ISSN 2186-7437

NII Shonan Meeting Report

No. 2015-11

Engineering Adaptive Software Systems (EASSy) NII Shonan Meeting Report

Tetsuo Tamai
Hausi Muller
Bashar Nuseibeh

September 7–10, 2015

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

Engineering Adaptive Software Systems (EASSy)
NII Shonan Meeting Report

Organizers:
Tetsuo Tamai (Hosei University)
Hausi Muller (University of Victoria)
Bashar Nuseibeh (Open University, Lero)

September 7–10, 2015

It goes without saying that the modern society is founded on the information infrastructure, which is basically constructed with software, but its sustainability is now in question. Information systems will become obsolete and unusable if they cannot respond to requirements changes demanded by the users and their reliability and safety will be severely degraded if they cannot cope with their environment changes. The key issue here is how to engineer adaptive systems that conquer such sustainability threats.

This workshop, the third edition of its series, focused on this issue of engineering adaptive systems. The technical goals were:

Adaptation to environment changes Software systems are deployed or embedded in various environments and those environments frequently change. Software, in spite of its name, often shows stiffness and inflexibility to changes. How to provide adaptability to software is an important and challenging goal.

Quality requirements When software obtains the property of adaptability, quality requirements, including those for dependability and usability, may be affected. It is crucial to monitor the current quality requirements satisfaction status and maintain the qualities at the required level.

Specifically, topics discussed at the workshop included the following.

- 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? .
- How do we reengineer legacy software systems in order to turn them into adaptive ones?
- Comparative review of adaptation mechanisms in Robotics, Multi-Agent Systems, Software Engineering, Socio-Technical Systems, Ubiquitous Computing, etc.

- Usability issues for adaptive software systems. How do we ensure effective human interaction with complex software systems that have adaptive components?
- Evolution of adaptive software systems. How do deployed adaptive systems evolve? How can we ensure convergence and stability for such a system, particularly when it is a system-of-systems composed of component systems, each with its own requirements and own adaptation mechanism?
- How do we reason with runtime models to support adaptation functions, i.e., monitoring, diagnosis and compensation? How can we support incremental runtime reasoning that predicts and/or prevents failures?

Meeting Schedule

Arrival Day – Sun, Sep 6

19:00 - 21:00 – Welcome Reception

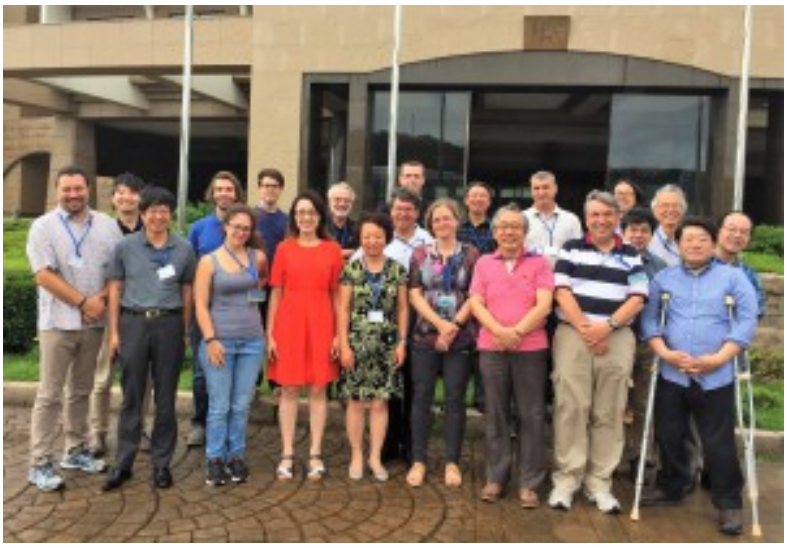

Figure 1: Shonan-No52-Group-Photo

Day 1 – Mon, Sep 7

7:30 – 8:30 – Breakfast

8:30 – 9:00 – Session 1 – Chair: Tetsuo Tamai

- Welcome and Overview of Seminar No. 052
  Tetsuo Tamai (Organizer), Hosei University, Japan
- National Institute of Informatics (NII)
  Shinichi Honiden, National Institute of Informatics, Japan

- Shonan Meetings – Video
  Zhenjiang Hu, National Institute of Informatics, Japan

9:00 – 10:30 – Session 2 – Chair: Hausi Muller

- Adaptation and Boundaries
  Bashar Nuseibeh (Organizer), Lero and The Open University, UK
- Engineering Adaptive Software Systems: A Research Agenda
  John Mylopoulos, University of Trento, Italy –
- Discussion
  Recorder: Lionel Montrieux, NII, Japan
  Recorder: Martina Maggio, Lund University, Sweden

10:30 – 11:00 – Break

11:00 – 12:00 – Session 3 – Chair: Tetsuo Tamai

- Dynamic Software Evolution – Approaches and Issues
  Shinichi Honiden, National Institute of Informatics, Japan
- Towards Effective Management of Dynamic Software Evolution
  Yasuyuki Tahara, The University of Electro-Communications, Japan
- Discussion
  Recorder: Clement Quinton, Politecnico di Milano, Italy
  Recorder: Martina Maggio, Lund University, Sweden

12:00 – 13:30 – Lunch

13:30 – 15:00 – Session 4 – Chair: Shinichi Honiden

- How to Capture Context and Context-dependent Behavior
  Tetsuo Tamai (Organizer), Hosei University, Japan
- Contexts and Unit of Adaptation in Context-oriented Programming
  Hidehiko Masuhara, Tokyo Institute of Technology, Japan
- Context-Oriented Programming for Adaptive Software Systems
  Tetsuo Kamina, Ritsumeikan University, Japan
- Discussion Recorder: Martina Maggio, Lund University, Sweden

15:00 – 15:30 – Break

15:30 – 17:00 – Session 5 – Chair: John Mylopoulos

- Dynamic Software Composition for Run-time System Evolution
  Robert Hirschfeld, Hasso Plattner Institute at the University of Potsdam, Germany
- High Variability Models for Better Adaptive Systems
  Kostas Angelopoulos, University of Trento, Italy
- Discussion
  Recorder: Amel Bennaceur, The Open University, UK

Figure 2: YijunYu Challenge

18:00 – 19:30 – Dinner

19:30 – 21:30 – Ping Pong Tournament

Day 2 – Tue, Sep 8

7:30 – 8:45 – Breakfast

9:00 – 10:30 – Session 6 – Chair: Marin Litoiu

- Integrated Control and Systems Science for Cyber-Physical Systems: A Research Agenda
  Hausi A. Muller (Organizer), University of Victoria, Canada
- Next Generation Collaborative Distributed Visualization Systems on the Distributed Cloud
  Rick McGeer, SAP, San Francisco, USA
- Discussion
  Recorder: Kostas Angelopoulos, University of Trento, Italy

10:30 – 11:00 – Break

11:00 – 12:00 – Session 7 – Chair: Rick McGeer

- Guaranteeing Solution Quality for SAS Optimization Problems by Being Greedy
  Ulrike Stege, University of Victoria, Canada
- Discussion
  Recorder: Yijun Yu, The Open University, UK

12:00 – 13:30 – Lunch

13:30 – 15:00 – Session 8 – Chair: Hausi Muller

- Towards Robust Linear Quadratic Control of Software Systems Marin Litoiu, York University, Canada

- Discrete Time Adaptive Linear Control for Software Systems
  Martina Maggio, Lund University, Sweden
- Assured Graceful Degradation with Discrete Controller Synthesis
  Kenji Tei, National Institute of Informatics, Japan
- Discussion
  Recorder: Nobukazu Yoshioka, NII, Japan

