute of Informatics
Japan

Bidirectional transformations [1, 2] provide a novel change propagation mechanism for synchronizing and maintaining the consistency of information between input and output, while adaptive software systems are able to adapt to changes that may occur in the system, its requirements, or the environment in which it is deployed. However, the relationship between bidirectional transformations and adaptive software systems are unclear.

In this talk, after reformulating the concepts of component-based systems, system scopes, and contexts, we explain how changes on components can be globally propagated based on the predefined local change propagation policies assigned to the components, and show how bidirectional transformations would play a key role in this global change propagation. It would be interesting in the future to investigate whether the local change propagation policies and the local feedback loop of bidirectional transformations could provide yet another implementation of adaptive software systems rather than using the existing global MAPE loop.

References

[1] K. Czarnecki, J.N. Foster, Z. Hu, R. Lammel, A. Schurr, J.F. Terwilliger, "Bidirectional Transformations: A Cross-Discipline Perspective", *International Conference on Model Transformation (ICMT 2009)*, ETH Zurich, Switzerland, June 29-July 3 2009. LNCS 5563, Springer. pp. 260–283.

[2] Z. Hu, A. Schurr, P. Stevens, J. Terwilliger, "Dagstuhl Seminar on Bidirectional Transformations", *SIGMOD Record*, Vol.40, No.1, 2011. pp. 35–39.