Component-Based Software Engineering: Modern Trends, Evolution and Perceived Architectural Risks

PhD thesis

Odd Petter Nord Slyngstad

Trondheim, 1st April 2011
Overview

• Introduction
• Research Questions RQ1-RQ4; from Design to Discussion
• Contributions
• Lessons Learned and Future Work
Introduction

• The SEVO (Software EVOlution) project (supported by Research Council of Norway under the ICT-2010 program).

• Focus on improvements in knowledge of, and methods to handle, software evolution; CBSE as overall umbrella.

• Research performed in the Norwegian and European IT-industry; focused study within Norwegian company StatoilHydro ASA.

• Prof. Reidar Conradi as project leader, 2 PhD students, 1 Post Doctoral position.
Motivation

- **Increased use** of Commercial Off-The-Shelf software (COTS) components and Open Source Software (OSS) (means changes in e.g. component selection, integration, testing, vendor control).

- **Risk Management** closely related to Software Evolution;
  - Awareness of architectural evolution risks since architectural changes can permeate a software system.
  - Focus in earlier studies commonly only on structured architecture evaluations, but methods used range quite widely [Babar 2007a].

- Need to improve integration of architectural activities, notations and artifacts into software processes and tools [Buchgeber 2008].
CBSE, COTS/OSS, and Risk

• **Component-Based Software Engineering (CBSE):** The process to define/implement/integrate/compose components into software systems [Sommerville 2010]. Emphasizes:
  – **Software reuse:** reuse of e.g. components, design, knowledge [vanVliet 2008].

• **Component:** unit of composition, specified such that the interfaces are separate from the implementation [Crnkovic 2002].

• **Commercial Off-The-Shelf software (COTS):** Pre-existing, commercially ready-to-use software [Oberndorf 1997]. Vendor control of requirements, release schedule, and evolution [Vidger 1996][Vidger 1997].

• **Open Source Software (OSS):** free use, adaptation, and improvement of original source code [OSI 1998-2010] [FSF 1985-2010].

• **Risk:** any issue that can affect a project adversely if not handled correctly [Boehm 1991].
Maintenance, Evolution and Architecture

- **Software maintenance**: work to keep already released software running and up-to-date, categorized as corrective, perfective, adaptive and preventive changes. Consumes 50%++ of the total software costs [Sommerville 2010].

- **Software Evolution**: under maintenance “umbrella” (non-corrective) [Sommerville 2010], or separate lifecycle step [Bennett 2000].

- **Software Architecture**: the structure(s) of the system, which comprise(s) software elements, the externally visible properties of those elements, and the relationships among them [Bass 2004].
Research Design

• Three main types of empirical studies: quantitative, qualitative, and mixed-method [Wohlin 2000][Creswell 2003] approaches;

  – **Quantitative**: Using one or more study objects in combination for e.g. theory testing, while attempting to minimize contextual effects (i.e. “noise”).

  – **Qualitative**: drawing information from a natural environment or social context [Denzin 1994] (i.e. context is essential and not “noise”).

  – **Mixed-method**: combination of quantitative and qualitative methods, which complement each other with respect to individual scopes, strengths, limitations and biases [Seaman 1999].

• 4 surveys, 2 case studies in this thesis; using the mixed-method approach towards gaining deeper insights, and triangulating data/methods.
Research Overview

Studies in StatoilHydro ASA

Phase 1: State-of-practice of SPI in CBSE
- RQ1, G1
- P1
- C1
- P2
- SP1 - SP7, SP10
2004 - 2008

Phase 2: Actual evolution; Software defects & changes
- RQ2, G1
- SP8, SP9
- P3
- C2
2005 - 2006

Phase 3: Risk Management
- Impact of one risk management strategy (Test Driven Development)
- RQ3, G2
- P4
- C3
2006

- Perceived architectural risks and employed strategies
- RQ4, G2
- P5
- P6
- C4a
- C4b
2006 - 2008