15:00 – 15:30 – Break

15:30 – 17:00 – Session 9 – Chair: Yijun Yu

- A View-based Approach to Software Adaptation
  Zhenjiang Hu, National Institute of Informatics, Japan
- Bidirectional Programming for Self-adaptive System
  Lionel Montrieux, National Institute of Informatics, Japan
- Evolving Dynamic Software Product Lines
  Clement Quinton, Politecnico di Milano, Italy
- Discussion
  Recorder: Amel Bennaceur, The Open University, UK

18:00 – 19:30 – Dinner

19:30 – 21:15 – Sing-along

Figure 3: Sing Along

Day 3 – Wed, Sep 9

7:30 - 8:45 – Breakfast

9:00 – 10:30 – Session 10 – Chair: Zhenjiang Hu

- The Aftermath of Mystery Flight MH370: What Can Adaptive Software Engineers Do?
  Yijun Yu, The Open University, UK –

- Software Self-Adaptivity Measurement based on Requirements Models
  Zhi Jin, Peking University, China –
- Modularity for Uncertainties in Adaptive Software Systems
  Naoyasu Ubayashi, Kyushu University, Japan
- Discussion
  Recorder: Kenji Tei, NII, Japan

10:30 – 11:00 – Break

11:00 – 12:00 – Session 11 – Chair: Bashar Nuseibeh

- Requirements-Driven Mediation for Collaborative Security
  Amel Bennaceur, The Open University, UK
- An Adaptive Framework for Individual Privacy
  Nobukazu Yoshioka, NII, Japan
- Discussion
  Recorder: Lionel Montrieux, NII, Japan

12:00 – 13:30 – Lunch

13:30 – 19:00 – Excursion canceled due to Typhoon No.18

19:00 – 20:45 – Banquet in Kamakura

Figure 4: Banquet in Kamakura

Day 4 – Thu, Sep 10

7:30 – 8:45 – Breakfast

9:00 – 10:30 – Session 12

- Breakout Session 1: Context Oriented Programming with MAPE – Robert Hirschfeld
- Breakout Session 2: Bidirectional Programming – Lionel Montrieux

- Breakout Session 3: Control Theory and Optimization – Martina Maggio and Ulrike Stege
- Breakout Session 4: Security and Privacy – Amel Bennaceur

10:30 – 11:00 – Break

11:00 – 12:00 – Session 13

- Report from Breakout Session 1
  Chair: Robert Hirschfeld, Hasso Plattner Institute at the University of Potsdam, Germany
- Report from Breakout Session 2
  Chair: Lionel Montrieux, NII, Japan
- Report from Breakout Session 3
  Co-chairs: Martina Maggio, Lund University, Sweden and Ulrike Stege, University of Victoria, Canada
- Report from Breakout Session 4: SAS Security and Privacy
  Chair: Amel Bennaceur, The Open University, UK

12:00 – 13:00 – Lunch

13:00 – Departure

Fri/Sat, Sep 4-5, 2015

We climbed Mt. Fuji (Fuji-san) with perfect weather.


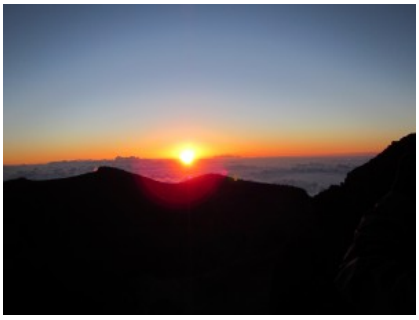

Figure 5: Mt. Fuji Climbing

Overview of Talks

Engineering Adaptive Software Systems: A Research Agenda

John Mylopoulos, University of Trento, Italy and University of Toronto, Canada

Abstract — Adaptive software systems need to be capable of multiple behaviours for fulfilling their requirements, so that if one behaviour is failing for whatever reasons, the system can switch (“ reconfigure ”) to an alternative behaviour. The most important question then in engineering such systems is: how do we design and implement systems that are capable of fulfilling their requirements in multiple ways? What are the concepts in terms of which such systems are conceived, designed and implemented? What techniques do we use to analyze and design them? What are the adaptation mechanisms through which an adaptive system monitors its behaviour, determines root causes for failing requirements and selects a suitable adaptation?

The presentation will review answers we have given to some of these questions in the PhD theses of Vitor Souza and Kostas Angelopoulos (on-going), also some of the open questions that constitute our current research agenda.

How to Capture Context and Context-dependent Behavior

Tetsuo Tamai, Hosei University, Japan

Abstract — Adaptive systems change their behavior dependent on context changes. The key issue in modeling and developing such systems is how to capture context and context-dependent behavior. Factors that determine context may include location, time, interacting agents and technical environment. In this talk, I'd like to discuss ways of determining appropriate contexts from the requirements engineering point of view.

Adaptation and Boundaries

Bashar Nuseibeh, Lero and The Open University, UK

Abstract — In this speculative talk, I will explore the role of boundaries in engineering adaptive software systems, and will revisit claims that they are ‘disappearing’. I will use this exploration to inform the development of a research agenda in the area of engineering adaptive software.

Integrated Control and Systems Science for Cyber-Physical Systems: A Research Agenda

Hausi A. Muller, University of Victoria, Canada

Abstract — Cyber-physical systems (CPS) are smart systems that encompass computational and physical components, seamlessly integrated and closely interacting to sense the context of the real world. The societal impact of CPS is enormous. Virtually every engineered system is affected by advances in these interconnected capabilities. Today CPS R&D affords transformative opportunities due to the convergence of analytical and cognitive capabilities, real-time

and networked control, pervasive sensing and actuating, as well as compute and storage clouds. Advancement in CPS requires an integrated control and systems science (CSS) that encompasses both physical and computational aspects. Engineering and computer science researchers have provided impressive advances in CPS foundations?control, communications and computing, in general, and embedded, real-time, self-adaptive, and autonomic systems, in particular. We now need to address the unique scientific and technical challenges for this new type of integrated CSS for CPS.

The Aftermath of Mystery Flight MH370: What Can Adaptive Software Engineers Do?

Yijun Yu, The Open University, UK

Abstract — The aftermath of the missing MH370 flight a year ago remains still a mystery: no one knows firmly where the crash was and what caused the problem. In order to answer these questions, worldwide search has been carried out ever since to locate first hand evidence in on-board flight data recorders (also known as blackboxes). To enhance aviation security, a proposal was using cloud computing to analyse live streamed flight data. This talk elaborates this proposal from an adaptive software engineering perspective.

Dynamic Software Evolution ? Approaches and Issues

Shinichi Honiden, National Institute of Informatics and The University of Tokyo, Japan

Abstract — In my talk, I will mention some approaches to enable dynamic software evolution. In particular, the reflection technique is a promising approach because it enables the software to change itself dynamically. I will also talk about some issues to apply these approaches.

Discrete Time Adaptive Linear Control for Software Systems

Martina Maggio, Lund University, Sweden

Abstract — Modern software systems are complex entities, that should guarantee their behavior satisfy a certain number of requirements and constraints, during their execution. Control theory has been identified as one of the possible design drivers for runtime adaptation, but the adoption of this discipline's principles often requires additional knowledge to be processed by a specialist. To overcome this limitation, automated methodologies have been proposed, that try to extract the necessary information from experimental data and design a control system for runtime adaptation. In this talk I will overview the research journey from the early adoption of ad hoc linear control systems to these automated methodologies and present some of the results that we obtained with different problems, from clock synchronization to cloud performance predictability.

