Software Architecture in Distributed Software Development


Universiteit Leiden
The Netherlands
Werner Heijstek

- Sinds 2007 promovendus aan het LIACS obv. Dr. Michel Chaudron (Software Engineering Group)

- Onderzoeksinteresses
  - gedistribueerde softwareontwikkeling
  - software architectuur en design (als product én als proces)
  - modelgedreven softwareontwikkeling
  - software productiviteit

- Samenwerkingen met verschillende (internationale) industriële partners en universiteiten
Software Architecture in Distributed Software Development

Introduction

Outline

Theory

Practice

Cases Studies

Expert Interviews

Recommendations
Software Architecture in Distributed Software Development
Theory

What is Software Architecture?

Definition

“The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.”

Architecture addresses and guarantees non-functional requirements like security maintainability, extendability and portability.
The system described in this diagram provides support for creating new mortgages and alteration of existing mortgages.

The design aims to separate the complexities of the business logic from the Financial Application Frontend by bundling all mortgage-related services on a central Mid Office System. This system provides services for the setup of all ‘mortgage actions’.

The Front Office Component hosts a Financial Application Frontend which contains a Mortgage-specific Application Component. Due to concerns regarding decreased Back Office availability, mortgage action requests may have a maximum size of 300 kilobytes.

The Mortgage Webservice provides an additional method to update mortgage attributes. This service only connects to an interface provided by the Mortgage Attribute Update specialization.
Global Software Development

- Also: outsourcing, offshoring or distributed SD
- Many motivations (cost, skill shortage, development speed)
- India is a (very) common offshore destination
- Introduces three “distances”
- Fails more often than co-located software development
Two strategies exist to transfer knowledge:

- Personalisation
- Codification

A mix or **Hybrid** strategy is commonly used.

Common GSD approach is “Transfer by Development Stage”

In GSD, codification is thought to be more dominant
RUP’s UML-centric Software Architecture Document (SAD) is commonly used in practice.

In a set of 59 industrial SADs we found that:
- Less than 45% of diagrams is UML.
- Non-UML diagrams mostly lack legends.
- Great variation exists in the ratio text-to-diagram.
- This is not different for GSD SADs.

Diagram-dense SADs are not better understood.

Linguistic distance from English greatly impacts SAD understanding.
Research Objective

Architecture matters but appears to be hindered by GSD.

- How is software architecture design and dissemination organized?
- What is the role of the architect(s) during the software development life cycle?
- How is software architecture documentation used?
- How is architecture compliance organized?

By means of three case studies and additional expert interviews
# Case A & B Characteristics

<table>
<thead>
<tr>
<th>Case A</th>
<th>Case B</th>
</tr>
</thead>
<tbody>
<tr>
<td>funct. size</td>
<td>34 use cases</td>
</tr>
<tr>
<td>pl. duration</td>
<td>10 months</td>
</tr>
<tr>
<td>methodology</td>
<td>RUP</td>
</tr>
<tr>
<td>budget</td>
<td>€800,000</td>
</tr>
<tr>
<td>technology</td>
<td>.Net</td>
</tr>
<tr>
<td>project objective</td>
<td>Expansion of an existing system for indexing and making searchable information.</td>
</tr>
<tr>
<td>interviewed onshore</td>
<td>Senior Mgr, Architect, Project Mgr., Arch. Reviewer, Test Lead</td>
</tr>
<tr>
<td>interviewed offshore</td>
<td>Project Mgr. #1, #2, Architect, Developer #1, #2, #3, #4</td>
</tr>
</tbody>
</table>
General Observations

Case A

• The **strict deadline** was overly ambitious

• **No proof of concept** (POC) was built

• Requirements were changed **late** during the project.

Case B

• frequent **requirement changes**

• high degree of **employee turnover**

• The offshore development team was **unable to package for deployment**

*Note that these factors are not at all uncommon.*
• Offshore was to deliver the architecture but failed or refused.
• Architecture responsibilities moved onshore

• Architect did not travel offshore

• Developers often required extensive code examples

• A hierarchy of developers exists offshore:
  1. technical team lead (architect)
  2. senior developer
  3. junior developer
“Limited Front Office Availability”
“No Problem Here”
“Chinese Whispers”
SAD was an important document, used as the primary architecture communication vehicle, but

- Some (important) parts were **written in Dutch**
- Most diagrams were not UML and box-and-line diagrams lacked legends.

- Developers
  - found the SAD **irrelevant**
  - read only “**their parts**”
  - had **too little time** to read the SAD
  - were **denied** the SAD
developers find architecture “very important”. However:

- Developers were not very knowledgeable about the software architecture

- Onshore architecture reviewed half of all code (straining the project budget)

- Much code was reworked by the both onshore and offshore
## Case C Characteristics

<table>
<thead>
<tr>
<th>Case C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>funct. size</strong></td>
<td>800 funct. pts.</td>
</tr>
<tr>
<td><strong>pl. duration</strong></td>
<td>4 months</td>
</tr>
<tr>
<td><strong>methodology</strong></td>
<td>Agile / Scrum</td>
</tr>
<tr>
<td><strong>budget</strong></td>
<td>€400,000</td>
</tr>
<tr>
<td><strong>technology</strong></td>
<td>.Net</td>
</tr>
</tbody>
</table>

**objective**  
Rebuild of an existing application

**int. onshore**  
Senior Mgr, Project Mgr.

**int. offshore**  
Sr. Developer, Developer
Non-Model-Driven Development

Figure: A ‘normal’ software development process
Software Architecture in Distributed Software Development

Cases Studies

Model-Driven Development

Figure: A model-driven software development process
Case C Development Process

- A **proven, home grown MDD platform** was employed.

- This project was **less complex** than Cases A and B

- Most of the architecture was **generated**.

1. Weekly, a low-level design was made by the onshore
2. This design was conferred by means of a video-link.
3. The offshore team was required to summarize what they understood.
4. This was then reviewed and implemented
Findings of Case Studies

1. Dissemination of software architecture is not formalized (while this might benefit the development process)

2. The role of the architect in GSD is not clearly defined

3. The SAD is intended to be used extensively but
   - is of disputable quality
   - developers use the SAD sparingly if at all

4. Developers find architecture very important but they are mostly knowledgeable about “their own” component
Two Major Influences

1. Knowledge Gap
   Exists between the onshore and offshore location regarding
   • software architecture design
   • and its role during the software development life cycle.

2. Implementation Focus
   Prematurely forces projects into the construction phase.

Cost reduction
• drives offshore development
• forces more responsibilities towards offshore
• reduces opportunity for training
• leads to favoring implementation-related disciplines
Implications and Consequences

- an unclear and incomplete SAD
- software architects available less time
- less direct interaction
- incorrect / incomplete knowledge of the software architecture
- more architectural freedom exists
- developers make assumptions
- compliancy violations are more likely
- less code is reviewed
- leads to rework, delays and overrun
Software Architecture in Distributed Software Development

Recommendations

1. Include offshore team in SAD development
2. Implement only after architecture is stable
3. The SAD should be mature too
4. Architects should be available more and travel
5. Verify continuously
6. Keep the SAD current to battle turnover

Invest in architecture design, dissemination and coordination!
Questions?
Contact Informatie

Werner Heijstek

heijstek@liacs.nl
http://www.liacs.nl/~heijstek

Snellius building room 150
+31 (0) 71 – 527 7050

Skype: wernerheijstek