university of groningen

University Medical Center Groningen

University of Groningen

Second International Workshop on Variability in Software Architecture

Galster, Matthias; Avgeriou, Paris; Weyns, Danny; Becker, Martin

Published in:
EPRINTS-BOOK-TITLE

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2012

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):
Galster, M., Avgeriou, P., Weyns, D., & Becker, M. (2012). Second International Workshop on Variability in Software Architecture. In *EPRINTS-BOOK-TITLE* University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science.

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Download date: 15-04-2021

Second International Workshop on Variability in Software Architecture

Matthias Galster
University of Groningen, The Netherlands
mgalster@ieee.org

Danny Weyns
Linnaeus University, Sweden
danny.weyns@lnu.se

Paris Avgeriou
University of Groningen, The Netherlands
paris@cs.rug.nl

Martin Becker
Fraunhofer IESE, Germany
martin.becker@iese.fraunhofer.de

ABSTRACT
Variability is the ability of a software system or artifact to be adapted for specific contexts, in a preplanned manner. Many of today's software systems are built with variability in mind, e.g., product lines and families, self-adaptive systems, open platforms, or service-based systems that support dynamic runtime composition of web services. Variability is reflected in and facilitated through the software architecture. Also, as the software architecture is a reference point for many development activities and for achieving quality attributes, variability should be treated as a first-class and cross-cutting concern in software architecture. Therefore, the Second International Workshop on Variability in Software Architecture (VARSA 2012) aims at identifying critical challenges and progressing the state-of-the-art on variability in software architecture. VARSA 2012 is a follow-up of the First International Workshop on Variability in Software Architecture (VARSA 2011), held at WICSA 2011.

Categories and Subject Descriptors
D.2.10 [**Software Engineering**]: Design – *methodologies*. D.2.11 [**Software Engineering**]: Software Architectures – *data abstraction*. K.6.3 [**Management of Computing and Information Systems**]: Software Management – *software development*.

General Terms
Management, Documentation, Design.

Keywords
Variability, software architecture, VARSA.

1. THEME AND GOALS
The theme of the full-day workshop is variability in software architecture. The workshop is a follow-up event of the First International Workshop on Variability in Software Architecture (VARSA 2011), held at WICSA 2011 in Boulder, Colorado. The workshop report of VARSA 2011 can be found in [1].

Copyright is held by the author/owner(s).

WICSA/ECSA '12, Aug 20–24 2012, Helsinki, Finland
ACM 978-1-4503-1568-5/12/08.

VARSA 2012 aims at exploring current and emerging methods, languages, notations, technologies and tools to handle (including modeling, implementing, and managing) variability in the software architecture. We are particularly interested in industrial practice and experience. Therefore the goal of this workshop is to bring together researchers and practitioners from various areas of software architecture interested in a) variability as it occurs in software architecture (types of variability, evolution of variability, etc.), particularly in relation to quality attributes (e.g., performance, security), b) how variability can be facilitated in architecture descriptions and means to achieve variability, c) particular challenges of variability in software architecture, and d) variability as a key technique to realize self-adaptation of software-intensive systems at the architecture level.

2. OVERVIEW
Variability in software-intensive systems is understood as the ability of a software artifact (e.g., components, subsystems, or systems) to be changed for a specific context, in a preplanned manner. Here, change refers to configuring, customizing, extending, or adapting software artifacts. This means, variability is "anticipated change", facilitated through known locations of change in the architecture. Variability a) helps manage commonalities and differences between software systems, b) supports the development and evolution of different versions of software, c) facilitates planned reuse of software artifacts in multiple products, d) allows the delay of design decisions, e) supports instantiation and assessment of architecture variants during design space exploration, and f) supports runtime adaptations of deployed systems.

Many of today's software systems are built with variability in mind, e.g., product lines and families, self-adaptive systems, open platforms, or service-based systems with dynamic runtime composition of web services. Mechanisms to accommodate variability include variant management tools, configuration wizards and tools in commercial software, configuration interfaces of software components, or the dynamic runtime composition of web services. As variability is primarily reflected in and enabled through the software architecture, VARSA 2012 investigates variability at the software architecture level.

The scope of the workshop goes beyond "managing" variability. Managing variability is only one of several activities in the context of handling variability. Additional activities involved in handling variability include identifying variability (i.e., determining where variability is needed), reasoning about, specifying and implementing variability (i.e., use a variability

realization technique resolve variability at variation points and to implement a certain variant).