Assured Graceful Degradation with Discrete Controller Synthesis

Kenji Tei, NII, Japan

Abstract — System will face unexpected consequence in its operating environment. In such a situation, the system should degrade gracefully to avoid catastrophic situation. Research questions I address here are how does the system degrade gracefully at runtime with assurance and how does the system determine how much it should degrade. My talk will show a framework enabling guaranteed graceful degradation with discrete controller synthesis, and techniques to select functionality level that system can satisfy in the current perception of the environment, and to which the system can seamlessly degrade.

Software Self-Adaptivity Measurement based on Requirements Models

Zhi Jin, Peking University, China

Abstract — Self-adaptivity is currently becoming a more and more important property of software which will run in an open and dynamically changing environment. How do we know if the to-be-built software system will possess the necessary capability of adjusting its behavior for responding the changes in environment? The measurement can be conducted as earlier at the system modeling stage. This talk is trying to deliver some thinking about modeling of the self-adaptive software systems and measuring of the software self-adaptivity based on requirements models.

Contexts and Unit of Adaptation in Context-oriented Programming

Hidehiko Masuhara, Tokyo Institute of Technology, Japan

Abstract — Context-oriented programming (COP) is proposed for modularizing dynamically-changing, context-dependent behaviours. A motivation behind COP is that many modules in a program tend to have fragments of descriptions that are specific to a specific context. The approach in COP is providing a new abstraction that is specific to a context, yet affects many modules. While COP has been successful as a programming mechanism, the notion of a context and the unit of adaptations in COP are not yet clear. For example in some COP languages, contexts are merely “callers” of a module, while in some others, contexts exist outside of a running program, whose changes are observed as events in a program. In this talk, we discuss several required properties of contexts and unit of adaptation, and possible language designs to generalize those notions.

Dynamic Software Composition for Run-time System Evolution

Robert Hirschfeld, Hasso Plattner Institute at the University of Potsdam, Germany

Abstract — The longer systems run, the more likely they will need to be revised to keep up with the changes in their environment ranging from user expectations over technological advances to mistakes made. Since many such situations are unanticipated, planning for them in advance is often impossible. While computational reflection as such allows for changing running systems at the language level, the mechanisms provided are often very primitive and too general to be applied comfortably. Context-oriented programming, or COP for short, offers modularity constructs and composition mechanisms that allow for run-time adaptation and evolution at a higher level of abstraction in a more structured way. We will give an introduction to COP, present recent developments of COP language and infrastructure support in our group, and hope to learn about and discuss new and interesting application areas and scenarios to derive novel research questions.

Requirements-Driven Mediation for Collaborative Security

Amel Bennaceur, The Open University, UK

Abstract — Collaborative security exploits the capabilities of the components available in the ubiquitous computing environment in order to protect assets from intentional harm. By dynamically composing the capabilities of multiple components, collaborative security implements the security controls by which requirements are satisfied. However, this dynamic composition is often hampered by the heterogeneity of the components available in the environment and the diversity of their behaviours.

In this talk I will present a systematic, tool-supported approach for collaborative security based on a combination of feature modelling and mediator synthesis. This approach ensures that the implemented security controls are the optimal ones given the capabilities available in the operating environment. I will show how we used the FICS (Feature-driven Mediation for Collaborative Security) tool to make two robots?a humanoid robot and a vacuum cleaner?collaborate in order to implement an additional security control for protecting a mobile phone from theft.

Towards Robust Linear Quadratic Control of Software Systems

Marin Litoiu, York University, Canada

Abstract — Adaptive software systems cope with changes in environment by self-adjusting their structure and the behaviour. Robustness refers to the ability of the system to deal with uncertainty, that is parameter perturbations or not-modeled system dynamics that can affect the quality of the adaptation. In this presentation we show a formal method to design and implement a model

identification adaptive controller (MIAC) using a combination of performance and control models. The controller optimizes a linear quadratic objective function. We show preliminary results on a cloud-deployed application and show that the controller performs well for a wide range of perturbations.

Evolving Dynamic Software Product Lines

Clement Quinton, Politecnico di Milano, Italy

Abstract — In many domains, systems need to run continuously and cannot be shut down for reconfiguration or maintenance tasks. Cyber-physical or cloud-based systems, for instance, thus often provide means to support their adaptation at runtime. The required flexibility and adaptability of systems suggests the application of Software Product Line (SPL) principles to manage their variability and to support their reconfiguration. Specifically, Dynamic Software Product Lines (DSPL) have been proposed to support the management and binding of variability at runtime. While SPL evolution has been widely studied, it has so far not been investigated in detail in a DSPL context. Variability models that are used in a DSPL have to co-evolve and be kept consistent with the systems they represent to support reconfiguration even after changes to the systems at runtime. In this presentation we describe the consequences of such changes on the consistency of the DSPL and analyze their impact on the running system.

Guaranteeing Solution Quality for SAS Optimization Problems by Being Greedy

Ulrike Stege, University of Victoria, Canada

Abstract — When dealing with self-adaptive systems one is regularly tasked with solving optimization problems. A frequent strategy to solve such a problem is the greedy approach. Unfortunately, often no quality guarantees are known for a specific greedy algorithm solution-specific solutions are often validated and compared using simulations. In this talk we discuss a mathematical framework to investigate the quality of greedy approaches for maximization problems systematically and, thereby, often eliminating the need for extensive simulation runs. Furthermore, we discuss methods to tweak the problem at hand to improve the solution quality obtained using the greedy approach.

Bidirectional Programming for Self-adaptive Systems

Lionel Montrieux, NII, Japan

Abstract — A bidirectional transformation is a pair of functions, ‘get’and ‘put’, allowing developers to keep two documents, a source and a view, synchronised. ‘get’takes a source and produces a view, whilst ‘put’takes a source as well as an updated view, and reflects the changes made to the view into the source. Bidirectional programming uses Domain-Specific Languages (DSLs) to facilitate writing bidirectional transformations, while ensuring that the transformations satisfy some important properties. This talk will explore our use of bidirectional

programming in self-adaptive systems. In particular, we will discuss the use of bidirectional transformations to keep the knowledge base synchronised with the monitored system, and to extract and manipulate parts of the knowledge base into smaller, more manageable models.

High Variability Models for Better Adaptive Systems

Kostas Angelopoulos, University of Trento, Italy

Abstract — Variability is essential for adaptive systems, because it capture the solution space where the alternative adaptations a system can perform, when it adapts. In our work we investigate a) what types of variability are present in software systems and their environment, b) what models are suitable for eliciting all the variables that affect the performance (wrt requirements fulfillment) of the system-at-hand and c) how these models are related with each other. Finally, we examine methods from Control-Theory in order to efficiently handle the elicited variables and maintain a stable and optimal satisfaction of the prescribed requirements.

Next Generation Collaborative Distributed Visualization Systems on the Distributed Cloud

Rick McGeer, SAP, San Francisco, USA

Abstract — We describe the Distributed Collaborative Scientific Visualization System, a system designed to permit real-time interaction and visual collaboration around large data sets, with an initial emphasis on scientific data. The Visualization System offers such a collaborative environment, with real-time interaction on any device between users separated across the wide area. The Visualization System provides seamless interaction and immediate updates even under heavy load and when users are widely separated: the design goal was to fetch a data set consisting of 30,000 points from a server and render it within 150 milliseconds, for a user anywhere in the world, and reflect changes made by a user in one location to all other users within a bound provided by network latency. The system was demonstrated successfully on a significant worldwide air pollution data set, with pollution values on a 10 km, 25 km, 50 km, and 100 km worldwide grid, with monthly values over an 18-year period. It was demonstrated on a wide variety of clients, including laptop, tablet, and smartphone.

