10431 Abstracts Collection
Software Engineering for Self-Adaptive Systems
— Dagstuhl Seminar —

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

[1] Hasso-Plattner-Institut - Potsdam, DE
holger.giese@hpi.uni-potsdam.de
[2] University of Victoria, CA
[3] Carnegie Mellon University - Pittsburgh, US
mary.shaw@cs.cmu.edu
[4] University of Kent, GB
R.Delemos@kent.ac.uk

Abstract. From 24.10. to 29.10.2010, the Dagstuhl Seminar 10431 "Software Engineering for Self-Adaptive Systems" was held in Schloss Dagstuhl – Leibniz Center for Informatics. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Software engineering, self-adaptive systems, design spaces, verification and validation, processes, decentralization

10431 Report – Software Engineering for Self-Adaptive Systems

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

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

Joint work of: de Lemos, Rogerio; Giese, Holger; Müller, Hausi; Shaw, Mary

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2011/3088

Modeling & Verification of Self-Adaptive Service-Oriented Systems

Basil Becker (Hasso-Plattner-Institut - Potsdam, DE)

Future and recent software systems are promised to be extremely complex, large and for in the case of ultra-larg-scale systems (ULSS) impossible to stop for maintenance purposes. Self-adaptive systems try to provide a solution for these problems.

One way to implement an self-adaptive system is to use a service-oriented architecture. At the level of the architecture the runtime binding of service contracts, starting new component instances and terminating components result in a dynamic assembly and runtime reconfiguration of complex open systems. As new services contracts can be added at runtime as well, the dynamics goes even further and permit that the systems evolves at the level of its components as well as service contracts. I will present an approach to decompose self-adaptive service-oriented systems into smaller parts and introduce a compositional reasoning scheme that allows to show the overall system's correctness.

The case of Requirements-Aware Systems (emphasis on Self-Explanation)

Nelly Bencomo (Lancaster University, GB)

I will talk about Requirements-Aware Systems for self-adaptive systems:

Requirements are sensitive to the context in which the system-to-be must operate. Where such context is well-understood and is static or evolves slowly, existing RE techniques can be made to work well. Increasingly, however, development projects are being challenged to build systems to operate in contexts that are volatile over short periods in ways that are imperfectly understood. Such systems need to be able to adapt to new environmental contexts dynamically, but the contextual uncertainty that demands this self-adaptive ability makes it hard to formulate, validate and manage their requirements.

Different contexts may demand different requirements trade-offs. Unanticipated contexts may even lead to entirely new requirements. To help counter this uncertainty, we support requirements reflection- that is, making requirements avail able as runtime objects. We argue that requirements for selfadaptive systems should be run-time entities that can be reasoned over in order to understand the extent to which they are being satisfied and to support adaptation decisions that can take advantage of the systems' self-adaptive machinery.

We have identified five complementary and inter-linked areas needing research to realize requirements-aware systems:

1. Run-time representations of requirements
2. Evolution of the requirements model and its synchronization with the architecture
3. Dealing with uncertainty
4. Self-explanation.

I will also talk about specific partial results about RELAX language.

Credits also given to Jon Whittle, Pete Sawyer and Gordon Blair from Lancaster University (UK), Emmanuel Letier, Anthony Finkelstein from UCL (UK) and Betty H.C. Cheng (Michigan State University)

Just do it: Exploring the future.

Yuriy Brun (University of Washington - Seattle, US)

Part of most self-adaptive systems' adaptation loops involve evaluating a number of possible adaptations and selecting one to employ. Such evaluation is often performed through model analysis or simulation. In this talk, I propose executing multiple copies of the system, with adaptations enacted, in the background, helping select the best adaptation.

Keywords: Speculation

Requirements driven performance and security risk trade-off in socio technical systems

Bojan Cukic (West Virginia Univ. - Morgantown, US)

When designing critical applications, trade- offs between different security solutions and their performance implications are common. Unfortunately, understanding the precise implications of such tradeoffs early in the system development lifecycle is difficult. This presentation will outline a methodology for combined analysis of performance and security risk and their mitigation through requirements driven system adaptation. We transform system requirements into a Layered Queueing Network (LQN) model that subsequently provides analytical performance analysis feedback when considering a set of security mechanisms and incurred security risks. We quantify security risks using cost curves. The proposed approach is illustrated through a realistic case study of a border management application.

