Open Source and Software Quality

Prof. Eila Niemelä
Eila.Niemela@vtt.fi

SQM/INSPIRE 2007 International conference
Tampere, Finland, 1st August 2007

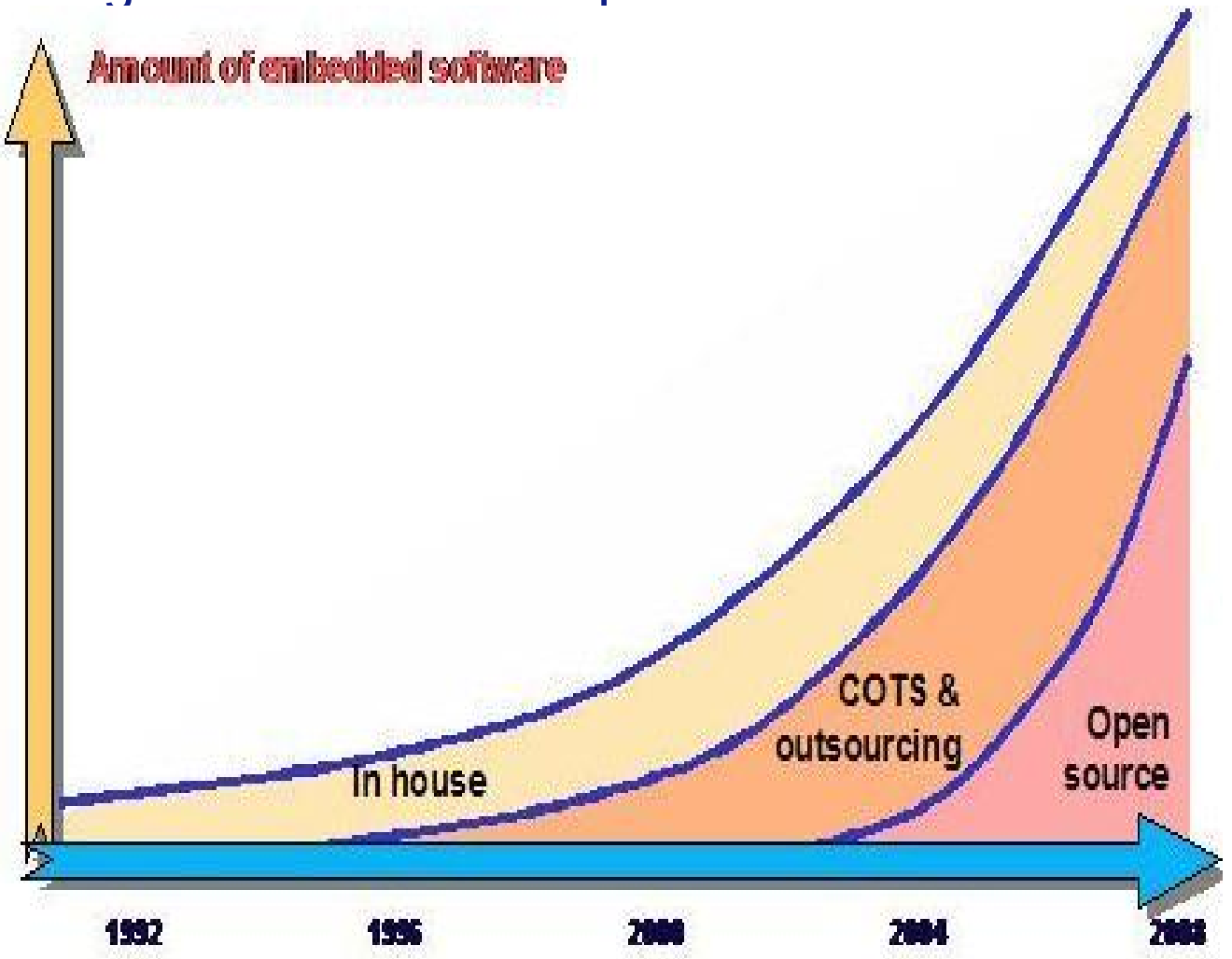
Business from technology

Outline

- Challenges in
  - Open Source software development (OSSD)
  - Software quality assurance
- OSS development and quality
  - Contributions of OSS communities
  - OSS Integration
- Quality assurance within Open Source
- Ongoing work and future research directions

Challenges in OSS development

- Open Source is emerging into the development of software intensive systems



Source: COSI/ITEA project

Challenges in OSS development

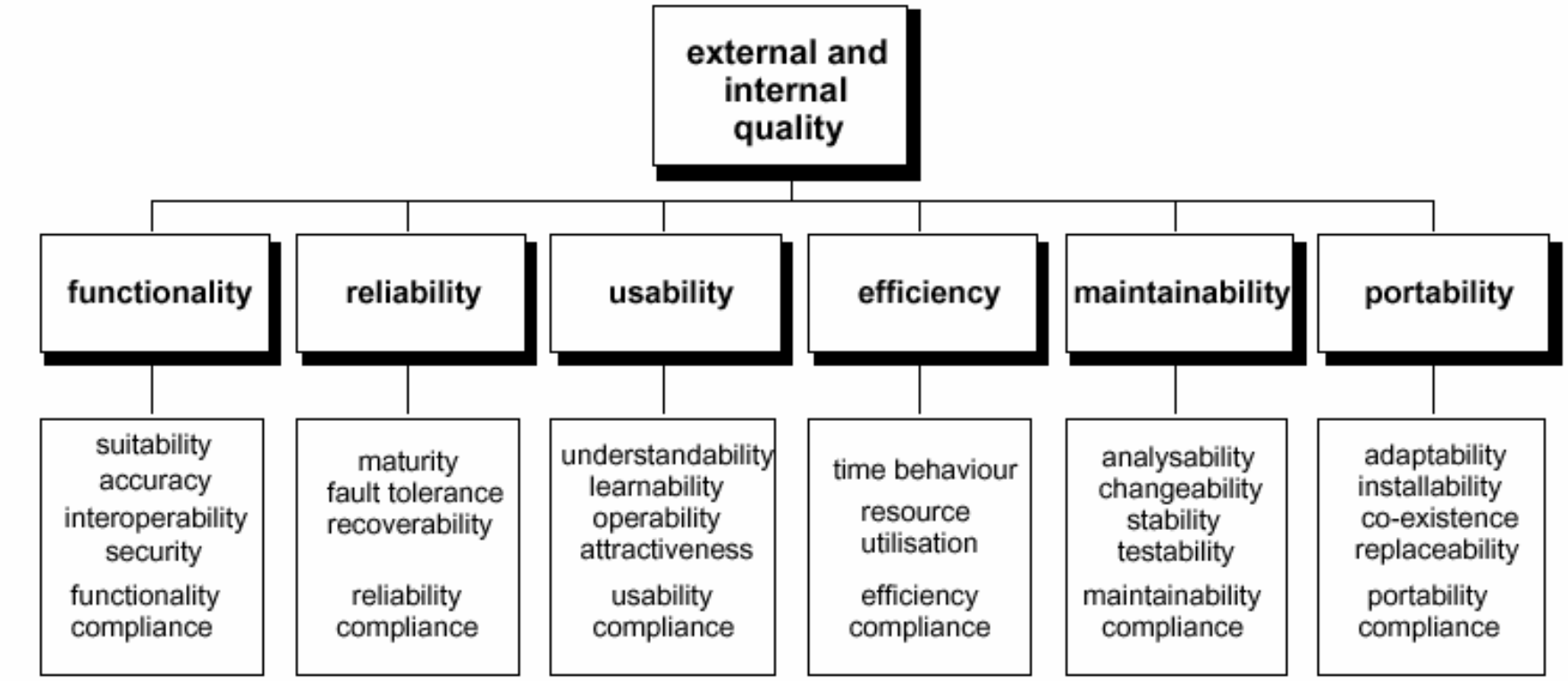
- Open Source has given many promises but there is not knowledge how well OS fits to existing practices of companies, e.g. architecture design, software integration and testing

- Companies are facing with the problems
  - How to get benefit of OSS?
  - What licenses to use?
  - How to integrate Open Source and closed software ?
  - How to align model-based software development and OSS?
  - How to ensure trustworthiness of the OS community and OSS?

Challenges in Software Quality Assurance

- Definitions of quality attributes
  - ISO/IEC 9126-1: Quality model
  - Not suitable for modern software engineering
- Quality metrics
  - ISO/IEC 9126-2…4; external and internal metrics, quality in use
  - Most metrics concern process not product (software)
  - Lack of metrics, e.g. for security
- Prediction/measuring techniques
  - Prediction models exist but not used in practice because of missing tools (design-time)
  - No systematic techniques for run-time measuring; some add hoc approaches applied by researchers

9126-1 Quality Model



OSS Development and Quality

- Contributions of OS communities
- State of the art and practice of OSS integration

Reasons for High Quality in OS 1/2

(Source: Zhao & Elbaum 2003)

Peer-to-Peer Review

- Peer-to-peer review is an effective quality control mechanism which is rarely used in closed source environment.
- Even a posibility 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.
- Eric Raymond: *"Every problem is transparent to someone."*
  "Given enough eyeballs, all bugs are shallow" (Linus's law)

Democracy

- Averaged opinion of a big mass of equally competent experts is usually more reliable than the opinion of a single expert.

Reasons for High Quality in OS 2/2

End-User Participation

- Given an enough large beta-tester base, almost any bug can be found quickly.
- In mature and big open source projects, end-users discover and report 50 % of the faults.
- Users who do not wish to participate in testing, can select an older, stable release.

Skilled Developers

- Best programmers are those who enjoy what they are doing. Hackers have fun doing what they are good at!
- Willingness to work unpaid often indicates high personal motivation.

