First International Workshop on Traceability, Dependencies, and Software Architecture TDSA2011

Satellite Event of ECSA 2011, September 13, 2011, Essen, Germany

Background

Software architecture has to consider the goals of an efficient development process, a minimization of risks, and a communication of the core concepts of a software system. As relations between requirements, design concepts, and the implementation can be captured by traceability information, design traceability enables the explicit representation of such architectural knowledge.
Having established an adequate scheme for design traceability, tool-based analyses can be performed on the basis of traceability links and other dependencies that interconnect architecture desriptions with requirements, source code fragments, and other important software artifacts. These analyses can aid in various applications, such as reengineering, architecture evaluation, and change impact analysis.
However, the challenges and problems posed by traceability with all its related activities are not entirely solved yet. The TDSA workshop will therefore cover latest approaches from research and practice.

Goals and Topics

The TDSA workshop aims to bring together researchers and practitioners in the area of traceability in the context of software architecture. State-of-the-art approaches as well as practical experiences shall be discussed to identify current research questions and encourage future research activities in that area.
In detail, the topics of the workshop includes the following:

- utilization of dependencies and traceability links,
- retrieving and querying traceability and dependency information,
- defining rules and constraints for traceability links,
- visualization of dependency information, and
- maintaining traceability information

for the purposes of

- software architectural development,
- reengineering on software architecture level,
- architecture evaluation regarding quality, and
- change impact analysis.

Furthermore, we look for reports on successful applications of traceability on architectural level, and reports on tool support. Approaches integrating and combining the above mentioned topics are especially welcomed.

Programme

09:15-10:30	Invited Talk: Dependencies, Traceability and Consistency in Software Architecture: Towards a View-based Perspective	*Matthias Galster, University of Groningen, Netherlands*
	Architectural Design Decision Visualization for Architecture Design: Preliminary Results of A Controlled Experiment	*Mojtaba Shahin, Izlamic Azad University, Iran, Peng Liang, Wuhan University, China and Zengyang Li, University of Groningen, Netherlands*
11:00-12:15	Concepts and diagram elements for architectural knowledge management	*Bojan Orlic, Ionut David, Rudolf Mak and Johan Lukkien, Eindhoven University of Technology, Netherlands*
	Rationale, Decisions and Alternatives Traceability for Architecture Design	*Fabian Gilson and Vincent Englebert, University of Namur, Belgium*
14:00-15:30	Invited Talk: Reverse Engineering of Dependency Graphs via Dynamic Analysis	*Wilhelm Hasselbring, University of Kiel, Germany*
	Group Discussion I	
16:00-17:30	Group Discussion II and Workshop Conclusion	

Workshop Chairs

Matthias Riebisch	Ilmenau University of Technology	Germany
Hannes Schwarz	University of Koblenz-Landau	Germany
Yijun Yu	The Open University	UK

Program Committee

Stephan Bode	Ilmenau University of Technology	Germany
Steve Counsell	Brunell University	UK
Alexander Egyed	University of Linz	AT
Matthias Galster	University Groningen	NL
Thomas Goldschmidt	ABB Corporate Research	Germany
Wilhelm Hasselbring	Universität Kiel	Germany
Andrea Hermann	Axivion GmbH	Germany
Norbert Klein	Capgemini sd&m Research	Germany
Ivan Kurtev	University of Twente	NL

Steffen Lehnert	Ilmenau University of Technology	Germany
Carola Lilienthal	C1 WPS Workplace Solutions GmbH Hamburg	Germany
Nikos Matragkas	University of York	UK
Patrick Mukherjee	Darmstadt University of Technology	Germany
Rocco Oliveto	University of Molise	Italy
Matthias Riebisch	Ilmenau University of Technology	Germany
Hannes Schwarz	University of Koblenz-Landau	Germany
Bernhard Schaetz	fortiss	Germany
Yijun Yu	The Open University	UK