3. MOTIVATION

Handling variability in software architecture as well as managing its evolution is crucial when developing and maintaining software systems. The architecture is the centerpiece of software development and foundation for implementation, reviews and testing, and thus requires special attention. Identifying and managing variability of a system (either single systems, product lines, system of systems, etc.) early on, and in particular during architecting, is preferred over discovering and addressing variability later in the life cycle when adaptations are more expensive to make. Furthermore, the architecture is paramount for achieving architecture quality attributes. Currently, variability is primarily addressed in the product line domain, but not thoroughly studied as a concept that affects and is affected by many areas of software engineering. For example, the focus of modeling variability in product lines is on features and decisions, rather than treating variability related to the architecture and all architecture aspects, as a cross-cutting concern or even as a quality attribute. Moreover, a product line assumes the existence of a product line infrastructure, including related processes (e.g., core asset development, product development, management). This is rarely the case for many software systems which should support variability. However, product lines are only one way of facilitating variability.

On the other hand, in the software architecture domain, variability seems to be not well-understood and any change or difference between systems is potentially characterized as variability, without systematically managing variability [2]. A clear understanding of variability in architectures is missing. However, software architecture acknowledges that variability is a concern of different stakeholders, and in turn affects other concerns.

Also, the relation between quality attributes and variability in the architecture is only poorly understood, even though quality attributes cause major challenges in architecture practice. For example, architectures are usually described using multiple viewpoints and views. In views, only some variability concerns are of interest for a particular stakeholder. On the other hand, quality attributes are concerns that are affected by variability and thus must be represented in the architecture.

Moreover, variability in the software architecture goes beyond feature or decision models but encompasses models and views that are particularly relevant for software architecture (e.g., deployment models, information models, development models, adaptation models). As the software architecture is the centerpiece of software systems and acts as reference point for many development activities (e.g., requirements, design, implementation, maintenance), and many of today's software systems are built to accommodate variability, variability should be treated as a first-class concern in software architecture.

4. TOPICS OF INTEREST

Topics of the workshop include but are not limited to:

1. Variability in architecture description.
   a. Variability in software architecture as a cross-cutting concern beyond product line engineering / architectures.
   b. Modeling variability in different architecture model types (e.g., information models or development models) rather than merely annotating component-and-connector models or feature models.
   c. Architecture viewpoints and views (including model kinds, correspondences and correspondence rules and traceability support) to handle variability.
   d. Reference architectures for variability-intensive systems.
2. Means to achieve variability.
   a. Methods, techniques, tools, notations or languages for handling variability at the software architecture level.
   b. Architecture patterns, styles and tactics for variability with an emphasis on ensuring consistency between different software architecture artifacts and traceability to the artifacts of the following development phases.
3. Particular challenges with variability in software architecture.
   a. Variability in the architecture of large-scale systems and software eco-systems, and evolution of architectures.
   b. Variability in the context of emerging architecture paradigms (e.g., SOA, REST), in unprecedented systems and in critical domains (e.g., cyber physical systems, embedded systems, smart grid).
   c. Handling variability from requirements to / via architectures to implementation and quality assurance.
   d. Variability in quality attributes.
   e. Variability in architecture knowledge and design decisions.
4. Self-adaptive systems and architectures.
   a. Dynamic construction of applications and variability to support runtime adaptations in self-adaptive systems.
   b. Modeling variability to support runtime selection of variants.
   c. Decision making of variant selection at runtime.
   d. Consistency management of variant adaption at runtime.

5. ACCEPTED PAPERS

The workshop is open to all WICSA 2012 attendees, but presentations were selected based on reviewing submitted papers. The following papers were accepted for presentation at the workshop (listed in alphabetical order of the author's last name).

- Nadeem Abbas, Jesper Andersson and Danny Weyns – Modeling Variability in Product Lines Using Domain Quality Attribute Scenarios: While developing an educational software product line, the authors identified a lack of support to specify variability in quality concerns. To address this problem, the authors propose an approach to model variability in quality concerns. In particular, the authors propose domain quality attribute scenarios, which extend standard quality attribute scenarios with additional information to specify variability and to derive product-specific scenarios. The authors demonstrate the approach with scenarios for robustness and upgradability requirements in the educational software product line.
- Jaap Kabbedijk and Slinger Jansen – The Role of Variability Patterns in Multi-tenant Business Software: It is crucial for software vendors in the business software domain to comply with different customer requirements. Traditionally, vendors have been offering different products to different customers.