Reasons for Poor Quality in OS

Lack of Participation

- Unlike big and mature projects, small projects may not receive much feedback from co-developers nor end-users.

Lack of Interest in Documentation

- Hackers enjoy programming, not writing documents. Even big projects sometimes have no documentation other than installation instructions.
- Public mailing list and issue trackers serve as an alternative way of documenting. However, they may be difficult to read for an outsider.

Lack of Interest in Usability

- Many projects are born out of personal need for a tool. Developers are happy when THEY can use the tool.

OSS Integration

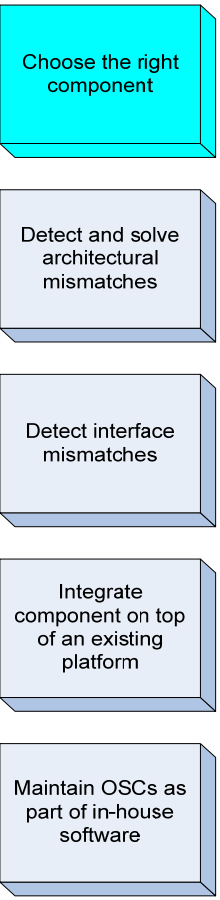
- Software integration is the process where separately developed pieces of software are merged together in order to get enhanced functionality of a system

- Integration can be horizontal or vertical
  - Horizontal integration is solved by selecting a proper architectural style and the patterns which support the specific needs of integration
  - Vertical integration is supported by three techniques; model-driven architecture, middleware services and virtual machines

OSS integration - state of the art and practice

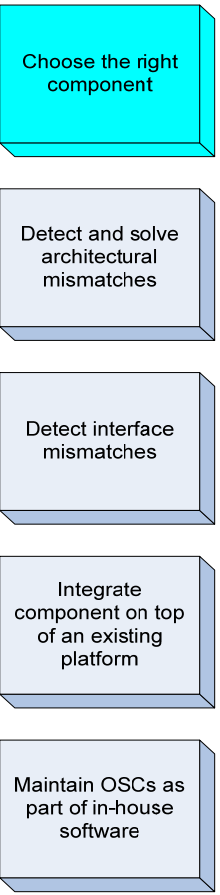
- How to
  - Choose the right component
  - Detect and solve architectural mismatches
  - Detect interface mismatches
  - Integrate components on top of an existing platform
  - Maintain OS components (OSCs) as part of in-house software

Choosing the Right Component 1/2



- Choosing the right OS component is not easy
  - There are thousands of components available to choose from
  - There is no common practice for documenting OSCs to ease the evaluation although there are templates for documenting components
    - The lack of documentation is one of the fundamental problems with the current OSS development trend
      - Alternative source of information are usenet articles, bulletin boards and chat logs
      - Reverse-engineering the OSC may also be considered to produce more documentation

Choosing the Right Component 2/2



- As an example of OSC selection criteria, OSC should:
  - Build on and follow a mature and commonly used industry standard
  - Have a strong OSS community, i.e. lots of satisfied users
  - Be broadly supported by several ISVs (independent software vendor) for distribution, evolution and support
  - Have a clear, indisputable legal status regarding IPR and the right to use it
  - Provide simple, integrator friendly design, and
  - Be mature enough (version)

Horizontal Integration
-Architectural Characteristics

Choose the right component

Detect and solve architectural mismatches

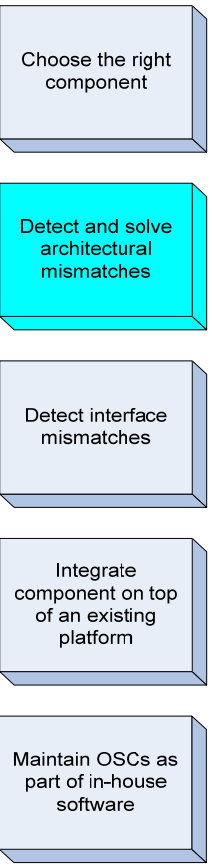
Detect interface mismatches

Integrate component on top of an existing platform

Maintain OSCs as part of in-house software

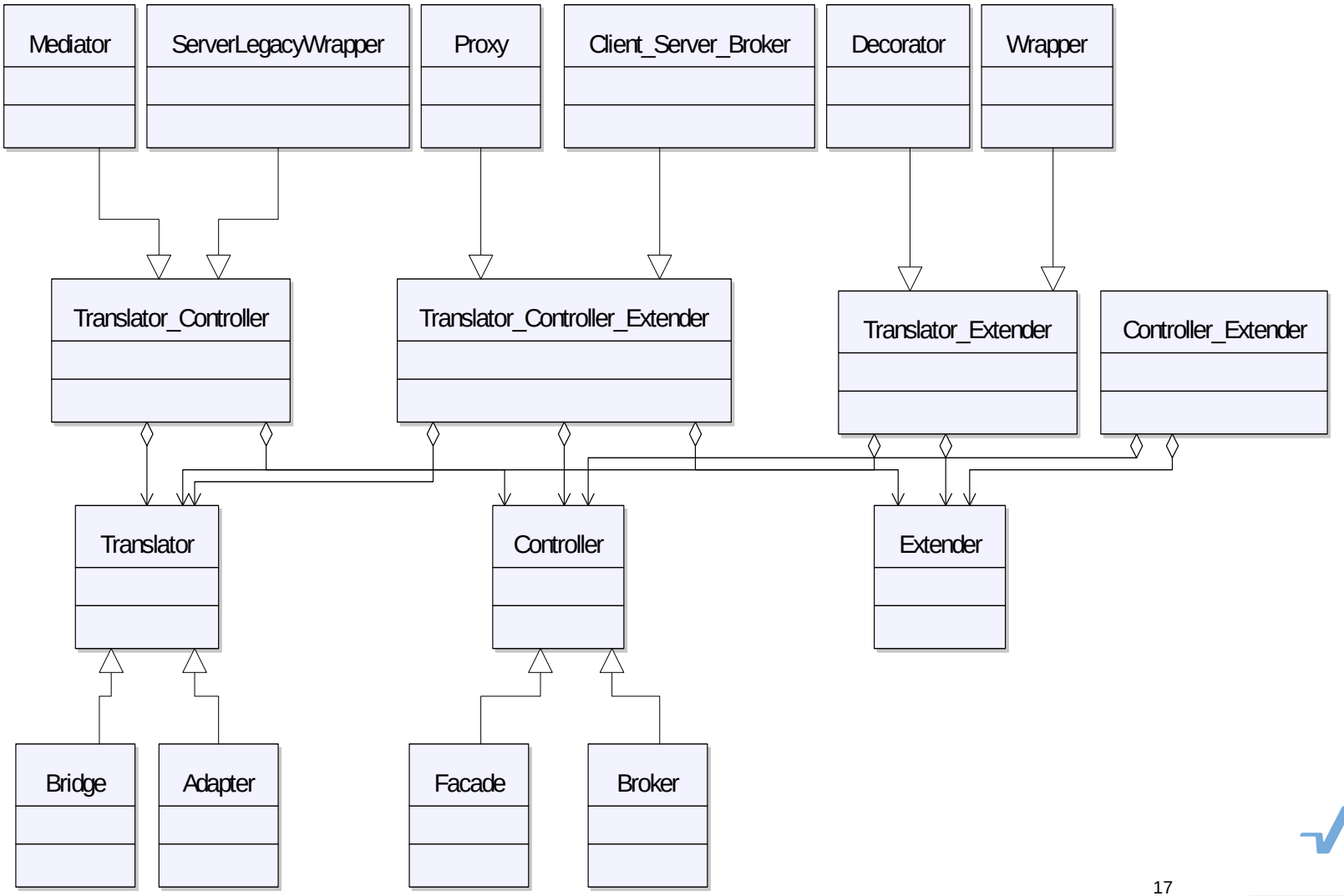
- Software architecture denotes a structure or structures of a system, which represent software components and their externally visible properties, and relationships among them (Bass et al.)
- Architectural characteristics describe coarse-grained features of an architecture
  - Davis et al. have gathered 21 architectural characteristics from the initial set of 74 found from literature
    - Orientation level
    - Latitude level
    - Execution level
  - Architectural characteristics have potential to denote early warnings of interoperability problems among components

Horizontal Integration
-Problematic Architecture Interaction



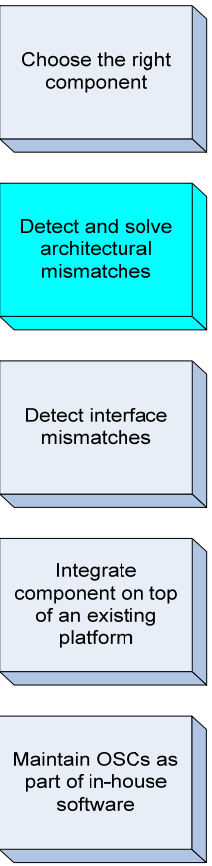
- Problematic architecture interactions (PAI) are identified by applying a set of rules
- PAIs are categorized into three categories
  - Control transfer
  - Data transfer
  - Interaction initialization
- When the PAIs are identified, *integration elements* can be utilized to overcome the possible architectural mismatches
  - Three kinds of *integration elements*
    - Translator
    - Controller
    - Extender
  - *Integration elements* can be implemented with design and archtectural patterns

Inegration Elements



Horizontal Integration

-Architectural Characteristics Mapped to QADA®



- In Davis et al. approach, architectural characteristics were mapped directly to the components
  - Only one viewpoint supported
- In QADA, architecture is described at two abstraction levels
  - Conceptual
  - Concrete
  - Viewpoints of QADA
    - Structure, Behaviour, Deployment, Development
- Immonen et al. map the architectural characteristics to the viewpoints of QADA
  - Provides more insight on how the architectural characteristics are realized
  - Assists interoperability analysis while architecting
  - Model-driven approach!