Towards Effective Management of Dynamic Software Evolution

Yasuyuki Tahara, The University of Electro-Communications, Japan

Abstract — Recently, there are arising several research topics dealing with dynamic software evolution, such as self-* systems, including self-adaptive ones, autonomic computing, models at run time, and requirements at run time. Because the behaviors of dynamically evolving software tend to become complicated, it is more difficult to manage those behaviors effectively than conventional

software. In my talk, I will examine various existing approaches, including formal verification, to management of dynamic software evolution and their problems, and will suggest some research directions to solve the problems.

An Adaptive Framework for Individual Privacy

Nobukazu Yoshioka, NII, Japan

Abstract — Privacy is a right of users to control their private information. In other words, users can decide to reveal their private information to others. Services with users' private information should be provided with respect to privacy. Some frameworks to preserve the right have been proposed. It, however, still hard for developers to meet users' privacy preferences and develop a privacy friendly service.

In my talk, we propose a new framework to address these problems. The framework allows users to choose their privacy preferences from the viewpoints of both the privacy risk and the value of services. In addition, developers can design the behavior with the variation of a service, so that the development costs are reduced. Furthermore, our framework can adapt to the changes of a preference automatically. We illustrate the effectiveness with a case study of an exercise service.

Context-Oriented Programming for Adaptive Software Systems

Tetsuo Kamina, Ritsumeikan University, Japan

Abstract — Behavior adapted to the system at runtime in response to environment changes often crosscuts several parts of the system. Context-oriented programming (COP) is an emerging programming paradigm to modularize such behavior adaptation. In this work, we identify the challenges in developing COP languages and what we have achieved through the development of our COP language, ServalCJ. We then propose a software development methodology based on COP. In particular, we develop a systematic way to find an appropriate linguistic mechanism to implement context-dependent behavior and dynamic adaptations of it at modeling and design time, leading to the mechanized mapping from requirements and design artifacts formed by our methodology to the COP implementation. Through case studies, this mapping is demonstrated using ServalCJ. Moreover, we discuss the applicability of COP to the development of adaptive software systems.

A View-based Approach to Software Adaptation

Zhenjiang Hu, NII, Japan

Abstract — I will talk about a new approach to structuring adaptation rules so that the rule set can be reconstructed dynamically for different purposes and goals. The key idea is to embed a local invariant view to each rule so that the global view of a desired adaptation logic can be realized safely by combining rules with relevant local views. Our new view-based adaptation framework

combines the strengths of the rule-based and goal-based adaptation approaches, and enjoys the advantages of both. This is joint work with Tianqi Zhao, Haiyan Zhao, and Zhi Jin from Peking University, and Tao Zan from NII.

Modularity for Uncertainties in Adaptive Software Systems

Naoyasu Ubayashi, Kyushu University, Japan

Abstract — Embracing uncertainties in software evolution is one of the crucial research topics in engineering adaptive software systems. As the research on uncertainty is so young, there are many issues to be tackled. Modularity for uncertainties is one of them. If uncertainties can be dealt with modularly, we can add or delete uncertain concerns to/from a software system whenever these concerns arise or are fixed to certain concerns. To deal with this challenging issue, we propose a modularization mechanism for uncertainties. Agile methods embrace changes to accept changeable user requirements. On the other hand, our approach embraces uncertainties to support exploratory software evolution.

Session Report

Session 2 – Chair: Hausi Muller

Note Recorded by Martina Maggio, Lund University, Sweden

- Bashar Nuseibeh – Adaptation and Boundaries

  Ubiquitous and mobile systems: depends a lot on the context in which they are placed and used. Seamless: it hides the boundaries between two things, it makes information flow across boundaries. Mention of security and privacy of systems and information flow.

  Sometimes boundaries are a feature, sometimes they are a constraint. Considering these boundaries explicitly may be useful. Difficulty is that sometimes boundaries are unclear and changing. So we need to think about them both at design time and at runtime. 20 years ago the boundary between software and hardware was very clear. Now it is not so clear anymore and some things are implemented in hardware and software, example given is smart cities.

  In the PhD spent time trying to understand how each person can specify partial knowledge about these boundaries. Every person has different perspective, the interesting part is how one perspective relates to the others.

  - Boundary critique (Ulrich 2002, following Churchman 1970)
  - The disappearing boundary between development time and runtime (Baresi & Ghezzi 2010)
  - Interaction design: making and using (Nakakoji 2011)

  The boundaries are still there: tacit or explicitly. Security is also all about boundaries. Security and safety: a bird getting in an airplane engine makes it explode. Someone is throwing the bird. The first is a safety problem, the second is both a safety and a security problem. The engine should be bird resistant (safety) but also nobody should enter the perimeter and be able to throw the bird (security). Trust assumptions are important.

  Interesting thing is not disappearing boundaries but changing boundaries. Boundaries between mobile devices, infrastructure and people are difficult to identify and manage. Trust assumptions that we were using to bound security problems no longer hold. Adaptive security challenge is to try to understand and cope with those changes.

  Notion of topology of context, structure of the context in terms of the operational environment. Not necessarily physical, but also for example due to the network and the connections. You have a number of layers of this topology. Maps that are partly physical and partly virtual (Pasquale, SEAMS 2012, Tsigkanos RE 2014
  http://lili-pasquale.lero.ie/papers/RE2014.pdf
  and ICSE 2015
  http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7203054).

  How can someone cross a digital boundary? Interplay between physical and digital gives better analysis.

- John Mylopoulos – Engineering Adaptive Software System: a research agenda

  Concepts, tools and techniques for building an artifact – in this case an adaptive software system.

  What is special about adaptive software system and why cannot we use classical SE techniques? They are special in the sense that they have a distinctive architecture that separates concern between the base system and the adaptation mechanism. This adaptation functions (monitoring, analyzing, planning and execution) are separated from the main components.

  Some class of requirements may lead to feedback loops. Where did this MAPE function come from and what are the requirements about those four components? What are the additional requirements that lead to the feedback loop?

  - Awareness requirements (Souza 2011)
    This requirement R will not fail more than three times per month.
  - Evolution requirements (Souza 2012)
    If R fails more than three times per month, change it to R-.
  - Adaptation requirements (Angelopoulos 2014)
    Adaptation should not affect non-failing requirements.
  - Contextual requirements (Ali 2010)
    R is a requirement only in context C.

  We need to represent runtime requirements and their state and other information about them. You have to take design time models and you have to extend them. At runtime, you have to create instances of these models, you have to keep history (for example to deal with awareness requirements and the "it will not fail more than a certain number of times") to deal with these requirements (Dalpiaz 2013). Fuzzy runtime requirements models (Baresi 2010). Runtime requirements model for reflection (Bencomo 2010).

  We need to be able to find the root causes of failures, for example using AI diagnostic techniques (Wang 2007).

  System with large adaptation spaces. How large an adaptation space can we support. We are interested in making it as big as possible. A lot of the literature uses a requirement model to define an adaptation space (Souza 2011). The alternative is looking at the adaptation space as the architecture and see in how many possible ways you can change the architecture.

  Lots of feedback loops in the system (each one of them introduced by a requirement like an awareness requirement or similar). Adaptation can cancel out each other. Somehow we must be able to deal with concurrent failures and deal with aggregated answers. In Rainbow (Garlan 2006), you can have rules where you can have "if R1 and R2 are both failing do A". Enumeration of rules (they overwrite other rules). Another way to deal with that is qualitative reasoning (Angelopoulos 2014) . "if an

adaptation A for problem P interferes with another problem P' don' t use it).

Optimization requirements . “minimize meeting cost while meeting some requirements”. Combination of a minimization requirement and the constraint that we have to restore another requirement (failing for example because the number of meeting rooms is not enough). The check of requirements consistency is usually done with a SAT solver but it is difficult to combine this with the minimization one. Use latest results from the Automated Reasoning community, maybe Sebastiani 2015.