However, as multi-tenant business software systems use one software product to serve all customers, this is no longer possible. In multi-tenant systems, software vendors have to make sure that one instance of a product is variable enough to support all different requirements from all customers. This ability is defined as "tenant-based variability". The authors present a conceptual model that helps explain the role software patterns to facilitate variability in multi-tenant business software. Important aspects of patterns are explained, like forces and consequences, and how they are linked to concepts in the problem domain. The authors suggest that variability patterns help address variability in multi-tenant business software and provide a valuable vocabulary for researching, reporting, thinking about and communicating variability solutions in multi-tenant business software products.

- Juha Kuusela – How variation changes when an embedded product ceases to be embedded: The author discusses a phenomenon in the smartphone industry. The role of applications and services has increased so much that smartphone product families no longer behave like embedded product families. Product variation now happens mostly after purchase and successful product families are much smaller than before.

- Varvana Myllärniemi, Mikko Ylikangas, Mikko Raatikainen, Jari Pääkkö, Tomi Männistö and Timo Aaltonen – Configurator-as-a-Service: Tool Support for Deriving Software Architectures at Runtime: Variability in software architectures, and especially dynamic variability in software architectures, calls for tool support. The complexity involved in variability means that tools should be able to efficiently derive architectures at runtime. The authors contribute concepts and an expository instantiation of Configurator-as-a-Service (CaaS). CaaS provides integrability, separation of derivation concerns, and automation. The approach is validated with a case of social devices, where proximity-based, distributed service compositions of mobile devices are derived at runtime.

- Elisa Yumi Nakagawa – Reference Architectures and Variability: Current Status and Future Perspectives: Reference architectures are a special type of software architecture that provide a well-recognized understanding of specific domains, promoting reuse of design expertise and facilitating the development, standardization, and evolution of software systems. Designed for various domains and purposes, they have increasingly impacted important aspects of system development, such as productivity and quality. In another perspective, variability has been considered as a mechanism that facilitates software development and evolution. The author presents the current status regarding variability in reference architecture engineering. The author also presents future research perspectives, providing new directions to reference architecture engineering.

- Bedir Tekinerdogan and Hasan Sözer – Variability Viewpoint for Software Architecture Design: To facilitate the instantiation of a software architecture, variability needs to be explicitly addressed. Usually, architectural concerns are represented using architecture views that are derived from the corresponding architecture viewpoints. Different software architecture viewpoints have been introduced to support the modeling, understanding, communication and analysis of the software architecture for different stakeholders. In this paper, the authors first provide a short overview of the approaches for dealing with variability at the architecture design level and then introduce the variability viewpoint. The variability viewpoint addresses the concerns for variability and can be used to introduce variability in software architecture viewpoints.

6. PROGRAM COMMITTEE

- Jesper Andersson, Linnaeus University, Sweden.
- Len Bass, NICTA, Australia.
- David Benavides, University of Seville, Spain.
- Jan Bosch, Chalmers University, Sweden.
- Paul Clements, BigLever Software Inc., USA.
- Hassan Gomaa, George Mason University, USA.
- Rich Hilliard, Consulting Software Systems Architect, USA.
- Peter Knauber, University of Applied Sciences Mannheim, Germany.
- Jens Knodel, Fraunhofer IESE, Germany.
- Kai Koskimies, Tampere University of Technology, Finland.
- Philippe Kruchten, University of British Columbia, Canada.
- Patricia Lago, VU University Amsterdam, The Netherlands.
- Gerrit Muller, Buskerud University College, Norway.
- Martin Naedele, ABB Corporate Research, Switzerland.
- Elisa Yumi Nakagawa, University of Sao Paulo, Brazil.
- Claus Pahl, Dublin City University, Ireland.
- Klaus Schmid, University of Hildesheim, Germany.
- Michael Stal, Siemens, Germany.
- Bedir Tekinerdogan, Bilkent University, Turkey.
- Tim Trew, NDS Ltd, UK.
- Uwe Zdun, University of Vienna, Austria.

7. ACKNOWLEDGMENTS

We thank all authors for submitting their work to VARSA 2012. We also thank the members of the programming committee for their valuable reviews. This work has been partially supported by NWO SaS-LeG, contract no. 638.000.000.07N07.

8. REFERENCES

[1] M. Galster, P. Avgeriou, D. Weyns, T. Männistö, "Variability in Software Architecture – Current Practice and Challenges," ACM Sigsoft Software Engineering Notes, vol. 36, Sept. 2011, pp. 30-32.

[2] M. Galster, P. Avgeriou, "The Notion of Variability in Software Architecture – Results from a Preliminary Exploratory Study," Proc. VaMoS, ACM, January 2011, pp. 59-67.