Keywords: Performance modeling, security risk, biometric classification, adaptive software

Control and Scheduling of Cluster Resources using Cloud Infrastructure Services

Ron Desmarais (University of Victoria, CA)

Advances in virtualization and cloud services have changed traditional computating architectures by allowing the systems models to be dynamic rather than static. On-demand resource allocation (e.g., using clusters) have changed the scheduling paradigm significantly. Schedulers are no longer limited to allocating service requests to static resources, but rather have the power to create computating systems on-demand to meet the requirements of service requests. This paradigm shift provides a rich environment to re-visit how scheduling and control can be used together to provide improvements in resource usage and service provisioning. This talk discusses on-demand control and scheduling architectures for cloud computing and presents results from a testbed implementation.

Keywords: Cloud Scheduling Control

Adaptation using Cloud, Control and Scheduling

Ron Desmarais (University of Victoria, CA)

To investigate the capabilities of cloud computing, scheduling, control and feedback loops to provide adaptation of the cluster sites to meet the needs of user jobs in a cost effective way.

Keywords: Cloud scheduling control adaptation

Socially enhanced Adaptation Techniques

Schahram Dustdar (TU Wien, AT)

This presentation discusses our contributions in the field of adaptation techniques for compositions of Human provided services and software based services.

Keywords: Humanes Provided Services, Adaptation Techniques for Socio-Computational Systems

Full Paper:
http://www.infosys.tuwien.ac.at/prototype/HPS/HPS_index.html

See also: http://www.infosys.tuwien.ac.at/prototype/HPS/HPS_index.html

Semantic Interface Matching

Gregor Engels (Universität Paderborn, DE)

Self-adaptive systems exchange components at runtime. In order to decide that a component provides the expected and required functional and non-functional properties, it has to be decided at runtime whether a providing component fits to a requiring component. Thus, we need appropriate holistic interface description languages as well as appropriate metrics to measure and decide whether a providing interface fits to a requiring interface.

Based on our approach of visual contracts, we are developing new concepts for a holistic interface matching, which covers functional aspects (syntax, semantics), non-functional aspects (time, security, ...) as well as management aspects (trust, compentences, skills, ...).

In detail, the following research questions are tackled:

1. We are developing a Meta-Model Algebra, which offers operators like composition, projection of refinement of meta models, while taking care of consistency and redundancy aspects.
2. To deal with the semantics of modeling languages, we apply the Dynamic Meta-Modeling (DMM) approach, which is based on the usage of graph transformations.
3. Meta-Models are parameterized by language types. It is studied how those language type parameters can be instantiated taking the semantics into account.
4. Syntactic as well as semantic matching operators are defined.

Keywords: Interface matching, dynamic meta modeling

On socio-technical concerns in the design of ubiquitous computing applications

Kurt Geihs (Universität Kassel, DE)

Ubiquitous computing embraces a model in which resources and services are discovered dynamically at run-time and integrated into applications that seamlessly blend with our daily life environment.

This inherent dynamicity creates a need for context-aware, adaptive applications that can react during run-time to the changing context.

Ubiquitous computing has a clear focus on the user, i.e. it aims at supporting the user in an invisible and intuitive way. Such applications collect, process, and store highly sensitive personalized information. Consequently, in order to foster the user acceptance of such applications non-technical concerns need to be addressed during the development process.

We have started a rather large interdisciplinary project cluster, called VENUS.

The core of the project is the development of an interdisciplinary design methodology for ubiquitous computing applications that are situation-aware and self-adaptive. The new methodology will support the construction of applications that not only satisfy the functional requirements but also comply with the given user requirements in terms of usability, trust, and legal regulations. The project involves researchers from Computer Science, Man-Machine-Interaction, Business Informatics, and Law. In our (short) seminar talk we will discuss the motivations and guidelines for VENUS as well as its overall methodological approach.

