Quality and architectural issues in open source

Katja Henttonen
Eila Niemelä

Teknologiasta liiketoimintaa

What is the role of software architecture in F/OSS?

What is software architecture?

- A commonly agreed (e.g. *Wikipedia*) short definition of software architecture is the structure of the software system including components and relationships.
- Futher, literature (e.g. Bass & al '98) identifies following meanings of software architecture:
  - language for communication (a common abstraction that different stakeholders can use as a basis of communication)
  - manifestation of the earliest design decisions (provides analysis oportunity in the early stage of development)
  - a transferable abstraction of a system (a model that can be applied to other systems with similar requirements)

Role of software architecture in F/OSS communities

- Survey on the role of software architecture in F/OSS communities (Matinlassi, 2007) indicated the following:
  - Architecture is viewed as a set of rules which guide acceptable changes into the code
  - Architecture is also viewed as an abstraction which helps to understand the system
  - Analysis of architecture quality is done through public discussions, no explicitly defined methods are in use
  - Modularity is seen as a primary requirement for software architecture
  - Architecture is reasonably documented in the beginning, but the documenation is often out-of-date.

Reasons for good quality architecture in F/OSS
(Zhao & Elbaum 2003, Matinlassi 2007)

- Peer-to-peer review
  - Peer-to-peer code review helps to ensure that source code complies to the architectural guide lines.
  - Even a possibility of review may encourage to write better code.
- Peer-to-peer support
  - Given an enough large co-developer base, almost every problem can be charactirized and fixed quickly.
  - *"Every problem is transparent to someone"* (Eric Raymond)
- Democracy
  - Averaged opinion of a big mass of equally competent experts is more reliable than an opinion of a single expert.

5

Reasons for good quality architecture in F/OSS
(Zhao & Elbaum 2003, Matinlassi 2007)

- Skilled and motivated developers
  - Best software developers are those who enjoy what they are doing.
  - Willingness to work unpaid often indicates high personal motivation.

6

Reasons for bad quality architecture in F/OSS
(Zhao & Elbaum 2003, Matinlassi 2007, Puhakka 2008)

- Lack of participation and/or democracy
  - Peer-review and peer-support only work once project has grown to a certain size, what happens before it? Small projects may not receive much feedback from co-developers nor end-users. Most open source projects (over 90%) in SourceForge are either inactive or have only one active developer.
  - Many previosly mentioned "strengths" apply to community-driven F/OSS projects, what happens with company involvement?
- Lack of documentation
  - If documenation in not sufficient, integrators and contributors fail to understand architectural decisions and their rationals

7

Software architecture design is crucial

- F/OSS development model* places many requirements on the software architecture, especially in the terms modifiability, extensibility and integrability
- F/OSS projects can hardly mature without well-designed, modular software architecture.
- Take any F/OSS success story (Apache, Eclipse, Jboss) and you will find a set of good architecture design decision behind

* e.g decentralized and user-driven requirements engineering process, co-development by stakeholders whose skills, motivations and time of involvement may vary significantly

8

How to improve the quality of software architecture?

What is quality-driven architecting?

- Quality-driven architecture design emphasizes the importance of addressing quality attributes in the earliest possible phase, i.e. in architecture design
- Quality attributes are non-functional characteristics of a software system as extensibility and performance
- Quality attributes are gathered, categorized and documented as at least equally important requirements as functional requirements. The gained knowledge is used in software architecture design.
- Software architectural styles and patterns (e.g. model-view-controller, layers) are presumed to embody different quality properties. When patterns are applied into the architecture, quality-characteristics of the selected patterns are reflected to the entire software system.

What is QADA methodology?

- QADA (Quality-driven Architecture Design and Quality Analysis) is a methodology developed in joint projects by VTT, EU and Tekes. QADA provides a systematic way to transform quality requirements into software architecture.
- While QADA is an architecture-centric methodology, it is NOT based on the traditional "water fall" development model. QADA is a highly iterative approach to architecture development and promotes reuse of architectural models.
- QADA is suited for both open and closed sourced development.
- VTT has developed an Eclipse-compatible, open souce (EPL) tool chain which support QADA methodology.

What are the benefits of QADA?

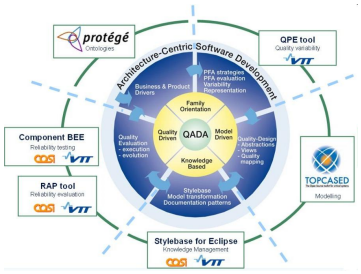
- Saves money and development resources because problems are noticed in the earliest possible stage (i.e. architecture design phase) and are thereby easier to fix
- Improves quality because architecture is made understadable to all stakeholders and their quality concerns are taken into account
- Increases productivity and reduces time-to-market because tested architectural solutions are resused several times (strong link between software testing and design)
- Decreases documenation effort since integrated tools ensure uninterrupted design flow

A Tool Chain for Quality-driven Software Architecting

Business from technology

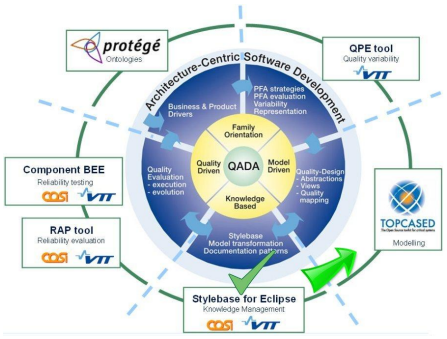
Overview

- The Quality-driven Architecture Design and quality Analysis (QADA)®
- The whole tool chain contains six co-operating tools.
- Tools work under the Eclipse platform, excluding Protégé which is open source ontology editor.