Horizontal Integration
-Component Level

Choose the right component

Detect and solve architectural mismatches

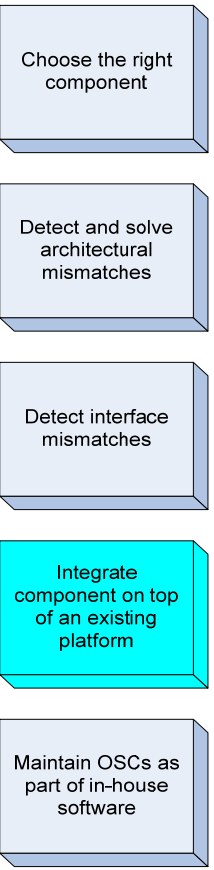
Detect interface mismatches

Integrate component on top of an existing platform

Maintain OSCs as part of in-house software

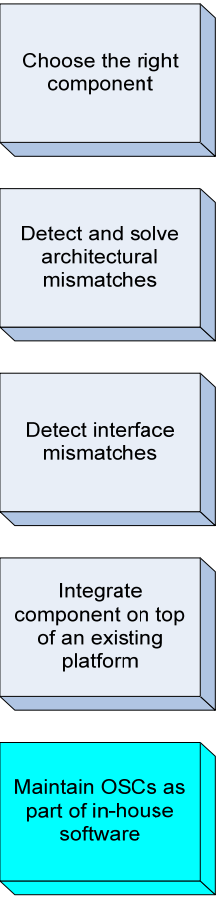
- Architectural characteristics describe coarse-grained features of components in a way that high-level mismatches can be detected and solved
  - However, neither component syntactic interface nor any other detailed behavioural aspects of components are defined
- Design level contracts are divided into four categories
  - Syntactic interface, i.e. operations, input and output parameters
  - Constraints, i.e. pre- and post-conditions
  - Synchronization and timing
  - Quality-of-service, e.g. timing and quality of result

Vertical Integration



- Three techniques for overcoming (some) vertical integration problems are identified
  - Middleware services, e.g. CORBA, J2EE
    - Middlewares restrict programming languages, operating systems etc.
    - Dependencies on middleware
  - Virtual machine, e.g. JVM
  - Model-Driven Architecture
    - Maturity of the technologies and frameworks is questionable

Maintaining OSCs as Part of In-House Software



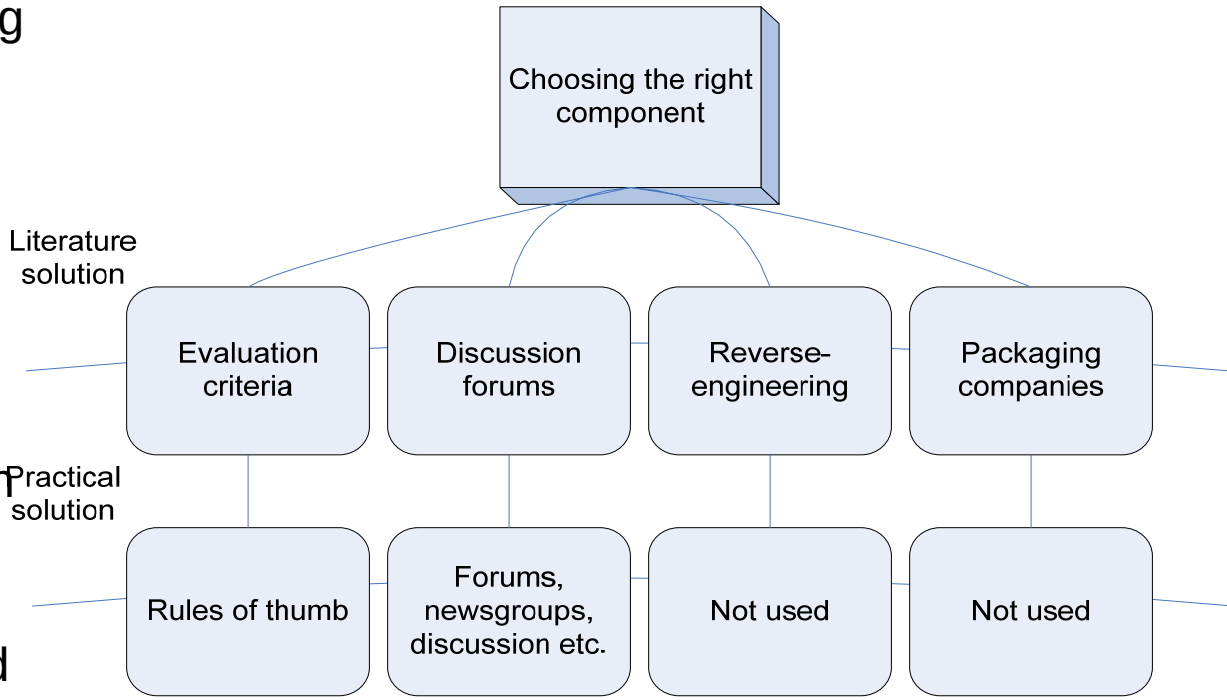
- Three ways for avoiding maintenance problems are identified
  - Treat OSCs similarly as closed source components, i.e. integrate the component with glue code
    - May be problematic when new versions of the OSCs emerge
  - Contributing to the OS project
    - Try to modify and tailor OSC so that it fits in-house software needs
      - *Outsourcing* the glue code
  - Use of packaging companies
    - Packaging companies provide support, stability of OSCs, training and documentation etc.
      - Treating OSCs as COTS

State-of-the-Practice of OSS Integration
- General info on the companies

- Age and employees
  - 7/9 companies are young (3 years or less) and have less than 10 employees
    - Breakthrough of OS and new business possibilities for small IT companies
- Use of OS
  - 5/9 companies have products which are mostly OS
  - 5/9 companies have more than 3 years of experience with OSCs
    - Business use of OS was many times backed up personal experiences
  - Most of the advantages of OS are related to development
    - E.g. Shared development, testing and maintenance
  - Most notorious disadvantage is related to business
    - Licensing issues were mentioned by 7/9 companies

State of the Practice
-Choosing the Component

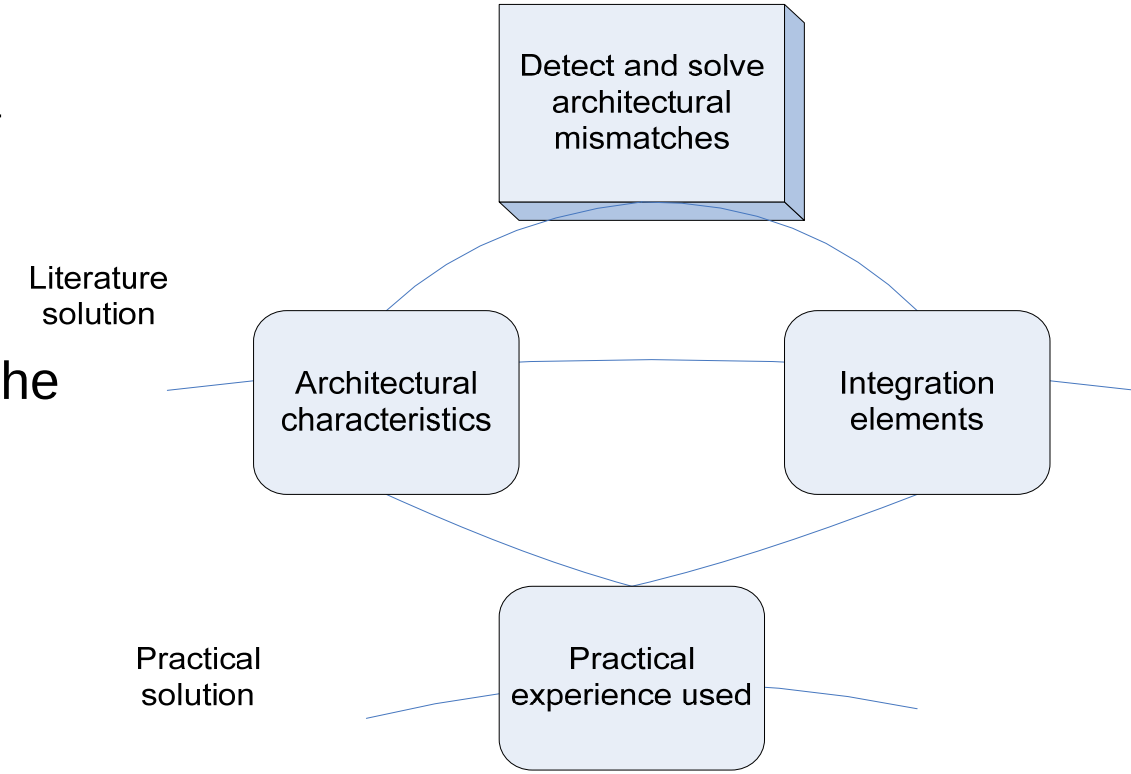
- No systematic methods for selecting OSCs
  - Usually rules of thumb were applied
    - Community age
    - Liveliness
    - Active members etc.
  - Generally, purpose was to gain an idea of the maturity of the components
  - No *packaging companies* used
- Lack of documentation is a common problem
  - Mailing lists, newsgroups, message boards etc. help the issue