Keywords: Adaptive applications, ubiquitous computing, socio-technical concerns

Model-Driven Engineering of Self-Adaptive Software

Holger Giese (Hasso-Plattner-Institut - Potsdam, DE)

While software is an immaterial object that does not decay with time, Parnas pointed out that it is in fact aging. Lehman's laws of software evolution accordingly states that a system that is being used undergoes continuing adaption or degrades in effectiveness. Consequently, we can observe that the ability to cost-effectively adapt software has become one of the most important critical success factors for software development today.

One particular vision to address this challenge is self-adaptive software that incorporates the capability to adjust itself to the changing needs into the software itself. This capability promises at first to considerably reduce the costs for required administration and maintenance and to avoid a decline in quality. In addition, future generation of software systems that interconnect the today more or less decoupled applications into complex, evolving software landscapes will require the capability to adapt itself as an important cornerstone as the software as whole can no longer be engineered at development time.

In this talk we want review why we should look for means to engineer self-adaptive software systematically and what requirements have to be fulfilled to achieve the systematic software engineering of self-adaptive systems. Then, we will look into the particular role of models for engineering self-adaptive systems and discuss the current vision for the model-driven software engineering of self-adaptive systems. Besides the means to build self-adaptive systems with models, we will also review the role of models for the validation and verification of such systems.

Building blocks for self-adaptive systems

Karl M. Goeschka (TU Wien, AT)

To deploy self-adaptive systems in practice, configurable re-usable building blocks for control loops have to be provided.

For the activity "act" we suggest to focus on structure and interaction rather than parameters of components and constitutents of the system. This may include the run-time selection and reconfiguration of (mutable) protocols to provide for (short- term) adaptivity. Long-term evolution on the other hand needs requirements that are computationally available during run-time. Here we have first results with run-time constraints that allow to change at least some of the requirements of a system during run-time.

Automatic Workarounds for Web Applications

Alessandra Gorla (Universität Lugano, CH)

Faults in Web APIs can escape the testing process, and consequently applications relying on these libraries may fail. Reporting an issue and waiting until developers fix faults in failing Web APIs is a time consuming activity, and in this time frame many users may be affected by the same issue.

In this talk I will present a technique that finds and executes workarounds for faulty Web applications automatically and at runtime.

Automatic workarounds exploit the inherent redundancy of Web applications, whereby a functionality of the application can be obtained through different sequences of invocations of Web APIs. In general, runtime workarounds are applied in response to a failure, and require that the application remain in a consistent state before and after the execution of a workaround. Therefore, they are ideally suited for interactive Web applications, since those allow the user to act as a failure detector with minimal effort, and also either use read-only state or manage their state through a transactional data store.

This work focuses on faults found in the access libraries of widely used Web applications such as Google Maps. It starts with a classification of a number of reported faults of the Google Maps and YouTube APIs that have known workarounds. From those we derive a number of general and API-specific program-rewriting rules, which we then apply to other faults for which no workaround is known. Our experiments show that workarounds can be readily deployed within Web applications, through a simple client-side plug-in, and that program-rewriting rules derived from elementary properties of a common library can be effective in finding valid and previously unknown workarounds.

A software lifecycle process to support consistent evolutions for context-aware adaptive systems

Paola Inverardi (Universitá di L'Aquila, IT)

It is increasingly important for computing systems to evolve their behavior at run-time because of resources uncertainty and emerging user needs.
In this work we propose a modeling approach for context-aware adaptive systems

which follows a feature engineering perspective. Based on the system definition, we suggest an evolution framework where users may change unpredictably their needs and the environmental resources may predictably change their values. A definition of consistency is given to support both evolutions.
The main contribution of this paper is to propose a generic software lifecycle process for context aware adaptive system that allows systems to be managed both at design and at execution time. This process supports static and dynamic decision making mechanisms, the evolution consistency checking and it is amenable to automatic support.