Figure labels:
- P1-P6 = Selected Articles,
- C1-C4 = Contributions,
- SP1-10 = Secondary Articles
- RQ1-4 = Research Questions
- G1-G2 = SEVO project goals

- Paper/Study
- Contribution
- Study Phase

NTNU
Norwegian University of Science and Technology

www.ntnu.no
SEVO goals vs. Research Questions (1)

- **G1.** Better understanding of software evolution, focusing on CBSE; investigating:
  
  - Increased usage of COTS/OSS; exploring e.g. vendor control, component selection and integration issues [Li 2004]. COTS/OSS development presents new risks, e.g. effort estimation [SP1].
  
  - **RQ1:** State of practices in CBSE for reusable components (articles P1, P2).
  
  - Software reuse in CBSE; enables e.g. quality, effort (cost) and time to market, and standards improvements [Sommerville 2010]. Reusable components have lower defect densities than non-reusable components across releases [Mohagheghi & Conradi 2004b].
  
  - **RQ2:** software defects and changes for individual, in-house reusable components (article P3).
SEVO goals vs. Research Questions (2)

• **G2.** Better methods to predict the risks, costs, and profile of software evolution in CBSE; investigating:
  
  – Test Driven Development (TDD); improving e.g. code comprehension, efficiency in defect detection. Prior research did not investigate TDD’s effect on non-corrective changes and on reusable components [Janzen 2005].

  – **RQ3:** impacts of TDD vs. test-last development for reusable components; defect and change density (article P4).

  – Perceived risks and risk management issues in software architecture evolution; little prior effort has been made.

  – **RQ4:** Perceived architectural risks and mitigation strategies from the viewpoint of software architects in industry (articles P5, P6).
Overview

- Introduction
- Research Questions RQ1-RQ4; from Design to Discussion
- Contributions
- Lessons Learned and Future Work
• **RQ1**: State of practices in CBSE for reusable components (articles P1, P2)

• **Design**: Survey at StatoilHydro ASA (P1) and survey in the European IT-industry (P2)

• **Context**: In-house: convenience sample at StatoilHydro ASA; all developers working with reuse. COTS/OSS: Convenience sample for pilot survey, expanded to stratified-random for larger survey (e.g. Census Bureau).
RQ1 – Findings and Discussion (P1, P2)

Findings:
• **In-house** components: defined/standardized **architecture** and **knowledge sharing** organization are important for reuse success.

• **COTS/OSS** components: cost factors and convenience/stability influence **development processes** to be **enriched** with COTS/OSS specific activities (e.g. for selection, integration).

Discussion:
• **Main positive impacts** agree with literature; **costs, development time, quality, standardized architecture** [Lim 1994]; **training** [Frakes 1995]; component **understanding** [Li 2004].

• Findings support literature on **importance** of **component-provider relationship** [Hauge 2010], and **influence** on component selection by **project-specific property constraints** [Hauge 2009a].
RQ2 – Design and Context

• **RQ2**: software defects and changes for individual, in-house reusable components (article P3)

• **Design**: “Data mining” case study at StatoilHydro ASA (ClearCase/ClearQuest tools). Defect density (#trouble reports/NSLOC) and change density (#change requests/NSLOC) as metrics.

• **Specific Context**: JEF framework of reusable components, mainly based on the Java and J2EE technologies; six of seven individual components (~20 KNSLOC) studied.
RQ2 – Findings and Discussion (P3)

Findings:
• **Decreasing defect density** over consecutive releases, for five of six components

• **Decreasing change density** for the reusable components;
  – **Increased maturity** as they are being adapted to accommodate new and altered requirements.
  – Improved reliability due to accumulated operational testing through usage.

Discussion:
• **Prior research:** reusable components were more stable (i.e. have lower densities of code modification) than non-reusable components over several releases [Mohagheghi & Conradi 2004b].