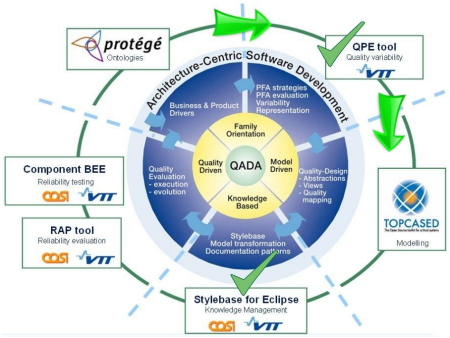
Stylebase for Eclipse
Knowledge Management

- StyleBase is a tool for Eclipse platform for managing architectural knowledge.
- The architect can search and select appropriate architectural and design patterns to achieve desired qualitative requirements, for example extensibility or modifiability.
- The StyleBase makes it possible to evaluate qualitative requirements, i.e. checking that proper styles and patterns are used.



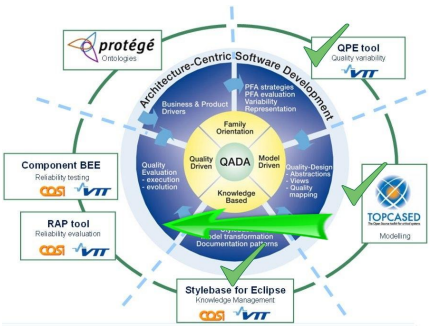
QPE tool
Quality variability

- Quality Profile Editor QPE for Eclipse platform.
- The architect can define measurable and variable quality properties.
  - These quality properties are derived from requirements specification of a family or product.
- The architect can select an appropriate metric, from the quality ontology, for each quality property. Ontologies can be defined with Protégé open source ontology editor.
- QPE stores quality properties in a UML profile form which is supported by TOPCASED.



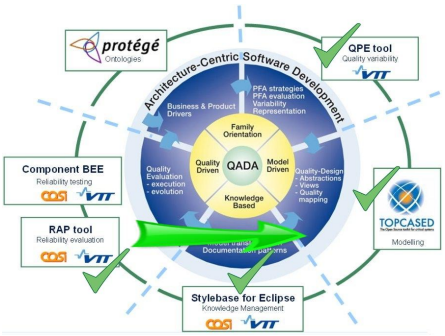
TOPCASED
Modelling

- TOPCASED is an open source UML modelling tool for Eclipse platform.
- The architect constructs software's architecture with TOPCASED using architectural and design patterns from Stylebase.
- Quantitative quality properties from the quality profile, made with QPE tool, are mapped to the architecture with TOPCASED.



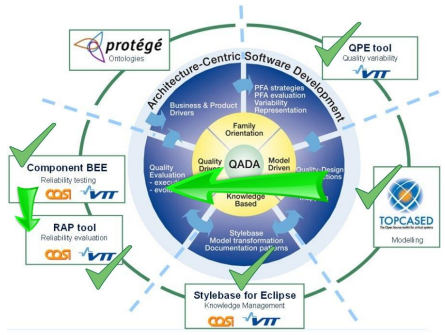
RAP tool
Reliability evaluation

- Reliability & Availability Prediction (RAP) tool for Eclipse platform.
- The architect can evaluate reliability from the architectural models.
- The RAP tool reads UML models and stores component's probability of failure values back to the UML models for TOPCASED.
- Results from the RAP tool facilitates the architect to decide whether the architecture is satisfactory and to find the most critical parts of the designed system.

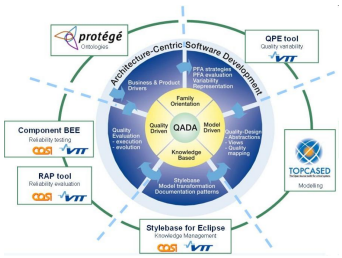


Component BEE
Reliability testing

- The ComponentBee is the only tool that does not purely work on the architectural level.
- Software designer can test an implemented software component with the ComponentBee.
- The ComponentBee produces measured reliability values for the implemented component.
- Architect can re-analyse architecture with these measured values by the RAP tool.



Questions?



Contact information:

- QADA®:
  - Eila Niemelä eila.niemela@vtt.fi
- QPE tool:
  - Antti Evesti antti.evesti@vtt.fi
- Stylebase:
  - Katja Henttonen katja.henttonen@vtt.fi
- RAP tool:
  - Antti Evesti antti.evesti@vtt.fi
- Component BEE:
  - Marko Palviainen marko.palviainen@vtt.fi