Keywords: Software lifecycle process, context aware adaptive systems, consistent evolution, feature engineering

Model-based Engineering of Self-Adaptive Systems

Gabor Karsai (Vanderbilt University, US)

Model-based development centers on the use of higher-level abstractions for a system: models. However, these models can also be used beyond the development phase; namely, at operation time - to facilitate run-time adaptation. Models can represent the entire 'adaptation space' of a system, and with the support of a dynamic component framework, can facilitate the self-adaptivity in a system. The talk will discuss some generic techniques and recent examples how this can be done within a software component framework.

Self-Managed Adaptive Systems - overview

Jeff Kramer (Imperial College London, GB)

Autonomous systems need to support dynamic software adaptation in order to handle the complexity and unpredictability of the execution environment, and the changing needs of the end user. We build on our previous work on adaptive self-assembly within the three-layer model for autonomous systems to provide planning and replanning, a decentralised technique for software component self-assembly, and support for distributed execution and dynamic configuration.

Joint work of: Kramer, Jeff; Magee, Jeff

Adaptive loops in cloud computing

Marin Litoiu (York University - Toronto, CA)

In this paper, we discuss several facets of adaptation in cloud computing, the corresponding challenges and propose an architecture for addressing those challenges.

We consider a layered cloud where various cloud layers virtualize parts of the cloud infrastructure. The architecture takes into account different stakeholders in the cloud (infrastructure providers, platform providers, application providers and end users). The architecture supports self-management by automating most of the activities pertaining to optimization: monitoring, analysis and prediction, planning and execution.

Keywords: Adaptive systems

Adaptive Service Compositions

Antonia Lopes (University of Lisboa, PT)

The problem of self-optimization and adaptation in the context of customizable systems is becoming increasingly important with the emergence of complex software systems and unpredictable execution environments.

This talk presents a framework for automatically deciding on when and how to adapt a composition of adaptable services whenever it deviates from the desired behavior. In this framework, the adaptation targets of the composition are described in terms of a high-level policy that establishes goals for a set of performance indicators. The decision process is based on information provided independently for each service that describes the available adaptations, their impact on performance indicators, and any limitations or requirements.

Some experimental results using a prototype implementation of the framework in the context of a web-based application are also presented.

Joint work of: L. Rosa, L. Rodrigues, M. Hiltunen and R. Schlichting

FUSION: A Framework for Engineering Self-Tuning Self-Adaptive Software Systems.

Sam Malek (George Mason Univ. - Fairfax, US)

Self-adaptive software systems are capable of adjusting their behavior at run-time to achieve certain objectives. Such systems typically employ analytical models specified at design-time to assess their characteristics at run-time and make the appropriate adaptation decisions. However, prior to systemŠs deployment, engineers often cannot foresee the changes in the environment, requirements, and systemŠs operational profile. Therefore, any analytical model used in this setting relies on underlying assumptions that if not held at run-time make the analysis and hence the adaptation decisions inaccurate. In this talk, I provide a high-level overview of FeatUre-oriented Self-adaptatION (FUSION) framework, which aims to solve this problem by learning the impact of adaptation decisions on the systemŠs goals. The framework (1) allows for automatic online fine-tuning of the adaptation logic to unanticipated conditions, (2) reduces the upfront effort required for building such systems, and (3) makes the run-time analysis of such systems very efficient.

Keywords: Self-Adaptation, Learning, Software Architecture, Feature Orientation

Full Paper:
http://cs.gmu.edu/∼smalek/papers/FSE2010.pdf

See also: Ahmed Elkhodary, Naeem Esfahani, and Sam Malek. "FUSION: A Framework for Engineering Self-Tuning Self-Adaptive Software Systems." In proceedings of the 18th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2010), Santa Fe, NM, November 2010.

A Taxonomy of QoS-based Adaptation Methodologies for Service-Oriented Systems

Raffaela Mirandola (Politecnico di Milano, IT)

Architecting software systems according to the service-oriented paradigm, and designing runtime self-adaptable systems are two relevant research areas in today software engineering.