Full adaptation: can we allow requirements to evolve due to failures in ways unanticipated at design time? A very preliminary answer proposes to use AI planning and domain knowledge to propose new requirements at runtime (Sabatucci, SEAMS 2015).

Note Recorded by Lionel Montrieux, NII, Japan

- Bashar Nuseibeh – Adaptation and Boundaries

  Bashar starts by warning us that most of his talk is speculation, no results, though he has some evidence to support his claims. He hopes to trigger discussions.

  Bashar focuses on boundaries, especially in ubiquitous and mobile systems. These systems should be able to reconfigure themselves, and adapt to changing contexts, but somehow users should not notice. Bashar argues that this seamlessness in fact hides boundaries between systems and components. These boundaries change, and information flows across them.

  Boundaries are prevalent in software engineering, and it is very instructive to think of boundaries and the flow information across them. Bashar argues that boundaries in SE (e.g. boundaries between design and run time) are not disappearing, but are often tacit. It is important to identify those tacit boundaries.

  In requirements engineering, he echoes Michael Jackson's argument that the essence of requirements engineering is to find the problem's boundaries.

  Security is all about boundaries. Get them wrong, and your system is not secure. Trust assumptions are the raw material of boundaries, as they affect what designers believe to be true at a certain point. These trust assumptions change over time, and so do the boundaries that follow from them.

  Bashar concludes by talking about topological boundaries, and the use of topology to describe the structure of the operational environment of a system, including its physical, digital, and/or social aspects.

- John Mylopoulos – Engineering Adaptive Software Systems

  John had prepared 10 questions, on the tools, techniques and concepts for building adaptive software systems (ADSSs). Due to time constraints, he had to skip a few of them.

  What makes ADSSs special, is their architecture that separates the base system from the adaptation system.

1. What requirements lead to a MAPE?
   4 types of requirements need a MAPE look: awareness, evolution, adaptation and contextual requirements. Tools and techniques are required to deal with each of those types of requirements.
2. What do runtime requirements look like?
   The literature includes hierarchies of goals, fuzzy runtime requirement models, or runtime requirement models for reflection. The critical issue is that monitoring becomes non-scalable and/or intractable in the modelling language is over-expressive. The description of the system's behaviour, in particular, is very important.
3. What failures trigger adaptation? [skipped]
4. Diagnosing the problem
   Failure is often a symptom of a problem, not the problem itself. Root cause analysis is necessary to understand what happened, and hopefully prevent it from happening again. Some answers may exist in the AI diagnostic community. Solutions are often difficult to scale, but some seem to work well in practice.
5. Systems with large adaptation spaces
   How big an adaptation space does a system support? What's the space of all possible adaptations?
6. Dealing with multiple failures
   An adaptation that deals with one problem may interfere with adaptations that deal with other problems. Some work in this area includes the use of rules, or quantitative reasoning. Rules tend not to scale well. It is necessary to give one coherent answer to several problems, rather than several answers, each solving a single problem, but not necessarily compatible with each other.
7. When can you reconfigure the system?
   [skipped]
8. Optimisations for adaptation requirements
   In general, there may be many adaptations satisfying all the adaptive system's constraints. How to choose the best one is an optimisation problem, which requires both SAT-based and optimisation-based reasoning. This is difficult.
9. The identification problem for ADSSs
   Finding the relationship between input and output is a well-known problem in control theory. In our case, we look at requirements and failures.
   Solutions in the literature include:
   - guestimate qualitative differential relations;
   - case-based reasoning;
   - learn over time.
10. Full adaptation
    See Jeff Kramer's rather excellent SEAMS'15 keynote. Can we adapt to failures in ways unanticipated at design time? Some have proposed the use of AI planning and domain knowledge to propose new requirements at runtime.

Note Recorded by Clement Quinton, Politecnico di Milano, Italy

- Adaptation and Boundaries – Bashar Nuseibeh (Organizer), Lero and The Open University, UK

  Interesting thing about adaptive systems: meant to be seamless. Side effect: it hides the boundaries between the different components.

  What is a boundary? There is almost always one: a feature, a constraint... Those boundaries are unclear and changing, we have to manage them, both at design time and runtime. In RE, it's all about bounding: defining the problem boundaries (scope).

  Are boundaries disappearing? (e.g. between design and runtime) No they're not, they're tacit or explicit. Boundaries are not disappearing but changing, so are trust assumptions.

Session 3 – Chair: Tetsuo Tamai

Note Recorded by Martina Maggio, Lund University, Sweden

- Shinichi Honiden – Dynamic Software Evolution - Approaches and Issues

  Software evolution implies that the software adapts to requirement changes. Facebook system is updated every day during ordinary operation. Motivating example: online shopping system. The first evolution is adding identification function with id and password and the second evolution is adding two-factor authentication functions. After the first evolution in the goal model there are added parts and also for the sequence diagrams there is something added.

  How to implement dynamic evolution? Use of Javassist, a class library to provide functional reflection. Dynamic evolution uses reflection, program can rewrite themselves at runtime. Class can be replaced to change things at runtime. Reflection is the only technique that allows a program to change itself. In terms of the location of changes, reflection is the only technique that can change things everywhere in the program.

  Continuous delivery. One should think about before evolution, after evolution and during evolution. Many users, which should seamlessly execute despite the evolution. What happens to the users that are logging in when the two factor authentication process is added.

  How to express the behavior of the specification of the dynamic evolution? Model checking. Model checking would be promising to verify the evolution behavior. Maude: algebraic specification language which supports reflection and model checking. But this does not scale.

- Yasayuki Tahara – Towards Effective Management of Dynamic Software Evolution

  Case study with the shopping website is an adaptation of (Chen et al. 2014, Qian et al. 2014). as contexts, existing data structures).

Note Recorded by Clement Quinton, Politecnico di Milano, Italy

- Dynamic Software Evolution . Approaches and Issues – Shinichi Honiden, National Institute of Informatics, Japan

  Software evolution is an activity to adapt to requirements changes. Motivating example with the evolution of a online shopping system, by adding a security form.

  Problematic: how to implement dynamic evolution in the software?

  ⇒ Using reflection with Javassist. A new class is created at runtime. Reflection is the only way to change the program in details. Such an evolution must be done without interrupting the system. By using model checking, evolution behaviors are verified.

  ⇒ Maude supports reflection and model checking.

  Limitations in terms of time to solve, e.g. 24h for 3 users in the motivating example.

  Description of the different sequence diagrams, before and after evolution. Technical slides on Maude, how to use meta-level representations to model reflection

Session 4 – Chair: Shinichi Honiden

Note Recorded by Martina Maggio, Lund University, Sweden

- Tetsuo Tamai – How to capture context and context-dependent behavior

  Major results of Kumiki project in component and composition design are the collaboration model Epsilon (Tamai et al.) and the aspect oriented model (Masuhara et al.). In formal verification variant parametric type system (Igarashi) and work model checking for component based systems. Kumiki 2 continued the work with aspect oriented programming with S. Chiba. Then we included the notion of context.

  Context awareness is necessary for adaptation. Example problems are conference guide systems, program editor and robot simulation. COP based on the concept of layers and the context is determined by many things like the results of sensors, location. These things define the environment of operation.

- Hidehiko Masuhara – What do self-adaptive systems adapt to?

