Architectural Decision Making for Service-Based Platform Integration: A Qualitative Multi-Method Study

Ioanna Lytra, Stefan Sobernig, Uwe Zdun

Faculty of Computer Science
University of Vienna, Austria

Institute for IS and New Media
WU Vienna, Austria
Architectural Decisions in Service-based Platform Integration

Software Platform: a collection of software subsystems (e.g., communication middleware, databases) and interfaces which form an infrastructure for developing a set of related software applications.
Research Questions

- What are the recurring architectural design decisions on service-based platform integration documented by existing software patterns and pattern collections?

- What are the levels of decision making when designing an architecture for service-based platform integration?
**Virtual Service Platform:** handles various integration aspects like interface adaptation between platforms, integration of service-based and non-service-based solutions, routing, enriching, aggregation, splitting of messages and events.

- **53 services to be integrated**
- **53 x n Design Decisions**
- **service adapters/proxies/facades**
Research Design

Systematic Literature Survey → Pattern Language (PL) → Architectural Decision Model (ADM) → Preparation for qualitative study

3 iterations:
- Interview
  - coding
  - summarizing
  - distill
- Coded Interview
  - summarizing
- Field Memos
  - distill
- Improved PL/ADM

Case Study:
- Design Decisions
- Architecture Design
- Design Levels
- Staging of Platform Integration Decisions

Distill: coding, summarizing, distill
Generalizing: generalizing

Quality Assessment: reviewed patterns, referenced in SOA technical domain

Pattern Extraction and Synthesis: 29 patterns selected + 11 patterns referenced
Interviews

- 3 interviews
- 9 experts
- 3 companies – 3 platforms

**Interview Instrument** *(4 categories, 29 questions)*

<table>
<thead>
<tr>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptation and Integration</td>
<td></td>
</tr>
<tr>
<td>1.1 Can the services from the source platform be directly used in the VSP platform?</td>
<td>closed</td>
</tr>
<tr>
<td>Interface Design</td>
<td></td>
</tr>
<tr>
<td>2.1 Have the platform services been exposed as services using standard interfaces/technologies?</td>
<td>closed</td>
</tr>
<tr>
<td>Communication Style</td>
<td></td>
</tr>
<tr>
<td>3.1 How important is performance for the connection?</td>
<td>open</td>
</tr>
</tbody>
</table>
**Case Study Design**

**confirmatory:**
to which extent our pattern language and our decision model correspond to the platform integration domain?

**exploratory:**
how is the decision making in platform integration being performed?
Relations between ADDs and Patterns

Integration with in/compatible interfaces?
Local or remote connection?
## ADD Model

- Adaptation and Integration Patterns (6)
- Interface Design (6)
- Communication Style (8)
- Communication Flow (9)

<table>
<thead>
<tr>
<th>Decision Point</th>
<th>Options and Patterns Dependencies</th>
</tr>
</thead>
</table>
| D1 – Which kind of component will be used for integrating the platform service into the service-based integration platform? | - None (direct calls from application to platform)  
- Integration component with same interface (select pattern PROXY or a PROXY variant)  
- Integration component with a different interface (select pattern ADAPTER or an ADAPTER variant) |
| D2 – Is the connection between platform and service-based integration platform a local or a remote connection? | - Local (Select local variant of PROXY or ADAPTER, as selected in other decisions)  
- Remote (Select remote variant of PROXY or ADAPTER, as selected in other decisions) |
Exemplary Levels and Stages of Decision Making

**Level-1 (architecture): integration architect**
- CommStyle
  - synchronous
  - asynchronous
  - (sync.) RPC

**Level-2 (platform): platform supplier**
- WCF
  - Channels
  - Invokers
  - AsyncPattern
  - Events

**Level-3 (integration): system integrator**
- VSPComm
  - ActiveMQ
  - ApacheCXF

**Level-4 (application): application engineer**
- AndroidComm
  - RabbitMQ
  - AIDL IPC/RPC
  - KSoap2

**Stage-0**
- implies
- derived from

**Stage-1**
- implies

**Stage-2**
- implies

**Stage-3**
- implies

AndroidComm
RabbitMQ
AIDL IPC/RPC
KSoap2

VSPComm
ActiveMQ
ApacheCXF

Producer
Session
MessageListener
Consumer

Consumer
OnReceiveMessageHandler

ActiveMQConnection
Producer
Session
MessageListener
Consumer

Request-Reply
One-Way
Request-Acknowledge

ActiveMQ
ActiveMQConnection
Producer
Session
MessageListener
Consumer

AndroidComm
RabbitMQ
AIDL IPC/RPC
KSoap2

VSPComm
ActiveMQ
ApacheCXF

Producer
Session
MessageListener
Consumer

Consumer
OnReceiveMessageHandler

ActiveMQConnection
Producer
Session
MessageListener
Consumer

AndroidComm
RabbitMQ
AIDL IPC/RPC
KSoap2

VSPComm
ActiveMQ
ApacheCXF

Producer
Session
MessageListener
Consumer

Consumer
OnReceiveMessageHandler
Example for 2 Levels and 2 Stages

Level-1 (architecture): integration architect

- CommStyle
  - synchronous
  - asynchronous
  - (sync.) RPC

Stage-0

- derived from

Level-2 (platform): platform supplier

- WCF
  - Channels
  - Invokers
  - AsyncPattern
  - Events

Stage-1

- derived from

- implies

- async. RPC
  - Messaging

- ...
Artifacts in Levels of Decision Making

1. Pattern-Based Architectural Decision Model
2. Design & Architecture of Platform P
3. Platform P: Service Wrapper Layer
4. Generic Parts of the Service-based Integration Platform Architecture SP
5. Architecture of a Domain-Specific, Service-based Integration Platform $SP_{AP}$

Derives from:

Domain-Specific Application Architecture $AP$ Based on Integration Platform $SP_{AP}$
Limitations and Threats

- **Systematic Literature Review**: completeness, authors’ bias
- **Interviews**: external and internal validity
- **Generalizability**: small sample but broad domain
Lessons Learned

1. Using software patterns facilitates iterative decision making.
2. Patterns are an important communication vehicle between interviewers and interviewees with different backgrounds.
3. Our research design should not impose design decisions onto the subjects.
4. The architecting process should be observed in the context of a real development project.
Conclusions

- **Architectural Decision Model**
  - Refine with further qualitative studies
  - Assess its cost-benefit balance

- **Decision stages and levels**
  - Tool support
  - How do they apply to other platform-like software development approaches?
Thank you for your attention!