In this talk we address issues that lie at the intersection of these two important fields proposing a taxonomy that can be used to characterize and classify different approaches to the design and implementation of self-adaptable service-oriented systems. The taxonomy not only highlights the design and engineering similarities and differences of state-of-the-art in adaptation methodologies for SOA systems, but also identifies the areas that need further research.

Joint work of: Mirandola, Raffaela; Grassi, Vincenzo

Towards Highly Adaptive Software-Intensive Systems

Hausi Mueller (University of Victoria, CA)

Continuous software evolution, software-intensive ecosystems, highly adaptive systems, run-time verification and validation, dynamic context-awareness, control systems, location intelligence, smart services and interactions, adaptive root cause analysis, context-aware cloud scheduling

Keywords: Continuous software evolution, software-intensive ecosystems, highly adaptive systems, run-time verification and validation, dynamic context-awareness, control systems, location intelligence, smart services and

Awareness Requirements

John Mylopoulos (University of Trento - Povo, IT)

We are interested in classes of requirements that lead to adaptive behaviour for a software system under design.

One such class (named awareness requirements or AwReqs, for short) involves requirements about the success/failure of other requirements, such as r = "requirement r0 won't fail more than 3 times within a year".

Here, r constrains the failure rate of another requirement r0. Our research is addressing the following issues: (i) How do we express such requirements, (ii) How do we monitor them, (iii) How do we diagnose what is the problem if such requirements fail, or are about to fail, (iv) Compensation strategies, i.e., if an AwReq isn't being met, what are classes of possible corrective strategies that can change the situation, and how to choose among them.

This is joint work with Vitor Souza (University of Trento), Alexei Lapouchnian (University of Toronto) and Bill Robinson (Georgia State University).

Keywords: Requirements engineering, self-adaptive software, feedback loops, self-awareness

Run-time and Language Infrastructure for Self-Adaptive Systems

Oscar M. Nierstrasz (Universität Bern, CH)

Software systems are increasingly required to adapt not only in the long-term to changing business needs, but also at run-time to changing contexts.

Unfortunately neither development tools, run-time environments nor programming languages offer appropriate mechanisms to support either short or long-term adaptation. We describe some of our current efforts to address this shortfall. We are working on explicit metaobjects to support run-time evolution, first-class contexts to enable context-aware adaptation of running applications, and first-class interpreters to adapt run-time systems to new requirements. We will demonstrate prototypes of our current work.

Keywords: Software evolution, reflection, adaptation, context-awareness, virtual machines

Engineering of Self-Adaptive System : Service Robot Domain

Sooyong Park (Sogang University - Seoul, KR)

Development of full scale large self-adaptive SW require engineering methodolgy.

SPL methodology can be used for Self-adaptive SW developement Additional research is needed such as situation modeling technique, identification of adaptation point in component design, dynamic binding .. to make dynamic software product line

Keywords: Dynamic software product line engineering

Towards a general approach for the design of self-healing software

Mauro Pezze (Universität Lugano, CH)

Self-healing systems can automatically reveal failures, and locate and heal the causing faults. So far most approaches focused on some elements of the self-healing cycle and on specific classes of systems and faults.

In this talk, I will present our project that aims to define a general approach that covers all aspects of self-healing systems and can be applied to general software systems.

I will illustrate the main results achieved so far and the open challenges.

Keywords: Self-healing software systems

Towards Distrubuted, Adaptive Systems

Christian Prehofer (Fraunhofer ESK, DE)

This brief presentation compares different approaches towards the design of distributed, adaptive systems. We first look into self-organization in networks and their design principles based on analysis of existing protocols. Then, discuss control of distributed, automotive networks based on layered control architecture with multiple feedback loops.