  COP (Hirschfeld 2008) tries do modularize context-dependent behavior by providing language mechanisms. If we want programs to be modular context is useful. Otherwise we have a lot of if branches. Example adventure game: hero in normal state, hero in drunk state (whenever you try to move the hero, he moves randomly). Another example is energy aware mobile applications (Cohen 2012) where the context can be the device status (plugged/unplugged) or the network status and the behavior of the application can be different (quality of the rendering, data saving frequency and so on).

  Context: surrounding of an adaptation unit. Research is definition of structured context. Uniform language mechanisms to work with context (surrounding objects as contexts, existing data structures

Note Recorded by Clement Quinton, Politecnico di Milano, Italy

- How to Capture Context and Context-dependent Behavior – Tetsuo Tamai (Organizer), Hosei University, Japan

  Presentation of the different research projects (Kumiki).

  Context-awareness is required in adaptive systems. Context is determined by location, time, natural environment, technical environment, social environment...

  In Context-oriented programming, layers modularize context-dependent behavior and are activated/deactivated.

- Contexts and Unit of Adaptation in Context-oriented Programming – Hidehiko Masuhara, Tokyo Institute of Technology, Japan

  Context-dependent behavior: behavior specific to context. COP affects modules (classes, objects, methods...)

  Nice motivating example with the adventure game (drunk character, balloon for dialog). But what is a context? Hero's status (drunk or not, boolean), location (town or field, structured)

  Context may be external or internal, structural, about one or many objects. The proposition is structure-based contexts, where a context is surrounding objects.

- What is the context in self-adaptive systems? Context-Oriented Programming for Adaptive Software Systems – Tetsuo Kamina, Ritsumeikan University, Japan

  ServalCJ to express context. COSoft.Eng.: I think high connexion with feature-oriented engineering!

  How to describe context. There are features, constraints, choices to be done, etc. Feature modeling could be a solution (?)

Session 5 – Chair: John Mylopoulos

Note Recorded by Amel Bennaceur, The Open University, UK

- Dynamic Software Composition for Run-time System Evolution – Robert Hirschfeld

  Robert started by introducing the basic concepts of context-oriented programming and explained its relation with aspect programming and reflection. He then presented the concept of layers and their use in behavioural and structural scoping. In particular he explained how to organise changes into layers, which are then used to enact changes at runtime to the object/classes concerned. Finally, Robert gave directions on how to use layers in the context of self-adaptive systems using MAPE-K loop.

- High Variability Models for Better Adaptive Systems – Kostas Angelopoulos

  Kostas discussed the relation between the variability of requirements, specified using awareness requirements, and the variability of architecture and

behaviour. He refers to the intertwining of these three models as three-peaks. He then presented an approach that uses control theory to trade-off a set of potentially conflicting requirements.

Recorded by Clement Quinton, Politecnico di Milano, Italy

- Dynamic Software Composition for Run-time System Evolution - Robert Hirschfeld

  Context: everything computationally accessible

  - location, time, temperature, bandwidth...
  - age, preferences, subscriptions, energy consumption...

  Reactive approaches: constraint-based composition (ContextJS)

Session 6 – Chair: Marin Litoiu

Note Recorded by Kostas Angelopoulos, University of Trento, Italy

- Integrated Control and Systems Science for Cyber-Physical Systems: A Research Agenda, Hausi A. Muller (Organizer), University of Victoria, Canada

  CPS systems should be part of the the self-adaptive systems research agenda. The social impact of CPS is significant, e.g. wireless technologies. A lot of applications of CPS are expected in the future. There are a lot venues and funding opportunities on the topic. NIST reports on CPS are useful reading on the topic.

  CPS are smart systems that integrate physical and computational components. There is need of combination of continuous control for the physical resources and discrete control for the computational components. The target is to enrich the capabilities of the physical systems such as adaptability, autonomy, efficiency, reliability, resiliency etc. Examples of CPS: smarter planet, sustainable cities, industrial internet.

  Difference between CPS and Internet of Things (IoT). IoT is just collecting information while CPS collects information and exploits it. CPS systems are networked, distributed, real-time and adaptive. They require models, V&V and requirements specification. CPS are systems require control, feedback etc and have function such as sense monitor and analyse. Examples are smart cities, connected cars, autonomous vehicles. In ICSE there was a keynote about the CPS system of the Ferrari car. The key point is that the driver is the main controller of the system while the CPS is supporting him.

  Foundations of CPS: computing, control and communications. Expertise in all the three is required for successful CPS. Engineers and Scientists must be educated in every one of them.

  CPS require various types of control such adaptive and predictive control, composition of control and reference models. Optimisation techniques are also useful. Finally, assurances using models and V& V are also critical.

Autonomic computing reference architecture (ACRA). A hierarchy of controllers. On the top there orchestrating components and the 2 layers bellow manage the resources of the CPS using policies that are coming from the top.

Examples of such three layer models is the Kramer Magee adaptation model and Dynamico model and MIAC model.

- Next Generation Collaborative Distributed Visualization Systems on the Distributed Cloud Rick McGeer, SAP, San Francisco, USA

  The speaker claims that the distributed ubiquitous clout is the internet of the future. The Zettaflood: a zettebyte added to the world' s disks every 2 years. The network can't handle that traffic. High bandwidth sensors such is iSight cameras flood the networks, programs though reduce data from sensors. Therefore programs are important for controlling the amount of data flowing in the network.

  The Big Data visualisation was very expensive, but using Distributed Cloud could be cheaper. Localising servers helped to get the requested data very fast. However in large scale (beyond a city) the speed wasn't good.

  Example of a distributed cloud: amazon EC2, PlanetLab, GENI, SAVI, FED4FIRE, Vnode/Flare.

  Problems:
  Heterogeneous ownership/administration
  Location- and context-aware programs can deal with this heterogeneity. (Sabatucci, SEAMS 2015).?

Note Recorded by Clement Quinton, Politecnico di Milano, Italy

- Integrated Control and Systems Science for Cyber-Physical Systems: A Research Agenda – Hausi A. Muller (Organizer), University of Victoria, Canada

  CPS is really important in Self-adaptive Systems (SAS). There is a NIST report on CPS. Difference between IoT and CPS? IoT is sensing, CPS is sensing and actuating. Several control loops interfering with each other.

  The controller is the MAPE feedback loop and controls the system. We need a new discipline: software engineering @ runtime

Session 7 – Chair: Rick McGeer

Recorded by Yijun Yu, The Open University, UK

- Guaranteeing Solution Quality for SAS Optimization Problems by being Greedy, Ulrike Stege from University of Victoria, Canada

  Abstract: The talk is about greedy algorithms, using the Data Centre Scheduling problem as an example.

  Given a set of jobs, distribute them on to a server. More specifically, given a job, arrival time, deadline, process time and revenue as constraints, try to maximise the revenue.

//Marin: Deadline is a hard constraint for a job? Ulrike: Yes

The goal is to produce a fast solution, to produce good quality solution. It defines greedy algorithms and indicates when it is good to use them for good quality solutions. Greedy algorithm is not always the best. Apply three layered policy solutions: action =¿ goal =¿ utility policies [POLICY 2004] Decide on what is a better algorithm or better problem is the aim. Generic maximisation problem defined

//John M: can constraints be logical formula? Yes, possible. F is a subset of U.

//Ric: are there other matrices? Yes there might be. Specifically, (job, starting_time) is the universe, constrained by

$arrival\_time \leq starting\_time \leq deadline - processing\_time$

Objective function is the sum of revenues in a given schedule.

Objective function can be characterized as linear, submodular, or unrestricted. If function f satisfies

$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$

then it is submodular.

// Cheerful sound from Kostas' laptop

Constraint can be characterized as matroid, k-extensible, or unrestricted. If the collection F of feasible sets satisfies.

$A \subset B, B \in F$ then $A \in F$

it is said downward closed.