State of the Practice

-Horizontal Integration at Architecture Level

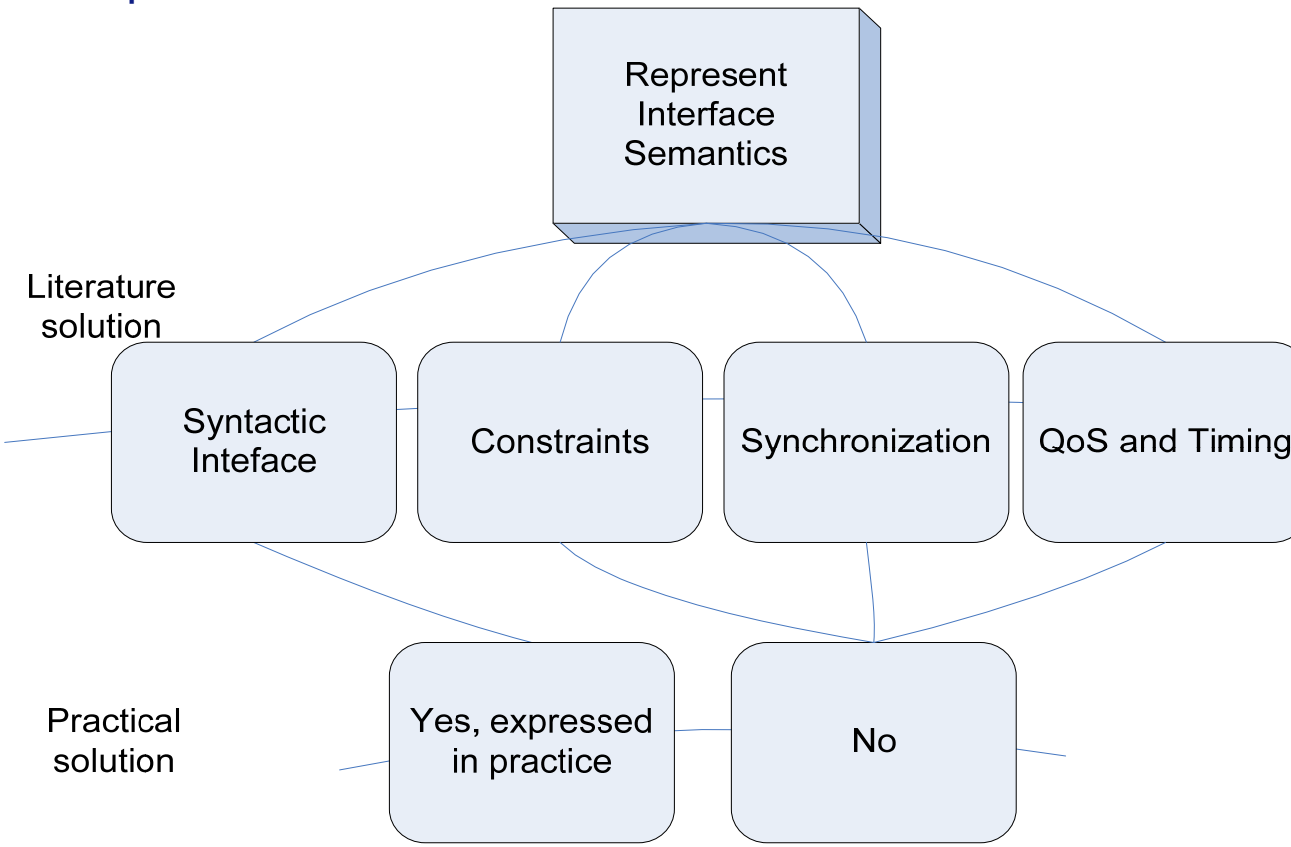
- Architectural mismatches were not a primary concern
  - Mismatches were avoided by proper component selection?
  - Mismatches are not handled at the architecture level?
    - Agile methods?
  - …or the interviewees were not aware of these?



State of the Practice

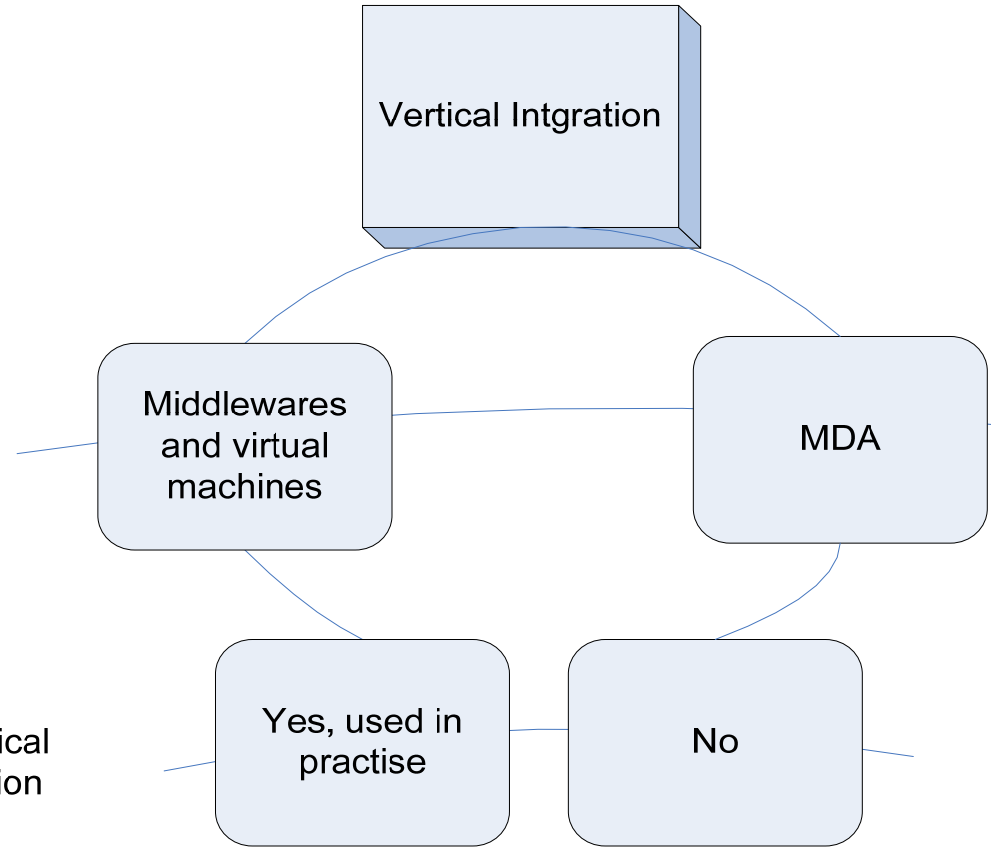
-Horizontal Integration at Component Level

- Describing component interfaces in more accuratelly than syntactics was not common
  - Not feasible although interface issues were a common problem
    - Lack of knowledge about the components the real problem?
    - Lack of knowledge about systematic description techniques?



State of the Practice
-Vertical Integration

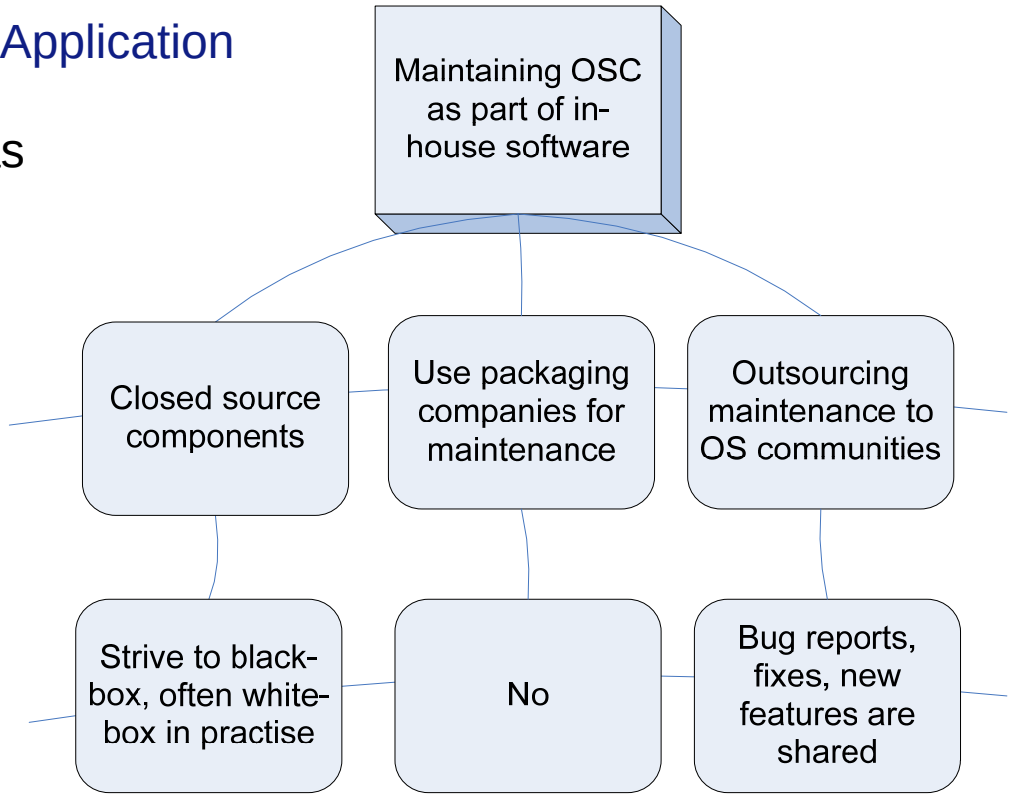
- Platform issues were the primary concern
  - Middlewares and virtual machines were used as a common practice to fight the vertical integration problems
    - …but these were not able to solve all of the problems
  - Model-Driven Architecture was not applied
    - Rigorous modelling was not practiced generally
    - No models of OSCs



State of the Practice

-Maintaining OSCs as Part of In-House Application

- Treating OSCs as black-box components was the primary means for avoiding maintenance issues
  - Often the white-box approach was applied in practice
    - Companies strove to influence the OS communities
      - Bug reports and fixes
      - New features
      - New projects (seldom)
  - Freezing the used OSC version was the primary technique to avoid maintenance troubles frequently arriving releases cause
  - No packaging companies were used



Quality Assurance within OSS

- Definitions
- State-of-the art in communities
  - QA done in the OS communities,
  - Advantages and Disadvantages of OSSD related to quality,
  - Comparisons between COTS and OSCs,
- State-of-the practice in companies
  - General on QA in companies,
  - QA activities in companies,
  - Perceptions about QA in OS communities,
- Summary

Definitions

- The definition quality assessment is usually referred as the process of evaluating the quality of the product. The quality assessment provides information on whether the (quality assurance and) quality control activities have been effective or not.

