Using Microservices for Legacy Software Modernization

Holger Knoche
Wilhelm Hasselbring
Software Engineering Group, Kiel University
http://se.informatik.uni-kiel.de/

February 21th, 2019

Agenda

1. Software Migration and Modernization

2. Motivation for Migrating to Microservices

3. Our Migration Case Study

4. Summary, so far
Migration to SOA

Dublo Migration Pattern [Hasselbring et al. 2004, 2008]
Migration to the Cloud

Migration to Microservices

Monolithic Architecture

User Interface
Business Interface
Daten Interface

Microservices Architecture

Microservice UI

Microservice

Microservice

Microservice

Microservices Poster, 2019
Agenda

1. Software Migration and Modernization

2. Motivation for Migrating to Microservices

3. Our Migration Case Study

4. Summary, so far
Drivers for Microservice Adoption

Table 5: Drivers for microservice adoption in different industries

<table>
<thead>
<tr>
<th></th>
<th>Overall</th>
<th>Development / Consulting</th>
<th>Energy / Industry</th>
<th>Financial Services</th>
<th>Retail / E-Commerce</th>
<th>Other / Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Score (mean)</td>
<td>SD</td>
<td>Score (mean)</td>
<td>Score (mean)</td>
<td>Score (mean)</td>
<td>Score (mean)</td>
</tr>
<tr>
<td>High Scalability and Elasticity</td>
<td>2.14</td>
<td>0.72</td>
<td>2.19</td>
<td>2.18</td>
<td>2.15</td>
<td>2.67</td>
</tr>
<tr>
<td>High Maintainability</td>
<td>2.11</td>
<td>0.75</td>
<td>2.31</td>
<td>2.10</td>
<td>2.15</td>
<td>2.17</td>
</tr>
<tr>
<td>Short Time to Market</td>
<td>2.07</td>
<td>0.82</td>
<td>2.12</td>
<td>1.45</td>
<td>2.10</td>
<td>2.67</td>
</tr>
<tr>
<td>Enabler for CD and DevOps</td>
<td>1.61</td>
<td>0.89</td>
<td>1.69</td>
<td>1.27</td>
<td>1.70</td>
<td>1.83</td>
</tr>
<tr>
<td>Suitedness for Cloud and Docker</td>
<td>1.55</td>
<td>0.89</td>
<td>2.00</td>
<td>1.09</td>
<td>1.35</td>
<td>1.50</td>
</tr>
<tr>
<td>Organizational Improvement</td>
<td>1.37</td>
<td>0.81</td>
<td>1.31</td>
<td>1.18</td>
<td>1.50</td>
<td>1.83</td>
</tr>
<tr>
<td>Polyglot Programming</td>
<td>1.28</td>
<td>0.90</td>
<td>1.00</td>
<td>1.82</td>
<td>1.15</td>
<td>2.00</td>
</tr>
<tr>
<td>Polyglot Persistence</td>
<td>1.27</td>
<td>0.83</td>
<td>1.06</td>
<td>1.09</td>
<td>1.40</td>
<td>1.33</td>
</tr>
<tr>
<td>Attractiveness as Employer</td>
<td>0.87</td>
<td>0.86</td>
<td>1.00</td>
<td>0.55</td>
<td>0.85</td>
<td>1.67</td>
</tr>
</tbody>
</table>

Source: [Knoche & Hasselbring 2019]
See also: [Hasselbring 2016, 2018, Hasselbring & Steinacker 2017]
# Modernization Goals