If it also satisfies the augmentation property:

for all $A, B \in F$ with $|B| > |A|$, there exists an element $x \in B - A$ such that $A \cup \{x\} \in F$

then F forms a matroid.

F is said k-extensible if it is downward closed and it satisfies the exchange property:

if $A \subset B$ and there exists $x \in U - B$ such that $A \cup \{x\} \in F$, then there exists $Y \subset B - A$, $|Y| \leq k$, $(B - Y) \cup \{x\} \in F$.

// Ric: what about an intermediate set C? It can help with the scheduling
// Hausi: Bashar do you have question? Bashar does not ask question, but smiles. Example: does not satisfy the augmentation property, but satisfy the exchange properties was shown. It is k-extensible but not matroid. //
Ulrike: Trust me: it works.
// Tetsuo: k is not specified? Yes, k needs to be specified
// Zhenjiang: is k a constant? Yes.

The greedy algorithm.

S = 0; A = 0;
repeat
    A = $\{e|$ S $\cup\{e\} \in$ F $\}$
    if A $\neq \emptyset$ then

u = $argmax(g(S \cup \{x\})|x \in A)$
S = S $\cup\{u\}$
end if
until A = $\emptyset$
return S

Edmonds 71: matroid + linear $\Rightarrow$ greedy is best
Fisher 78: matroid + submodular $\Rightarrow 1/2$ approximation
Mestre 06: k-extensible + linear $\Rightarrow 1/k$ approximation
Fisher 78: k-extensible + submodular $\Rightarrow 1/(k+1)$ approximation

// JM: is 1/k approximation a lower bound?
Hausi: yes, but it is a guarantee
add structure to the constraint set could get a guarantee
If processing time are unit, then it is optimal.

Three-step Recipe:

// Kostas: how does the recipe work? what about the overhead? You have to update the revenue with a trade-off function for the cost.
// Hausi: HP Lab has a report on how to formulate the objective function
// Tetsuo: complexity algorithm? it is often linear time
// Zhenjiang: Polynomial time.

Group activities to formalise your optimisation problem into Scheduling?

// fun is not over...
// case, how to recover?

Session 8 – Chair: Hausi Muller

Note Recorded by Nobukazu Yoshioka, NII, Japan

- Robust Linear Quadratic Control of Software Systems by Marin Litoiu

  There were questions as follows:

  - What's type of system can be applied with Adaptive Control?
  - How fast feedback loop should be?

- Discrete Time Adaptive Linear Control for Software Systems by Martina Maggio

  There were questions as follows:

  - It is complex with load-balancer, so can we think of a simplified model?
  - What is granularity, e.g., process, thread?
  - How about multiple executing models?
  - Is alternative computation possible?

- Assured Graceful Degradation with Discrete Controller Synthesis by Kenji Tei

  There were questions as follows:

- Do you assume that the relationship between E_{t+1} and E_t is always refinement relation?
- Does $Delta_{T+1}$ include $Delta$?
- How to define the boundary of winning region?
- How to discard abnormal case, how to recover?

Session 9 – Chair: Yijun Yu

Note Recorded by Amel Bennaceur, The Open University, UK

- A View-based Approach to Software Adaptation – Zhenjiang Hu

  Zhenjiang presented an approach to adaptation based on views as a means to bridge the gap between goal and rule-based adaptation. Views define the invariants that must be preserved through the application of the adaptation rules. The satisfaction of goals drive the definition of the views. Zhenjiang gave a formal definition of views and illustrated the proposed approach using an eCommerce example.

- Bidirectional Programming for Self-adaptive System – Lionel Montrieux

  Lionel started by introducing some background on bidirectional transformations. He then gave some ideas on potential uses of bidirectional transformations in the context of self-adaptive systems. In particular he explained how by specifying the concretisation process (put function), one can obtain some guarantees about the abstraction process (get function).

- Evolving Dynamic Software Product Lines – Clement Quinton

  Clement explained how updating variability models (feature models) at runtime may lead to inconsistencies in the associated software artefacts (Dynamic Software Product Lines). He then proposed a framework where adaptation rules are updated and managed so as maintain the variability models and the associated software artefacts consistent.

Session 10 – Chair: Zhenjiang Hu

Recorded by Kenji Tei, NII, Japan

- The Aftermath of Mystery Flight MH370: What Can Adaptive Software Engineers Do?, Yijun Yu, The Open University, UK

  To speculate causes of accidents of flights, much date is required. He showed a new concept “Internet of Flying Things” where data about flight is collected at runtime to identify symptoms of accidents in an early phase. However, bandwidth is not enough to send all raw sensor data from flights. Basic idea is to send knowledge instead of raw data. Our self-adaptive software engineering people use many kinds of models representing knowledge. These models and techniques will be useful. Challenge is how can knowledge used in MAPE-K loops be elicited, simulated, verified, and explained to ensure performance, privacy, security, and trust. We need control with the right-level of knowledge, and we need knowledge for explanation. Currently, pilots exchange knowledge with grand operators through human-to-human communications. If we do not trust pilots,

such automation is needed. However, this approach will introduce new vulnerabilities in flights. How should we deal with that? Can we trust the system?

- Software Self-Adaptivity Measurement based on Requirements Models, Zhi Jin, Peking University, China

  RE assumes that properties and constraints in the environment can be fixed in design time, but the environment and user goals may change dynamically. A certain approach is introducing a controller (MAPE-K loops) that detects changes and modify software system at runtime. How do we get specification of a controller, and how do we make it better? She introduced a view-based approach. We need to model different aspects of the outer world; environment, situations, and contexts. Controller should capture context changes and select appropriate requirements and architecture. How does the controller synchronize models used in different levels of E, R, and S, at runtime? How does it assure correctness of awareness and adaptation? Take-home message is that RE2016 is held at Beijing. Do not miss it.

- Modularity for Uncertainties in Adaptive Software Systems,
  Naoyasu Ubayashi, Kyushu University, Japan

  How do we cope with uncertainty, in particular known-unknowns, in development of software components. Among alternative designs and implementations, developers select one. However, the other designs and implementations may be valuable for some users. He introduced modularization techniques to cope with such a development time uncertainty, called "Archface-U". Archface-U modularized alternative or optional implementations. It also supports type checking. Discussions were about difference with late binding mechanism supported by other languages and about verification of such a program including non-deterministic behavior.

Session 11 – Chair: Bashar Nuseibeh

Recorded by Lionel Montrieux, NII, Japan

- Requirements-Driven Mediation for Collaborative Security by Amel Bennaceur

  Amel talks about collaborative security, i.e. the use of everyday technology to improve security.

  One of the challenges of collaborative security is to make multiple, heterogeneous, software-intensive components collaborate with each other to meet security requirements, even though they may not have been designed for it. This situation is typical in ubiquitous computing.

  Collaborative security builds on two research areas: adaptive security, and collaborative adaptation. The former allows her to reason about assets, threats, attacks and vulnerabilities. The latter allows her to reason about dynamic discovery and composition. She tries to unify these two areas, using an approach based on mediators.

Her framework (available on github) uses feature models, behaviour models, and KAOS models. Features and behaviour are strongly coupled, in the sense that a particular feature configuration will allow only a subset of behaviours. This allows her to simplify the components' behaviour depending on the feature selection, before using mediators to combine them in a way that satisfies the requirements.

Amel concludes with a few open questions related to her framework. Is it only applicable to security, or can it be generalised? What are its limitations, especially around mediators? How about users? Should they just be considered as another component? How to explain the framework's decisions in a meaningful way?

- An Adaptive Framework for Individual Privacy by Nobukazu Yoshida

  Yoshida-san starts off with a description of the Android application security model, and points out how it does not give users sufficient control over their data. Specifically, users cannot finely control their data according to their own privacy preferences.