- Quality assurance (QA): "*All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality*" [ISO 8402]

- Quality control (QC): "*A set of activities designed to evaluate the quality of developed or manufactured products*" [IEEE 610.12]
- The focus is on quality assurance and not quality assessment

State-of-the-Art
- Building blocks of QA that are focused

- QA components
  - Software QA system consists of several components.
  - The focus is on the development time project lifecycle components that are most relevant to OSSD and to companies utilizing OSCs
- Project life cycle components [Galin]
  - Two stages: development and maintenance
  - Development stage
    - Reviews,
    - Expert opinions,
    - Testing, and
    - Assurance of the quality of the external participants work

State-of-the-Art
- QA done in the OS communities

- Studies conducted by Zhao and Elbaum (2000 &2003)
- Determining the QA under the OSSD model through a survey among OS projects [Zhao 1] [Zhao 2]
  - Reviews
    - Average number of people conducting reviews in an OS project = 1.3
      - With large projects (10kLOC-100kLOC) = 7.5
      - Extensive peer-review?
    - 75% of the OS developers believed that anyone who downloaded the source code or documents would check them
  - Testing
    - In the year 2000 80% of the development projects did not have any testing plans
    - In the year 2000 majority of projects spent > 40% of the development time in testing; in 2003 only 15% spent > 40%
    - Larger projects do not spend more time in testing than smaller projects
    - Larger projects have more mature validation techniques

State-of-the-Art
- QA done in the OS communities

- Testing coverage
  - In the year 2000 majority of developers did not assess testing coverage. In 2003 the assesment had improved but still only 5% used tools for it.
    - For OSC users this may mean redundancy and inefficiency
- End-user testing
  - Users found 20-40% of the bugs in nearly 20% of the projects
  - In large projects users found 80% of the "hard bugs"
    - End user testing does not suit very well to commercial OSC users
- Roles in QA
  - OS project leaders and core developers are often responsible for the quality of the software [Halloran]
    - Assuring the quality of the work of other developers

State-of-the-Art
- Advantages and Disadvantages of OSSD related to quality

- Advantages: [Yilmaz et al.]
  - Scalable division of labour
  - Short feedback loops
  - Effective leverage of user community expertise
  - Greater opportunity for analysis and validation
- Disadvantages [Michlmayr et al.]
  - Unsupported code
  - Configuration testing
  - Bug reporting
  - Security updates
  - Attracting volunteers
  - Documentation

State-of-the-Art
- Comparisons between COTS and OSCs 1/2

- Vendors: Commercial COTS vendors vs. OS communities
  - Driving force
    - COTS vendors are driven by financial profits
    - OS developers are driven by variety of factors, most of which are personal
    - COTS vendors strive for cost efficient QA whereas OSSD is succesfull regardless of redundanciy in QA activities
  - Connection between vendors and users
    - In the OSSD much of the communication is open and there is a direct connection between users and developers (i.e. connection to the real expertise that created the component)
      - Easy to get information and help when problems occur (i.e. bugs are found) - also emphasized in the interviews

State-of-the-Art
- Comparisons between COTS and OSCs 2/2

- Components - biggest difference is the unavailability of source code which causes problems
  - Testing
    - Component user has to rely on black-box testing methods only
    - Locating and fixing bugs is practically impossible
    - Many test coverage criteria are based on source code
      - E.g. statement coverage, branch coverage, path coverage, …
  - Other
    - Developing or customizing the component is prohibited or restricted
    - Development and maintenance if the vendor goes out of business

State-of-the-Art
- Summary

- Focus is on development time project lifecycle components of QA
- OSSD has the potential for remarkable peer-review
  - Is it reality?
- Many OS projects lack test plans and coverage information [Zhao 1]
  - Maturity of validation techniques increases with bigger projects
  - When selecting a component and evaluating its quality it may be advisable to evaluate the community as well.

- OS solves many problems of COTS that are related to availability of source code but it suffers from problems that are caused by voluntarism

State-of-the-Practice of Quality Assurance
- General on the QA

- Formality of QA
  - 5/9 have a defined QA process
  - Only one company follows any quality standards (ISO 9001:2000 & 14001:1996)
- QA Problems
  - 6/9 state that QA is or can be a bottleneck in development
    - Only one of them sees it as a current problem
- Impact of OS to total QA costs
  - Most of the companies have products which are mainly OS and OS has been part of the development from the beginning - estimations are rough
  - Only 2/9 state that QA costs for OSCs are bigger than for in-house software
  - 3/9 state that QA costs depend heavily on the component
    - "Depends on the maturity of component, in worst case the same as in-house"
  - Overall the costs of QA for OSCs was in general moderate

State-of-the-practice of Quality Assurance
- QA activities

- Testing of in-house software
  - 9/9 conduct static testing (i.e. reviews) for in-house software
    - Formal testing (formal inspections) is more popular with in-house software than with OSCs
  - 9/9 conduct dynamic testing for in-house software
- Testing of OSCs
  - 8/9 conduct static testing on OSCs
    - Informal static testing (especially walkthroughs and checklists) are more popular than formal static testing
  - 8/9 conduct dynamic testing on OSCs
    - One out of these eight conduct dynamic testing only if executable tests are provided with the component

State-of-the-practice of Quality Assurance
- Perceptions about QA in OS communities

- QA information availability
  - 7/9 have found useful information about QA related to OSCs
    - The information was usually scarce and availability varied a lot
    - "If the information is available then it in itself is a sign of high quality"
  - General view was that more information on QA is required
- QA material usefulness
  - 6/9 would like to have test material related to OSC
    - Especially executable test scripts and test benches were preferred
  - The availability of QA material is not decisive when selecting components
    - Built-in tests, test scripts and/or test benches to test the component in the new environment are not decisive?
- Expectations concerning performed QA activities related to OSCs
  - 4/9 expect (but do not assume) that OSCs have gone through static and/or dynamic testing
  - 2/9 expect different QA activities based on the community and component

State-of-the-practice of Quality Assurance
-Analysis

- Formality of QA
  - Only a few companies have a defined QA process
    - **Unifying factors for these companies is that they have used OS for a short period of time or majority of their products is not OS**
  - The informality of QA in the companies can be explained with following factors:
    - Small company size (<10 people)
    - 5/9 are 3 years old or younger
    - In 5/9 companies majority of software is OS. Therefore the QA efforts are shared with the community.
    - Cultural difference - end users are testers
    - Fast release cycles of OSCs
    - Only one company identified QA as a current bottleneck
    - Agile development methods

State-of-the-practice of Quality Assurance
-Analysis

- Impact of OS to total QA costs
  - **None of the companies give very exact estimatations about the impact of OSCs to total QA costs or even comparisons to in-house software.**
  - This can be explained with following factors
    - Lack of comparative information
    - Informality of QA in general
    - Lot of the companies have used OS from the beginning - meaning that there has been no actual impact
    - QA costs seem to be component-specific - generalizations impossible

State-of-the-practice of Quality Assurance
-Analysis

- Testing
  - **The amount and variety of testing activities for OSCs and in-house software is surprising**
    - Testing is performed but how systematic and rigorous it is?
      - 4/9 have no defined QA process
    - Most of the companies had no special QA team (neither for in-house or OSCs)
  - **There is virtually no difference between testing activities conducted to in-house software and OSCs**
    - Same activities: static and dynamic - With OSCs the approach to testing was a bit more informal than with in-house software.
    - It may be that much of the dynamic testing with OSCs is actually integration testing with in-house software
      - No special component testing phase for OSCs

State-of-the-practice of Quality Assurance
-Analysis

- QA information availability and usefulness
  - **In general the QA information of OSCs is valued and companies are willing to acquiring it (especially executable test scripts)**
    - QA information was not particularly searched for, e.g. executable test scripts were hoped for but not expected
    - Cultural factors - generally only something executable was seen to have real value
- Expectations concerning performed QA activities related to OSCs
  - **Majority of companies have very little expectations about the QA**
    - All kinds of components
    - All kinds of communities
    - No regulations, contracts, obligations

State-of-the-practice of Quality Assurance
-Summary

- Quality assurance in general
  - Practiced but not usually very formal
- Testing
  - Many kinds of testing activities for OSCs and in-house software
- Availability and usefulness of QA information related to OSCs
  - Majority have found useful information and material

State-of-the-Art and practice of Quality Assurance
- Results

- The cornerstones of high quality OSSD are peer-reviews and testing
  - With bigger projects the validation methods are more mature
    - Introducing better validation methods (e.g. Model-Based testing) first to bigger projects (e.g. commercially sponsored) from where they can find their way to smaller projects
  - We need:
    - Efficient testing methodologies (e.g. test reuse)
    - New light test methods to be adopted by communities and companies
    - OS based tools for MBT
    - More OS based tools to support reviews, bug reporting and test coverage