• **Our results:** larger components had higher defect and change densities in the first release, but this trend did not continue over consecutive releases.
RQ3 – Design and Context

• **RQ3**: impacts of TDD vs. test-last development for reusable components (article P4)

• **Design**: “Data mining” case study at StatoilHydro ASA (ClearCase/ClearQuest tools). Defect (#trouble reports/NSLOC) and change (#change requests/NSLOC) density as metrics.

• **Specific Context**: JEF-framework at StatoilHydro ASA; seven components (~20 KNSLOC) refactored to five components (~10KNSLOC) in the two latter releases developed with TDD.
RQ3 – Findings and Discussion (P4)

Findings: influencing factors include:
• Reduction in mean defect density, higher mean change density.
• Higher inherent maturity/reliability growth due to operational testing.
• Process impact in terms of tests developed and level of adaptation.
• Context factors should be explicitly considered when writing new test cases.

Discussion:
• Reduction in mean defect density (36%) similar to non-reusable components [George 2004] [Maximilien 2003] [Janzen 2005].

• [George 1994] indicate the importance of context. Our results: specific characteristics of reusable components as part of this context.

• Refactoring overhead seen as disadvantage [George 1994]. Our results: worthwhile towards future reuse and adaptability.
RQ4 – Design and Context

- **RQ4**: Perceived architectural risks and mitigation strategies from the viewpoint of software architects in industry (articles P5, P6)

- **Design**: Pilot survey, followed up by full-scale survey.

- **Context**: Norwegian IT-industry: convenience sample for pilot, expanded to snowball sampling variant.
RQ4 – Findings and Discussion (P5, P6)

Findings:

- **Larger number** of risks **identified during project planning** (than evolution).
- Risks: indicate **architects discover** e.g. design errors from **evaluations**.
- **Mitigation strategies express recovery** from missing evaluation output.
- Risks and mitigation strategies organized in **three-part operational matrix** (technical, process, organization).

Discussion:

- The **15 of 21** (pilot survey) and **16 of 19** (main survey) **most influential risks fit** risk categories identified in [Bass 2007], [Ropponen 2000].