  He proposes a privacy-aware framework that allows users a better level of control over how their data is used. He illustrates his framework with an example, where health and fitness data is collected by a service, and used to provide users with expert guidance from personal trainers, monitoring and evolution of the measurements taken, etc. In Yoshida-san's example, users are able to select how much data they want to share, with whom, and at which granularity level. It is understood that sharing more, and more fine-grained, data will make the service more useful, but also expose the user to more potential privacy breaches. Yoshida-san seems to consider privacy breaches to be the result of misuse by third parties of data they had access to or were able to infer, as opposed to data "stolen" by malicious agents exploiting the system's vulnerabilities.

  In Yoshida-san's framework, context is important. Changes in context may have an effect on users' privacy, and hence the framework is able to react to that.

  The framework is based on risk assessment, where the likelihood and consequences of breaches are assessed in order to produce privacy requirements for each user. Users need to input their privacy preferences, where they describe (on a scale) how much they would be impacted by the disclosure of a particular piece of information to a particular category of third parties. A service specification is then selected, where a high value service will carry more privacy risks, and a low value service will carry less privacy risks.

  The framework is adaptive in the sense that, from a service specification, a controller measures changes in risk for each user, and produces service behaviour models.

Session 12 & 13 Breakout Sessions and Reports

Report from Breakout Session 2 on Bidirectional Programming in Self-Adaptive Systems, Recorded by Chair Lionel Montrieux, NII, Japan

We discussed the use of bidirectional programs, and bidirectional transformations, in the context of self-adaptive systems. We identified 5 areas of interest, connected to the participants' research.

Model abstractions Bidirectional programs can be used to transform a concrete, platform-specific model of the system into an abstract, platform-independent model of the system. Adaptation can then happen on the abstract model, and changes will be propagated to the concrete model. This allows developers to support heterogeneous environments, and to migrate from one implementation to another, or from one version to another, without having to update their self-adaptation architecture.

Extraction of sub-models for efficient analysis A large model (abstract or concrete) can be expensive to analyse. Using bidirectional programs, we can extract a portion of the model for a particular analysis. This can be done multiple times, and each of these views can be used by a separate MAPE loop. Synchronisation between the views is relatively simple: every 'put'to the large model can trigger a new 'get'to each view that could be affected by the changes made.

We could go further. If the amount of data needed by a particular MAPE loop can vary, it should be possible to adapt the transformation at runtime to narrow the view, giving the mape loop the smallest view possible, all from a single bidirectional program. If the program describes a transformation over the largest view that the MAPE loop could need, it is trivial to automatically generate a transformation that produces a subset of the largest possible view.

One of the participants likened the extraction of small views to the concept of crosscutting concerns in aspect-oriented programming. This is an interesting point of view to explore, and it may lead to more interesting uses of bidirectional programs.

Beyond self-configuration: current state of the system vs. desired state of the system Modifying system models in the context of self-configuration is relatively easy: if the part of the model that represents the system can be entirely translated into configuration files, then effecting the changes is as simple as updating the configuration files, and possibly reloading the system to take the new configuration into account.

However, in general, changes to the model may not always be effected by changes to configuration files. For example, changes may have to be done through an API. Those changes may or may not succeed, and hence failures must be taken into account. In such a solution, the model may represent the *current* state of the system (if the model is extracted from the system and the environment), but if modified, it then represents the *desired* state of the system, until (and if...) the modifications are successfully reflected in the system.

Bidirectional programs can help deal with this. A program can be written to isolate the changes to be made, and to keep track of their results. Two very similar programs can then produce the /current/ model of the system and the /desired/ model of the system.

Bidirectional programs and context-oriented programming/self-adaptation We discussed the 'traditional' self-adaptation model, where gauges and probes capture the state of the system and its environment, and effectors act on the system or its environment to enact adaptation. In general, the effectors can be completely different from the gauges and probes. However, if we were able to describe relationships between these, we may be able to configure and deploy compatible pairs of gauges/probes and effectors to achieve adaptation.

Bidirectional programs for partial model Partial models can represent alternatives to choose from. It should be possible to write bidirectional programs to synchronise each partial model with the overall model, keep then synchronised, and handle conflicts and merging.

Report from Breakout Session 3 on To Control or To Optimize? That's the question — Conflict resolution in the SAS community by Co-chairs: Martina Maggio, Lund University, Sweden and Ulrike Stege, University of Victoria, Canada

SAS problems-goals

- QoS
- Performance
- Security and safety
- Quality of experience
- Failure management
- Energy
- Cost
- Privacy
- Reliability

Both optimization and control require:

- measurable goals
- measurements
- control or optimization variables (parameters)
- objective prioritization (weights)

Dimension in x axis uncertain/certain and well defined
Dimension in y axis continuous and discrete
Extremes:

- discrete and well defined: control does not work, optimization does (geographical virtual machine placement problem)
- continuous and uncertain: control works very well, optimization doesn't
- continuous and certain: battery discharge of mobile device
- the other two are unclear

When the system shows some behavior that is oscillatory (for example, one more virtual machine will increase performance a lot and removing it will decrease it a lot below the threshold that you would like to hit), then control works better than optimization.

Human contributes to uncertainty, because when human makes some choices, it is hard to accurately predict/imagine what is going to happen. Also in this case, optimization may fail because it is difficult to write a proper model.

When do we need to do simulation and when we don't? And can you do simulations in all the situations?

Report from Breakout Session 4 on Security & Privacy for Self-Adaptive Systems, Recorded by Chair Amel Bennaceur, The Open University, UK

Premise

- Secure systems are adaptive by nature
- M: detect security violation, privacy leak
- A: compute risks
- P: ranking/prioritising/trading.off/selecting countermeasure
- E: enacting countermeasures

Question 1

- Can self-adaptation techniques help us engineer secure systems in a more systematic way?

  \+ adapt the protection according to assets/threats/environment

  − If attacker also adapt their behaviour this may hurt security

  矛盾

  $\updownarrow K_p \leftrightarrow K_a \updownarrow$
  $K_p \supseteq K_a \Rightarrow SAS \Uparrow S\&P$
  $K_p \subseteq K_a \Rightarrow SAS \Downarrow S\&P$

  K_p: protector's knowledge
  K_a: attacker's knowledge

Question 2

- What is the impact of self-adaptation on security and privacy?

- Both SAS and Secure system try to best deal with unforeseen situations/deal with uncertain behaviour
- Challenge: How to ensure/maintain security when the system involve uncertain adaptive behaviour?

List of Participants

- Kostas Angelopoulos, University of Trento
- Amel Bennaceur, The Open University
- Robert Hirschfeld, Hasso Plattner Institute
- Shinichi Honiden, National Institute of Informatics
- Zhenjiang Hu, National Institute of Informatics
- Zhi Jin, Peking University
- Tetsuo Kamina, Ritsumeikan University
- Marin Litoiu, York University
- Martina Maggio, Lund University
- Hidehiko Masuhara,Tokyo Institute of Technology
- Patrick McGeer, CDG Labs, SAP Americas
- Lionel Montrieux, NII
- Hausi Muller, University of Victoria
- John Mylopoulos, University of Toronto
- Bashar Nuseibeh, Open University
- Clement Quinton, Politecnico di Milano
- Ulrike Stege, University of Victoria
- Yasuyuki Tahara, The University of Electro-Communications
- Tetsuo Tamai, Hosei University
- Kenji Tei, NII
- Naoyasu Ubayashi, Kyushu University
- Nobukazu Yoshioka, NII
- Yijun Yu, The Open University