State-of-the-Art and practice of Quality Assurance
- Results

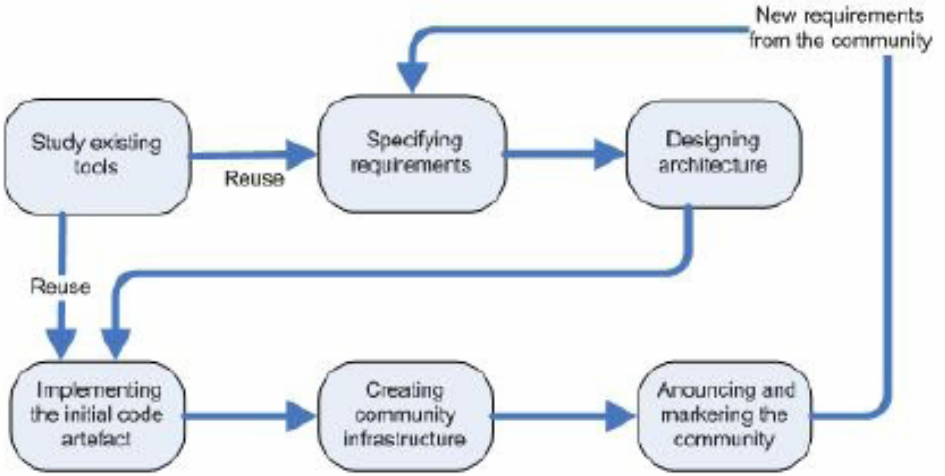
- Openness of the source code solves many problems related to COTS

- Based on literature study issues caused by voluntary nature of OSSD is the biggest problem
- Based on interviews licensing is the biggest problem

- QA information and material (e.g. test scripts) were valued but they were not decisive when selecting components
- Majority of companies have little expectations on QA in communities

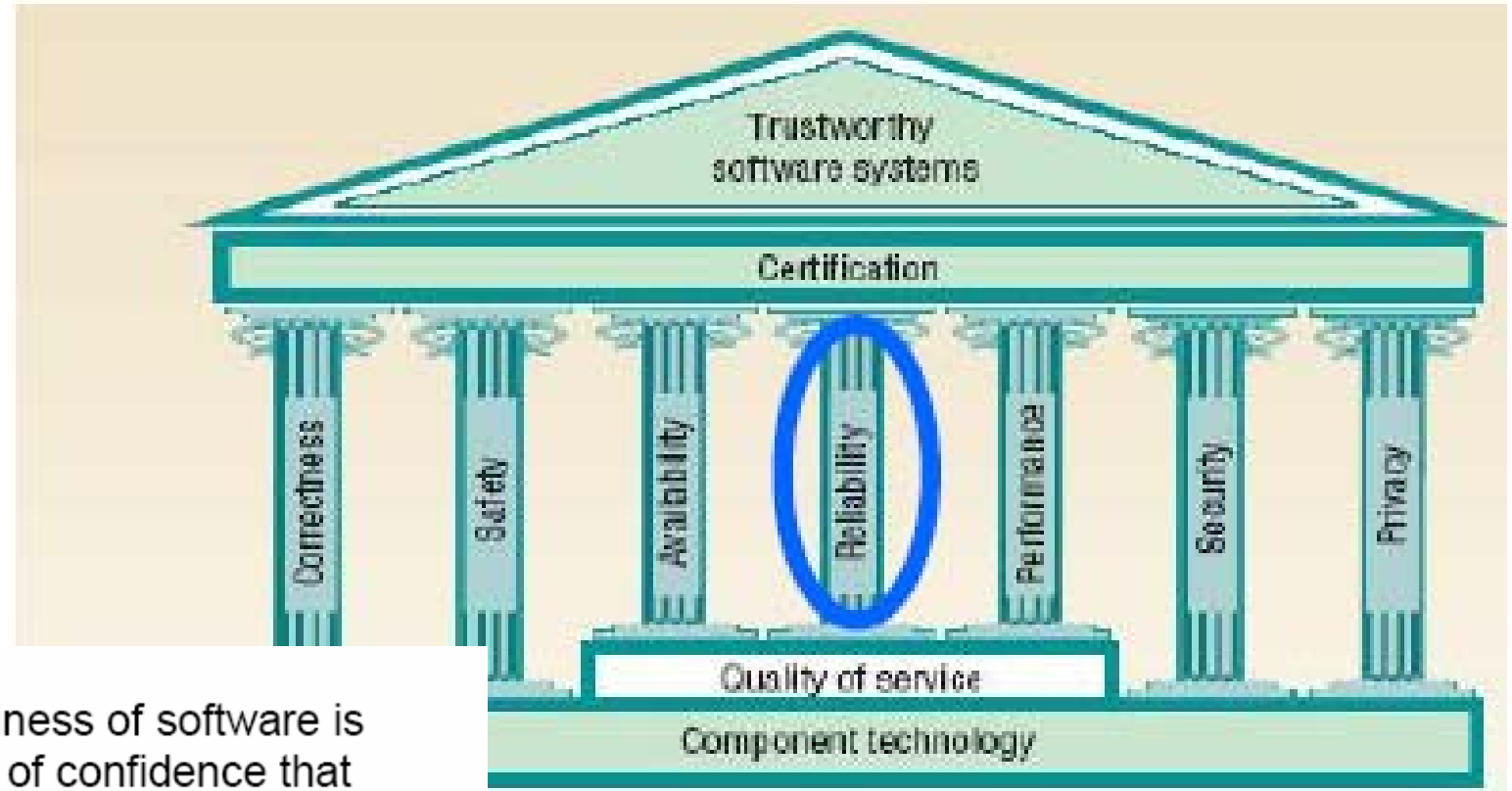
Our on-going work - Sylebase for Eclipse

- The Stylebase for Eclipse community has been established (2006)
- Evaluating existing OS projects on the given area
- Specifying requirements
- Selecting a license
- Developing the base code
- Applying the license to the code artefact
- Building infrastructure for information management
- Announcing and marketing the new community



http://stylebase.sourceforge.net/

Our on-going work
- Trustworthiness of OSS



[Hasselbring and Reussner]

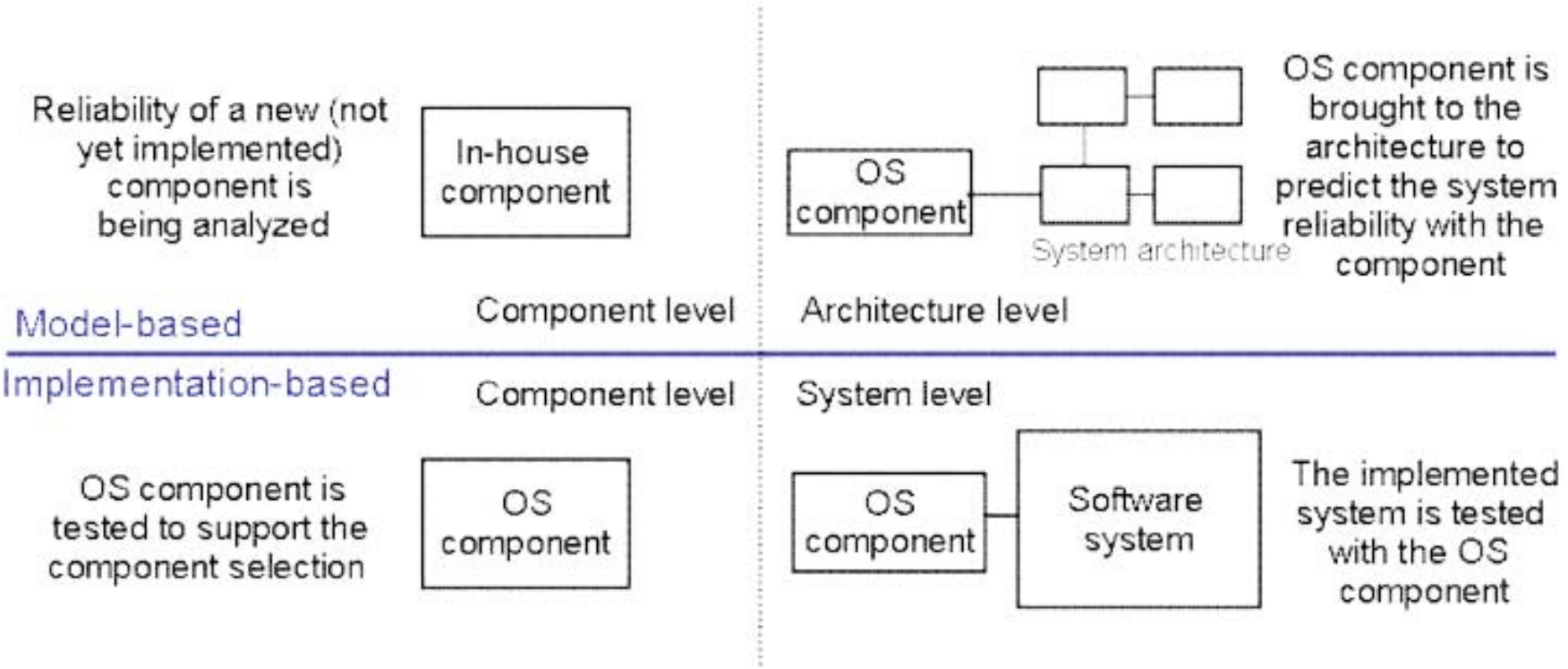
Trustworthiness of software is the degree of confidence that exists that the software meets a set of requirements.
[Amoroso et al.]