Results of the Discussion Session

Challenges for traceability: Clusters and associated questions

Practical application

- How can the "force" of traceability links and dependencies be quantified? (If quantitative information is useful.)
- How can evolution-critical properties and decisions be recognized?
- What is the right level of granularity for traceability?
- How can the level of granularity be clearly and unambiguously defined?
- How can collaboration and consistency between disciplines be ensured?
- How should traceability in agile development be dealt with?
- How can traceability links be adapted to new technologies (tools, programming languages, etc.)?
- How can customers be convinced that architecture is more than "boxes and lines"?
- How can developers be motivated to define and record traceability links?
- Are there formal methods to specify links between artifacts?
- What kind of metrics can be used for traceability?
- How can traceability dependencies and approaches be evaluated (methodologies, scenarios, empirical studies etc.)?
- In which context do traceability and dependency approaches work best?
- Does traceability really pay off?
- Can traceability be used to predict change effort?
- How can it be ensured that traceability is a means to an end and not a means to itself?
- What are good arguments for promoting traceability to corporate management?

- How can traceability be managed in highly volatile environments (e.g. product lines)?

Architectural design methodologies

- How can patterns in architectural descriptions be recognized?
- How can architectural descriptions be enhanced with the results of static and dynamic analyses?
- How can the effort to capture architectural description be reduced to facilitate fast return of investment?
- How can the average time for understanding system architecture from descriptions be decreased?
- How can evolution-critical properties and decisions be recognized?
- What is the right level of granularity for architectural descriptions?
- How can functional requirements be traced to architecture for the purpose of impact analysis?
- How can dependencies which violate good architectural design be identified?
- How can requirements be prioritized to make architectural decisions?
- How can traceability information be used to ensure consistency in architectural descriptions in case of system change?
- How can traceability and dependencies be positioned in the context of architectural knowledge management?
- How can the components of architectural descriptions be organized to cope more efficiently with system complexity instead of being a source of complexity of their own?

Tool support

- What is the right level of granularity for traceability?
- How can this level of granularity be clearly defined?
- How can traceability information between models be kept consistent?
- How can traceability links be identified and maintained automatically (in architecture tools)?
- How can traceability links be visualized?
- How can impact analysis be automated if decisions changed?
- How can different tools used in a development project be intergrated in order to interrelate their artifacts?

Completeness of traceability information

- How can the "force" of traceability links and dependencies be quantified? (If quantitative information is useful?)
- How can completeness of traceability links be ensured?
- How can it be ensured that all relevant artifacts are traced?
- What kind of metrics can be used for traceability?

Other

- How can dependency analyses help to identify domain and technical system aspects?

- How can architectural descriptions resulting from static and dynamic analyses be linked to an initial design?
- How can design changes be derived from code changes (using "diff")?
- To what extent can the variability between different devices etc. be managed?
- How can existing models be enriched with explicit traces? How can we make use of implicit traces?
- How can semantics in model transformations be preserved?
- How can implicit information be described explictly?
- Do quality attributes require special attention in the context of traceability?
- How can different approaches for traceability be integrated?

Benefits of practical application

- effort reduction for maintenance
- comprehension, assessment, prediction of resulting properties
- connecting architectural models to implementation (source files, configuration files), requirements specifications, etc.
- effort prediction
- capture constraints for changes
- coverage analysis (requirements, tests)
- requirements engineering (tracing to stakeholders, regulations, laws, etc.)
- consistency checking
- Traceability is a prerequisite if you want to work with models

Examples for tool support

- *Creation*
    - monitoring for dynamic analysis (Kieker)
    - analysis of co-change
    - AREL
    - PAKME
- *Maintenance*
    - discovery of patterns of changes (TraceMaintainer)
    - checking
- *Visualization*
    - mass data visualization (natural sciences)
    - Cytoscape
- *Exploitation*
    - graph querying language (e.g. Gremlin, XPath)
    - metrics calculation
    - test coverage analysis, requirements coverage analysis