<table>
<thead>
<tr>
<th>Overall</th>
<th>Development / Consulting</th>
<th>Energy / Industry</th>
<th>Financial Services</th>
<th>Retail / E-Commerce</th>
<th>Other / Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Score (mean)</strong></td>
<td><strong>Score (mean)</strong></td>
<td><strong>Score (mean)</strong></td>
<td><strong>Score (mean)</strong></td>
<td><strong>Score (mean)</strong></td>
<td><strong>Score (mean)</strong></td>
</tr>
<tr>
<td>Improve Maintainability</td>
<td>1.79</td>
<td>1.81</td>
<td>2.00</td>
<td>1.85</td>
<td>1.50</td>
</tr>
<tr>
<td>Improve Time to Market</td>
<td>1.54</td>
<td>1.88</td>
<td>1.09</td>
<td>1.60</td>
<td>1.83</td>
</tr>
<tr>
<td>Improve Scalability</td>
<td>1.46</td>
<td>1.44</td>
<td>1.64</td>
<td>1.60</td>
<td>1.50</td>
</tr>
<tr>
<td>Improve Quality</td>
<td>1.39</td>
<td>1.06</td>
<td>1.64</td>
<td>1.50</td>
<td>1.50</td>
</tr>
<tr>
<td>Prepare CD and DevOps</td>
<td>1.39</td>
<td>1.56</td>
<td>1.45</td>
<td>1.25</td>
<td>1.67</td>
</tr>
<tr>
<td>Introduce New Technology</td>
<td>1.04</td>
<td>0.88</td>
<td>1.09</td>
<td>0.90</td>
<td>1.50</td>
</tr>
<tr>
<td>Improve Team Motivation</td>
<td>0.97</td>
<td>0.94</td>
<td>0.82</td>
<td>1.10</td>
<td>1.50</td>
</tr>
</tbody>
</table>

Source: [Knoche & Hasselbring 2019]
Agenda

1. Software Migration and Modernization
2. Motivation for Migrating to Microservices
3. Our Migration Case Study
4. Summary, so far
Exemplar Legacy System

The exemplar legacy system...
• manages customer data of an insurance company
• was built in the 1970s and 1980s
• consists of ~1 million SLOC (COBOL)
• is part of a larger software ecosystem (COBOL and Java)
Why Modernize this System?

Primary drivers
• Lack of maintainability
• High time to market

Secondary drivers
• Vendor lock-in
• Programmer retirement
Architectural Modernization Goals

- Establish well-defined platform-independent interfaces
  - based on the bounded contexts of the underlying domain
- Reducing the number of entry points
- Eliminating redundant / obsolete parts of the application
- Incremental platform migration from COBOL to Java
Why Microservices to Achieve these Goals?

Microservices...

• provide strong component separation (including data)
• are independently deployable
• have independent lifecycles
• emphasize cross-platform interaction
• work well with CI / CD / DevOps
Migration Process:
Step 0: Initial Situation
Migration Process:
Step 1: Defining a Service Facade

Client A

Customer Application

Client B

Table TC1

Table TB1

Shared Database
Defining a Service Facade

• First, a target domain model was designed
  – and used to define service operations from scratch.
• Afterward, static analysis was employed to identify the “entry points” of the existing application
  – i.e., programs, methods, or database tables that were accessed from other applications.
• Then, similar or redundant operations were merged
Migration Process:
Step 2: Adapting the Service Facade
Migration Process:
Step 3: Client Migration

[Diagram showing the migration process involving Client A, Customer Application, Client B, and a shared database with tables TC1 and TB1.]
Migration Process:
Step 4: Internal Restructuring

See also:
Strangler Pattern [Newman 2015]
Migration Process: Step 5: Platform Migration
Current State

• What has been achieve so far?
  – The client migration has been decoupled from the backend migration.
    • Client migration is finished
  – First service operations have been migrated
  – New requirements were delivered on time
  – First legacy artifacts have been retired
  – Although the implementation is still based on the old COBOL code, it is now only accessed using well-defined, platform-independent interfaces.
  – In particular, the database has been decoupled such that, for instance, schema changes can now be performed without affecting client applications.

• Which challenges remain?
  – Database partitioning into (logical) bounded contexts
  – Transactions and Performance [Knoche 2016]
  – Batch jobs
Summary, for far

• A good modernization strategy delivers value even if some parts cannot be migrated
• Isolate risks, even if it means additional costs
• First careful steps toward infrastructure automation and DevOps practices,
  – as the new implementations create opportunities for experimenting with these approaches.
• It’s not only technology
  – It’s also about people
  – You always need a business case

Slides: http://eprints.uni-kiel.de/45795/
References


References