Our on-going work
- Evaluation of trustworthiness

Levels	Technical evaluation	Non-technical evaluation
Component	•Predicted reliability of new components •Analyzed reliability of OS components with the help of testing •Reliability testing of OS components	•Component certification performed by OS community •Community organization and reputation in a domain •Component development process •Component license, reputation understandability and evolution
Architecture	•Predicted reliability of the software system	•Dependencies, constraints •Compliance with standards
System	•Reliability testing of the system with the OS components.	•Component installation •Delivery

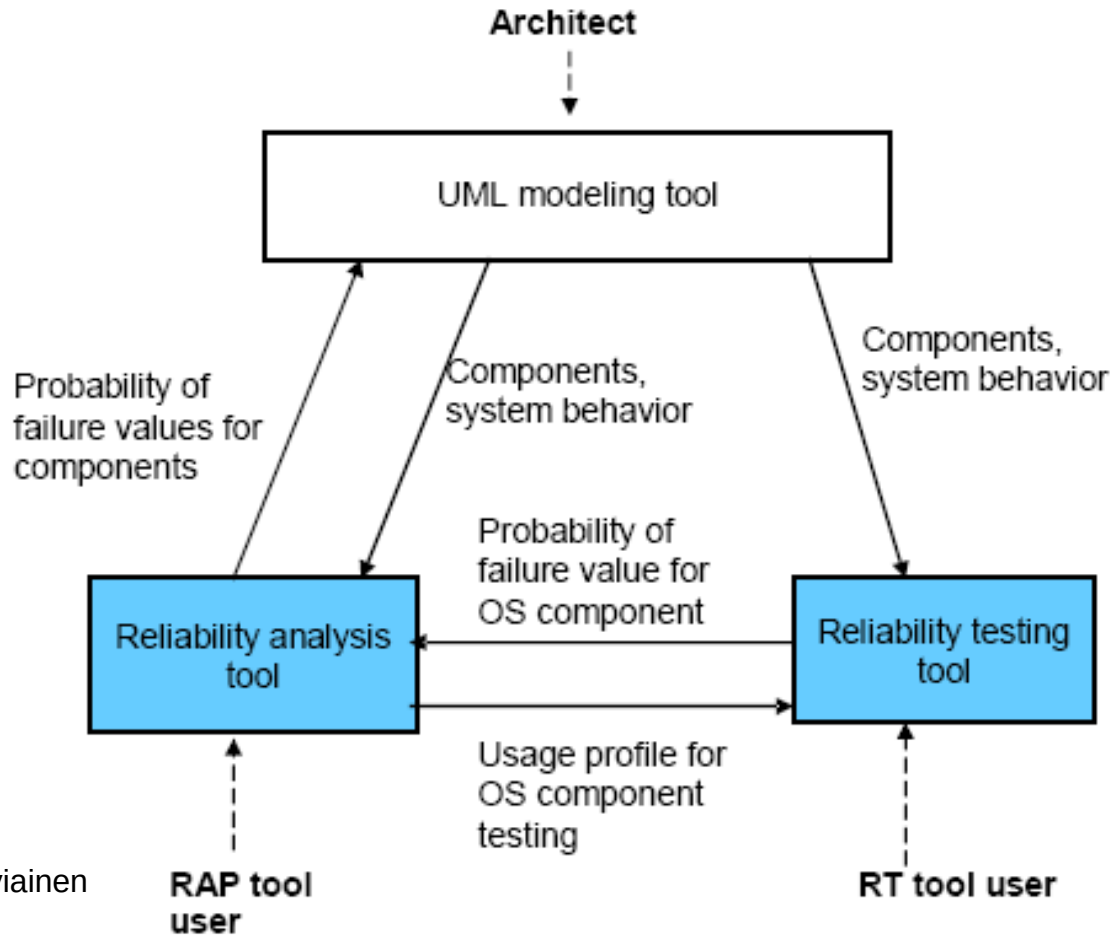
Ref: Immonen & Palviainen

Our on-going work
- Technical trustworthiness



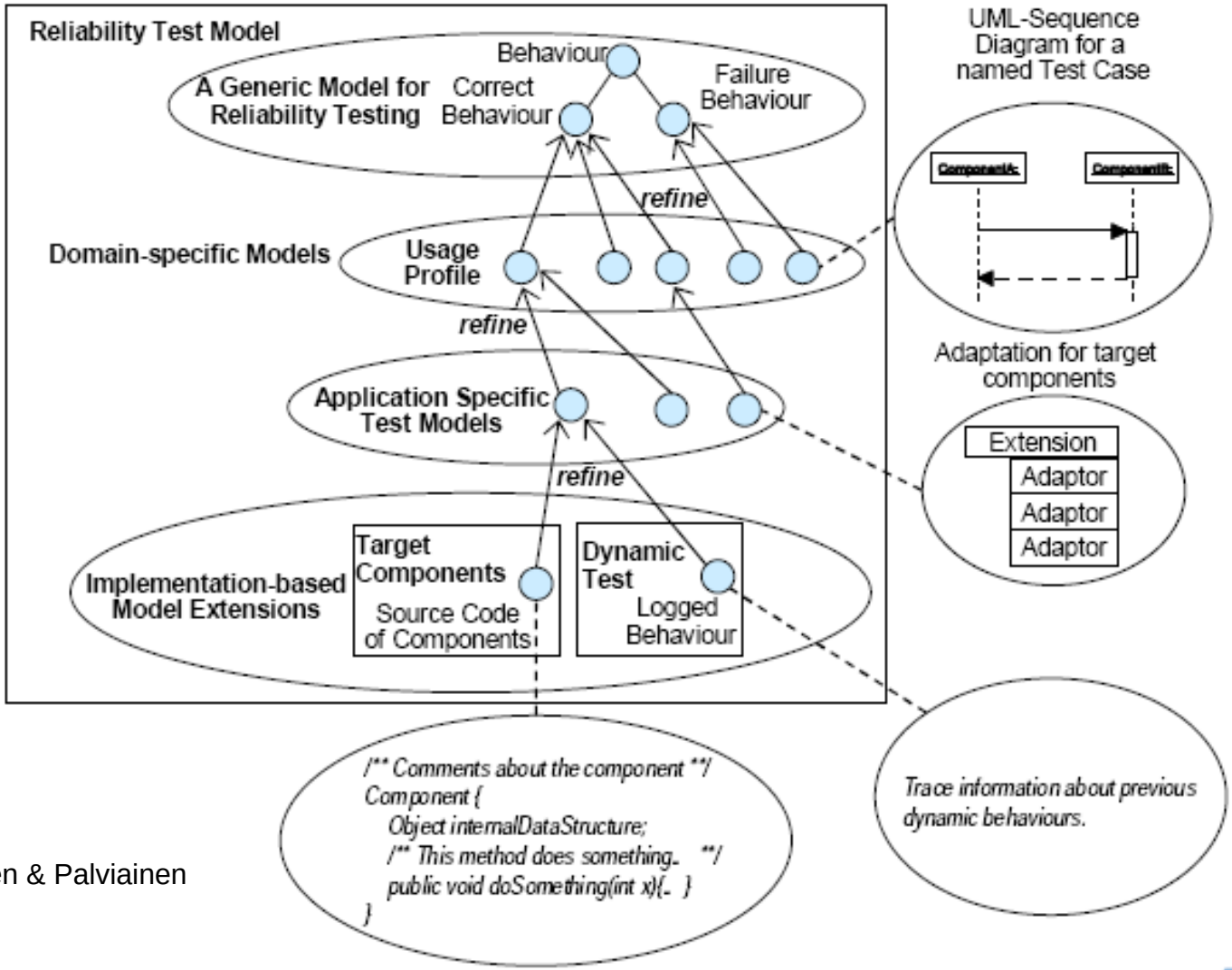
Ref: Immonen & Palviainen

Our on-going work - Evaluation tools



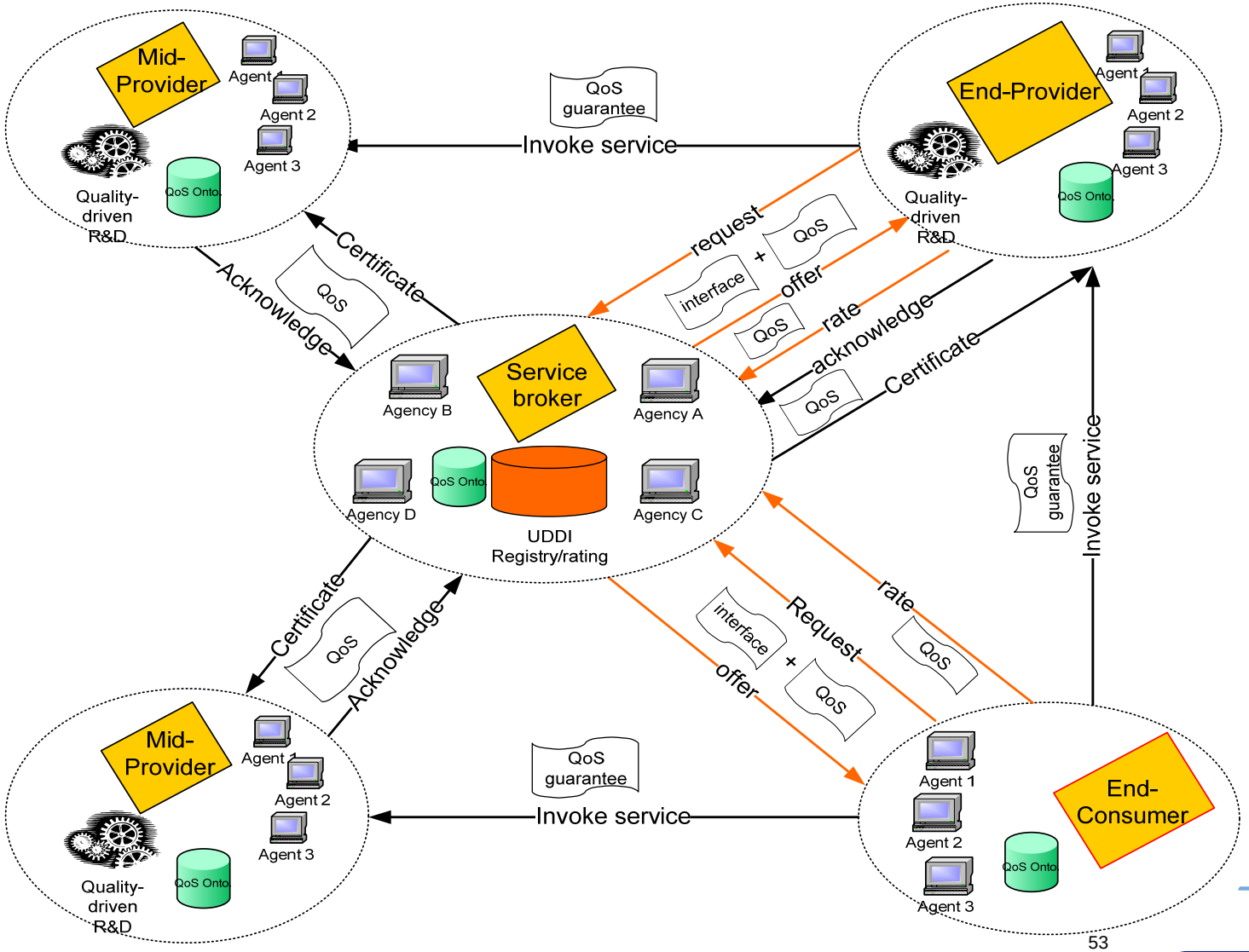
Ref: Immonen & Palviainen

Our on-going work
- Extendable test models



Ref: Immonen & Palviainen

Integrated quality-aware software/service development and management



Ref. Zhou et al.

Future research items

- Definition of software (service) related metrics
- Measuring (monitoring) technique(s) for each metric
- Test automation for OSSD
- OSS business models and licensing

Thank You!

Publications

See at: www.vtt.fi/proj/cosi

State-of-the-art and state-of-the-practice

- Merilinna, J. and Matinlassi, M. 2006. State of the art and practice of open source component integration. Proceedings of the 32th EUROMICRO CONFERENCE on Software Engineering and Advanced Applications (SEAA) Component-Based Software Engineering Track Cavtat/Dubrovnik (Croatia), August 28-September 1, 2006, pp. 170 - 177.
- Mäki-Asiala, P., and Matinlassi, M. 2006. Quality Assurance of Open Source Components: Integrator Point of View. Proceedings of the 30th Annual International Computer Software and Applications Conference, Second International Workshop on Testing and Quality Assurance for Component-Based Systems TQACBS, Chicago, September 17-21, 2OO6, Volume II/Short papers, pp. 189 - 192.
- Matinlassi, M. Role of Software Architecture in Open Source Communities. In the proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture WICSA 2007, Mumbai, India, January 6 - 9 2007, Working session paper, 4 p.

Integration techniques

- Merilinna, J., Matinlassi, M. 2007. Openware Integration Technique for In-house Software and Open Source Components. Poster. *Accepted* for the Third International Conference on Open Source Systems. June 11 - 14, Limerick, Ireland.
- Merilinna, J., Matinlassi, M. 2007. Product Family Approach for Integration of In-house Software and Open Source Components. *Accepted* for the The Third International Conference on Open Source Systems, OSSPL07, co-located with OSS2007 conference. June 14, Limerick, June.

Publications

Stylebase - initiating an open source community

- Henttonen, K. 2007. Stylebase for Eclipse - An open source tool to support the modeling of quality-driven software architecture. Bachelor's Thesis. Available as VTT Research Note.
- Henttonen, K., Matinlassi, M. 2007. Contributing to Eclipse - a case study. Proceedings of the Software Engineering 2007 conference, SE2007. Hamburg, Germany, 27 - 30 March 2007.

Trustworthiness in open source context

- Immonen, A., Palviainen, M. 2007. Trustworthiness Evaluation and Testing of Open Source Components. *Submitted* to the Seventh International Conference on Quality Software, QSIC2007, 11-12, October, 2007, Portland, Oregon, USA

Quality-aware service engineering

- Zhou, J., Niemelä, E., Savolainen, P. An integrated QoSaware Service Development and Management Framework, Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA), Mumbai, India, 6-9 Jan. 2007, 10 p.

References

- Zhao, L. & Elbaum, S. Quality assurance under the open source development model, Journal of Systems and Software, Volume 66, Issue 1, 15 April 2003, pp. 65-75 .
- L. Davis, Gamble, R. F., Payton, J., "The impact of component architectures on interoperability," The Journal of Systems and Software, vol. 61, 2002, pp. 31-45.
- L. Davis, R. Gamble, J. Payton, G. Jonsdottir and D. Underwood, "A Notation for Problematic Architecture Interactions," In Foundations of Software Engineering '01, 2001.
- R. Keshav, R. Gamble, "Towards a Taxonomy of Architecture Integration Strategies," 3rd International Software Architecture Workshop, 1-2, November, 1998.
- A. Beugnard, J.-M. Jézéquel, and N. Plouzeau, "Making Components Contract Aware," IEEE Computer, vol. 32, 1999, pp. 38-45.