- Corresponding **mitigation strategies** identified **appear not** to have been investigated earlier.
<table>
<thead>
<tr>
<th>Process (identified in planning), ID: Risk</th>
<th>Risk Influence</th>
<th>ID:Strategy:Outcome rating</th>
<th>PR 1: Lack of architecture documentation required more effort to be spent on planning during maintenance/evolution *</th>
<th>VH: 6, H: 25</th>
<th>PS 1</th>
<th>Recover needed architecture documentation using current architecture design and other artefacts as a basis</th>
<th>{0, 3, 2, 5, 1}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 2 Thorough planning before implementing maintenance/evolution changes</td>
<td></td>
<td></td>
<td>{0, 1, 8, 7, 1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 3 Recover architecture evaluation artefacts where needed</td>
<td></td>
<td></td>
<td>{0, 0, 4, 2, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 4 After process to capture important architecture details</td>
<td></td>
<td></td>
<td>{0, 0, 1, 3, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 5 Explicit training on architecture documentation</td>
<td></td>
<td></td>
<td>{0, 0, 1, 3, 0}</td>
<td></td>
</tr>
<tr>
<td>PR 2: Lack of architecture evaluation contributed to discovering potential problems later in planning of maintenance/evolution</td>
<td>VH: 5, H: 13</td>
<td>PS 1</td>
<td>Recover needed architecture documentation using current architecture design and other artefacts as a basis</td>
<td></td>
<td></td>
<td>{0, 0, 3, 4, 1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 3 Recover architecture evaluation artefacts where needed</td>
<td></td>
<td></td>
<td>{0, 1, 2, 4, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 4 After process to capture important architecture details</td>
<td></td>
<td></td>
<td>{0, 0, 2, 3, 0}</td>
<td></td>
</tr>
<tr>
<td>PR 3: Lack of business context analysis affected stakeholder relationships negatively</td>
<td>VH: 4, H: 13</td>
<td>PS 6</td>
<td>Integrate business context in planning of the maintenance/evolution</td>
<td></td>
<td></td>
<td>{0, 2, 5, 3, 1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 7 Include business context informally</td>
<td></td>
<td></td>
<td>{0, 1, 1, 4, 0}</td>
<td></td>
</tr>
<tr>
<td>PR 4: Insufficient requirements negotiation postponed important architecture decisions</td>
<td>VH: 4, H: 9</td>
<td>PS 8</td>
<td>Negotiate requirements early</td>
<td></td>
<td></td>
<td>{0, 0, 2, 2, 1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 9 More explicit communication</td>
<td></td>
<td></td>
<td>{0, 2, 3, 0, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 10 Allow additional time for communication and feedback</td>
<td></td>
<td></td>
<td>{0, 1, 1, 3, 0}</td>
<td></td>
</tr>
<tr>
<td>PR 5: Insufficient stakeholder communication contributed to insufficient requirements negotiation and affected implementation of new/changed architectural requirements negatively</td>
<td>VH: 7, H: 13</td>
<td>PS 13</td>
<td>Extra communication effort</td>
<td></td>
<td></td>
<td>{0, 1, 7, 3, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 14 Postpone some requirements to next maintenance/evolution cycle</td>
<td></td>
<td></td>
<td>{0, 0, 1, 2, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 15 Arrange plenary meetings for all stakeholders</td>
<td></td>
<td></td>
<td>{0, 0, 3, 4, 0}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 16 Negotiate project extension</td>
<td></td>
<td></td>
<td>{0, 1, 2, 2, 0}</td>
<td></td>
</tr>
<tr>
<td>PR 6: Poor integration of architecture changes into implementation process affected implementation process and the architecture design negatively *</td>
<td>VH: 2, H: 20</td>
<td>PS 17</td>
<td>Overlay architecture change process onto implementation of maintenance/evolution</td>
<td></td>
<td></td>
<td>{0, 0, 4, 7, 1}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PS 18 Integrate architecture considerations into implementation process</td>
<td></td>
<td></td>
<td>{0, 1, 9, 2, 0}</td>
<td></td>
</tr>
</tbody>
</table>

**Operational Matrix, part two:**

**Process level risks and strategies**
Validity Summary – Surveys (RQ1, RQ4)

- **Construct:** Well-connected research literature, RQs, and questionnaire questions. Pre-tested questionnaire. All terminology defined.

- **External:** P1, P5: Convenience sample, Norwegian IT-industry. P2: stratified random sample, Norway/Germany/Italy. P6: Constrained snowball sample.

- **Internal:** All respondents had required background and knowledge. Any ambiguities were directly clarified. Minor issues: linguistic errors and translation to national languages (P2).

- **Conclusion:** Relatively small sample size in P1, P5. Main survey P6 depends on background from pilot survey P5.
Validity Summary – Case studies (RQ2, RQ3)

• **Construct:** defect density and change density are well founded concepts. Data based on complete and stable releases.

• **External:** individual industrial systems at **one single company**; results relevant and valid for other releases of same components (others on a case-by-case basis).

• **Internal:** Records with missing data were excluded. Analysis was cross-checked to ensure compliance, consensus and correctness. **Potential** for classification and assignment **uncertainties at StatoilHydro ASA**.

• **Conclusion:** Relatively small, but complete sets of data to draw conclusions. Confounding factors (e.g. developer experience differences between teams) were not considered a threat; all development within same organizational unit.
Overview

• Introduction
• Research Questions RQ1-RQ4; from Design to Discussion
• Contributions
• Lessons Learned and Future Work
## Contributions, RQs and articles