Keywords: Self-organization, adaptive systems, autonomic computing, automotive networks

Mistral: Adaptive Management of Cloud Infrastructures

Richard D. Schlichting (AT&T Research - Florham Park, US)

Server consolidation based on virtualization is an important technique for improving power efficiency and resource utilization in cloud infrastructures. However, to ensure satisfactory performance on shared resources under changing application workloads, dynamic management of the resource pool via online adaptation is critical. Mistral is a holistic controller framework that optimizes power consumption, performance benefits, and the transient costs incurred by various adaptations and the controller itself to maximize overall utility. Mistral can handle multiple distributed applications and large-scale infrastructures through a multi-level adaptation hierarchy and scalable optimization algorithm. This talk will give a brief overview of the Mistral framework.

This work is joint with Gueyoung Jung (Georgia Tech), Matti Hiltunen (AT&T), Kaustubh Joshi (AT&T), and Calton Pu (Georgia Tech).

Analyzing Adaptation Strategies

Bradley Schmerl (Carnegie Mellon University - Pittsburgh, US)

Rainbow provides a framework for architecture-based self-adaptive systems in which an architectural model is used as the basis for observation, reflection, and adaptation. A continuing problem is how to verify that adaptation strategies solve the problems they are meant to address, and whether they leave the system in a consistent state. We are formally modeling architectural styles and operations to help us to do such verification. Our approach is to use Alloy to generate applicable states of an architecture during adaptation, and then to use Spin to check properties over these states.

Keywords: Architectural Style, Architectural Analysis

Model Problem Challenge

Mary Shaw (Carnegie Mellon University - Pittsburgh, US)

Shared examples serve a field well by providing a shared basis for explaining and comparing results and techniques. These slides set the stage for a discussion of how we can develop such a problem set for self-* problems

Self-Adaptation Challenges for SOA Environments

Dennis B. Smith (Carnegie Mellon University - Pittsburgh, US)

This talk identified research challenges for SOA systems such as design for context-awareness, automated governance, run-time monitoring, and dynamic service discovery. These challenges relate to the core themes of the seminar. They are especially problematic in using SOA in constrained environments. An early prototype in the use of SOA concepts in a constrained situation was presented. The protoype demonstrated the tracking of a vehicle by a UAV, the translation of the cursor on target data to SOAP message format and the display of the SO data on laptop displays as well as Android phones. Engineering tradeoffs included the use of UDP rather than TCP to improve video performance, the use of non-standard SOAP implementations, and employing a custom security strategy.

This prototype demonstrated that SOA principles can be used in a constrained environment. However, a number of next steps are needed to make the application realistic. These steps include building automated discovery mechanisms, accessing back-end databases and sensor devices, and building enhanced reliability, security and performance mechanisms. An additional project will focus on edge-enabled handhelds and will focus on addressing contextual and unanticipated user needs.

Keywords: Service orientation; constrained environments; mobile platforms

multi-Design

João Pedro Sousa (George Mason Univ. - Fairfax, US)

Multi-Design is a new way of thinking about multiuser systems that interact with the physical world, e.g., via the sensors and actuators in smart spaces. Multi-Design refers to a systemŠs ability to adopt multiple features and their supporting designs dynamically as a result of (a) user intentions for self-adaptation, as expressed in design artifacts, and (b) the comings and goings of users into a physical space, who bring with them design artifacts intended for automatic deployment.

Keywords: Self-adaptation; end-user design; multi-user systems; ubiquitous computing

Managing QoS Contracts Using Component-Based Self-Adaptation

Gabriel Tamura (INRIA - Lille, FR)

In this presentation we focus on the management of quality-of-service (QoS) contracts in component-based systems by using a formal approach. Our formal framework manages contracts by allowing system architects to specify service level agreements, system reflection structures and adaptation rules. These rules are triggered in response to the violation of the specified contracts when the changing context conditions invalidate them. Thus, our strategy to make a software system responsible for the preservation of its QoS contracts is to allow a system architect to reuse well known architectural patterns as adaptation rules. The soundness of this strategy is based on the recognized relationship between system QoS properties and architecture design.

For the adaptation mechanism, we use node and edge typed graphs (e-graphs) to model the managed elements of our framework, and e-graph rewriting rules to encode the adaptation rules. These rules are checked as safe, i.e., terminating and confluent, on its registration in the system to ensure the reliability of the adaptation process. The rules are applied by graph pattern-matching over the e-graph representation of the runtime system.