- C. Bac, O. Berger, V. Deborde, and B. Hamet, "Why and how to contribute to libre software when you integrate them into an in-house application?," in Proceedings of the 1st International Conference on Open Source Systems. Genova, Italy, 2005, pp. 113-118.
- C. Ruffin and C. Ebert, "Using open source software in product development: a primer," IEEE Software, vol. 21, 2004, pp. 82-86.
- A. Taulavuori, "Component documentation in the context of software product lines," VTT Electronics, Espoo, VTT Publications 484, 2002.
- M. Nijdam, "Five suggestions for selection of OSS components," Informatie (in Dutch), vol. 45, 2003, pp. 28-30.
- M. Levesque, "Fundamental issues with open source software development," First Monday, vol. 9, 2004.

References

- A. Chakrabarti, L. de Alfaro, T. Henzinger, M. Jurdzinski, and F. Mang, "Interface compatibility checking for software modules," In 14th international conference on computer aided verification, LNCS 2404: Springer, 2002, pp. 428-441.
- L. de Alfaro and T. Henzinger, "Interface Automata," Proceedings of the Symposium on Foundations of Software Engineering: ACM, 2001, pp. 109 - 120.
- W. Damm and D. Harel, "LSCs: Breathing life into Message Sequence Charts," Journal of Formal Methods Design, vol. 19, 2001, pp. 45-80.
- OMG. "UML Profile for Schedulability, Performance, and Time Specification," ptc/02-03-02, OMG Adopted Specification.
- M. Matinlassi, E. Niemelä and L. Dobrica, Quality-driven architecture design and analysis method: A revolutionary initiation approach to a product line architecture, VTT Technical Research Centre of Finland, Oulu, 2002, 129 p.
- A. Immonen, E. Niemelä and M. Matinlassi, Evaluating the integrability of COTS components - Software Product Family Viewpoint, Testing Commercial-off-the-Shelf Components and Systems, Springer, 2005.
- T. Trew, "Enabling the Smooth Integration of Core Assets: Defining and Packaging Architectural Rules for a Family of Embedded Products", 9th Software Product Line Conf., 2005.
- T. Trew and G. Soepenberg, "Identifying Technical Risks in Third-Party Software for Embedded Products", 5th Int. Conf. on COTS-Based Software Systems, 2006.
- ComFoRT Reasoning Framework, http://www.sei.cmu.edu/pacc/comfort.html
- K. Wallnau, S. Hissam and R. Seacord, ”Building systems from commercial components”, Addison-Wesley, 2002.

References

- Assurance - Vocabulary", Switzerland
- The Institute of Electrical and Electronics Engineers Inc., IEEE Std 601.12-1990 (1991). "IEEE Standard Glossary of Software Engineering Terminology", USA, 84 p.
- Galin D., (2004) Software quality assurance : from theory to implementation. Harlow, Essex, UK: Addison-Wesley, 590 p.
- Halloran, T., J., & Scherlis, W., L., (2002) "High Quality and Open Source Software Practices." In Proceedings of the 2nd Workshop on Open Source Software Engineering, International Conference on Software Engineering, Orlando, FL, USA, pp. 19-25
- Hars A., Ou S., (2001) “Working for Free – Motivations of Participating in Open Source Projects” In Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Volume 7, Hawaii, USA
- Michlmayr M., Hunt, F. & Probert D. (2005) "Quality Practices and Problems in Free Software Projects", In Proceedings of the First International Conference on Open Source Systems, Italy, pp. 24-28
- Nakakoji K., Yamamoto Y., Nishinaka Y., Kishida K., and Ye Y., (2002) "Evolution patterns of open-source software systems and communities", In Proceedings of the International Workshop on Principles of Software Evolution, International Conference on Software Engineering, Orlando, FL, USA, pp. 76 - 85

References

- Niemelä, E., Matinlassi, M., Taulavuori, A. 2004. Practical evaluation of software product family architectures. Third International Conference, SPLC 2004, Boston, MA,30 Aug. - 2 Sept. 2004. Lecture Notes in Computer Science 3154. Springer Verlang, ss. 130 - 145
- Raymond E.S. "The Cathedral and the Bazaar" (20.11.2005) URL: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/
- Yunwen Y., & Kishida K. (2003) "Toward an understanding of the Motivation of Open Source Software Developers", In Proceedings of the 25th International Conference on Software Engineering, Portland, OR, USA
- Yilmaz C., Porter A., Memon A., Krishna A. S., Schmidt D. C., & Gokhale A., (2006) "Techniques and Processes for Improving the Quality and Performance of Open-Source Software" In Software Process - Improvement and Practice Journal: Special Issue on Free/Open Source Software Processes, 2006.
- Zhao L., Elbaum, S. (2000) "A survey on quality related activities in open source" In ACM SIGSOFT Software Engineering Notes, Volume 25, May 2000, USA, pp. 54 - 57.