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C1</strong>: Improved knowledge of modern trends in CBSE and their impacts on software development processes (RQ1)</td>
<td>P1, P2</td>
</tr>
<tr>
<td><strong>C2</strong>: Improved understanding of evolution impact on individual reusable components in terms of defect and change densities (RQ2)</td>
<td>P3</td>
</tr>
<tr>
<td><strong>C3</strong>: Improved understanding of the impact and effectiveness of TDD (RQ3)</td>
<td>P4</td>
</tr>
<tr>
<td><strong>C4a</strong>: Identification of perceived risks and related mitigation strategies specifically for the evolution of software architecture (RQ4)</td>
<td>P5, P6</td>
</tr>
<tr>
<td><strong>C4b</strong>: An adapted operational matrix for risk management in software architecture evolution (RQ4)</td>
<td>P6</td>
</tr>
</tbody>
</table>
Contributions to SEVO project goals

• **G1. Better understanding of software evolution, focusing on CBSE:** on process and component aspects towards a profiling of software evolution; contributions C1 and C2.

• **G2. Better methods to predict the risks, costs and profile of software evolution in CBSE:** Effectiveness of TDD – defects and changes; contribution C3. Risk and strategies in software architecture evolution; contribution C4a, and an adapted operational matrix tool; contribution C4b.

• **G3. Contributing to a national competence based around these themes and G4. Disseminating and exchanging the knowledge gained:** Publications in refereed international conferences and journals, reported to the FRIDA national database of research results. SEVO results integrated in NTNU courses and arranged workshops. All contributions C1-C4b.
Contributions to general practitioners and to StatoilHydro ASA

*Improvements of components and processes for handling of software evolution:*

- Integration of management tools needed for proper data collection and analysis.
- Improved commitment required to properly report data on e.g. defects, changes.
- Reuse training and knowledge sharing: Organization, scheduling and active participation should be improved.

*Improvements in risk management of software architecture:*

- Software architecture evaluation: implemented as a complete end-to-end process.
- Need for improvements in implementing risk management strategies.
Overview

• Introduction
• Research Questions RQ1-RQ4; from Design to Discussion
• Contributions
• Lessons Learned and Future Work
Lessons Learned for Researchers

- The proposed operational matrix tool can serve as a base to incorporate future insight.

- Defect density and change density give only a limited view of software evolution, e.g. component complexity is not considered.
Lessons Learned for Practitioners

• *Standardized architecture*
  – Keep architectural design and assets as “live” artifacts for dynamic updates
  – Ensure long-lasting commitment to empirical studies

• Encourage *knowledge sharing* to counter frequent changes of personnel, requirements, and resources; *training* as important precursor.

• *TDD* can promote reuse; higher inherent focus on testing and refactoring.

• *Risk management*: need to improve architectural documentation artifacts, and evaluate the architecture on a regular basis.
Future Work

Impact of defects and changes in software evolution (G1):
• Costs/benefits of using COTS/OSS software
• Improving metrics for code level complexity

Impact of modern trends in CBSE on the development process (G2):
• Knowledge management and sharing
• Efficiency of COTS/OSS and communities; benefits vs. disadvantages
• Proposed improvements and their effects, e.g. at StatoilHydro ASA
• Divergent perspectives in software architecture and agile development
• Approach to TDD at StatoilHydro ASA

Architectural risks in software evolution (G2):
• Code-level and other artifact data
• Expanding the operational matrix tool further
• Architecture evaluation methods
• Architectural versus non-architectural changes
Acknowledgements

- Thanks go to SEVO (RCN contract number 159916/V30), and NTNU for financial support.

- Thanks to my supervisor, Prof. Reidar Conradi, and colleagues on the SEVO project and at IDI, especially Dr. Jingyue Li, Dr. Anita Gupta, Prof. Letizia Jaccheri and Prof. Tor Stålhane.

- Thanks to international colleagues Dr. Bunse, Dr. Morisio, Dr. Torchiano, Dr. Babar, and Dr. van Vliet for collaboration on the surveys.

- Thanks to StatoilHydro ASAs Harald Rønneberg, Einar Landre and Harald Wesenberg for encouragement during the studies and permission to publish the case study results.