The termination and confluence verification frees the system architect of being aware of (i) rule dependencies that may cause deadlocks in the adaptation process; and (ii) the rule application order and the specific procedure to make the adaptation itself.

Keywords: QoS Software contracts, runtime component architecture self-adaptation, dynamic self-reconfiguration

Joint work of: Tamura, Gabriel; Casallas, Rubby; Duchien, Laurence

A Control Engineered Reference Model for Context-Based Self-Adaptation

Norha Milena Villegas (University of Victoria, CA)

Feedback-loops are important models in the engineering of adaptive software. They define the structural control elements and the behaviour of the interactions among these that are required to guarantee system properties after the run-time adaptation process. However, obtaining a sound and explicit mapping between self-adaptive software architectures and feedback-loop structures is not trivial. In this talk we present a reference model that makes explicit the crucial elements of feedback loops and with them, the feedback-loops themselves that are required for engineering context-based self-adaptive software systems. This reference model improves the development of self-adaptive software by making explicit: (i) the management of self-adaptive properties as the control reference goals; (ii) the separation of control concerns by decoupling the different feedback-loops required to achieve the reference goals over time; and (iii) the separation of context management to preserve the relevance between monitoring and control reference goals.

Keywords: Reference models for self-adaptation, dynamic context management, control loops, explicit feedback loops, goal-based adaptation, separation of concerns

Joint work of: Villegas, Norha Milena; Muller, Hausi, Tamura, Gabriel

Runtime Models as Interfaces for Adapting Software Systems

Thomas Vogel (Hasso-Plattner-Institut - Potsdam, DE)

Runtime Models as Interfaces for Adapting Software Systems

Several approaches use an architectural model as a runtime representation of a managed system for monitoring, reasoning and performing adaptation. These models are often closely related to the implementation, at a low level of abstraction, and rather based on the system's solution space. This makes them as complex as the implementation and it limits reusability and extensibility of autonomic managers. Moreover, the models often do not cover different concerns, and therefore they do not support several self-management capabilities at once.

In contrast, we propose a model-driven approach that provides multiple architectural runtime models at different levels of abstraction as a basis for adaptation. Each runtime model abstracts from the underlying system and platform leveraging reusability and extensibility of managers that work on these models.

Moreover, each model focuses on a specific concern rather from a problem space perspective which can simplify the work of autonomic managers. The different models are maintained automatically at runtime using model-driven engineering techniques, especially the incremental and bidirectional model synchronization that also reduces development efforts.

Keywords: Self-adaptive software, model-driven engineering, models at runtime, model synchronization

Failure Diagnosis

Ken Wong (University of Alberta, CA)

Determining the root cause of a failure within a distributed enterprise application is a challenging task. This talk considers diagnosis by using data mining and machine learning techniques on log files. It may be possible to discover invariants for run-time verification and validation of these applications.

Automatic Runtime Checks for Self-Healing Systems

Jochen Wuttke (Universität Lugano, CH)

Detecting software failures by monitoring design-level constraints can enable automatic responses to failures, and can help developers maintain software systems, coding design-level assertions at the code level is hard and error prone.

In this presentation, I give an overview off LuMiNous, a model-based approach to automatically generate assertions that detect violations of design-level constraints.

With this approach, software engineers encode design decisions by annotating the involved design entities, and a tool automatically maps these design-level constraints to a set of corresponding code-level assertions.

Keywords: Assertions, runtime checking

Dynamic Generation of Plans

Rogerio de Lemos (University of Kent, GB)

In this talk we present the development of a framework for the dynamic generation of processes that factors out common process generation mechanisms and provides explicit customisation points to tailor process generation capabilities to different application domains. The framework encompasses a reference process for managing the dynamic generation of processes, a reusable infrastructure for generating processes and a methodology for its instantiation in different application domains. The framework explores model driven technology for simplifying the generation of processes in different domains, and includes fault-tolerance mechanisms for dealing with faults during generation and execution of processes.