Requirements Reconciliation for Scalable and Secure Microservice (De)composition

Authors:
Mohsen Ahmadvand
Amjad Ibrahim

Affiliation:
Prof. Dr. Alexander Pretschner
Technische Universität München
Fakultät für Informatik
Lehrstuhl XXII für Software Engineering
Context (Scalability)

- What is a monolithic service?

What if the majority of the users only need to use M1?
Context (Satisfaction and productivity)

- Large code base
- Highly coupled
- Changes are expensive
- Conflict handling
- Testing
- Prolonged release cycles

Customers satisfaction? Team productivity? 😞 😞
Context (Microservice)

Split a service into a set of smaller services that satisfy the following criteria:

- End-to-end task-oriented
- Covers a single business use case
- Self-contained (presentation, business, data)
- Light weighted (a couple of hundred lines of code)
- Autonomous (can be deployed individually)
- Loosely coupled

Scalability and shortened release cycles
Microservice vs. Monolithic Service
Problems with microservice-based architecture

**System architecture**

- new system:
- organization:

- MS1
- MS2
- MS3
- MS4

Scalable? System-wide view?

**System security**

- Service
  - UI
  - M1
  - M2
  - M3
  - M4

- API Gateway

Optimal security?
How to optimally (security & scalability) split a monolithic system into microservices?
Idea/ contribution

- Requirements
  - Requirement 1
  - Requirement 2
  - ...
  - Requirement N

- Reconciliation Methodology
- Requirement assessment
- Candidate recommender

- MS1
- MS2
- MS3
- MS4
Benefits

- Requirements at early stage
- Informed architectural decisions
- Decomposition recommendation
- Enhanced security compliance
- Better performance
Requirements assessment

Functional requirements

Security requirements

Scalability requirements

Dependency

Dependency weight
Example

Stream movie

Billing
One-to-one

Access control
Encryption
Anonymization

The anticipated number of simultaneous users?
Reconciliation methodology

Start

Functional req.

False

Scalab. > L

True

Standalone MS

Functional req.

False

Merge

List of MS candidates

True

Sec. AND Weight > L

Loop

False

Dependency

After last Dependency
Methodology on a toy system

• Online Movie Streaming System

• Service (SaaS) platform to stream movies and TV shows

• Designed as a monolithic service
  • Over 1 million users simultaneously.
  • Amazon Cloud Services

• Moving to a pay-as-you-go model
Use/Misuse cases

User

- Stream movie/show
- Pay the bill
- Register
- Search in the catalog

Profile user

- Steal user data
- Stream without payment
- Inject malicious content

Crook
## Summary of system dependencies

<table>
<thead>
<tr>
<th>Requirement</th>
<th>User Profiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Registration</td>
<td></td>
</tr>
<tr>
<td>User Profiler</td>
<td></td>
</tr>
<tr>
<td>Billing</td>
<td>H</td>
</tr>
<tr>
<td>Streamer</td>
<td>L</td>
</tr>
<tr>
<td>Search Engine</td>
<td></td>
</tr>
</tbody>
</table>
Methodology on a toy system

- User registration
  - No dependency
  - It is required to scale to respond to
    → Added as a microservice candidate

- User Profiler
  - Can be used to violate the users privacy
  - The impact of enforcing the security policy would be high
  - Billing is a dependency
    → Merge both and add as a candidate
Contribution

• How to decompose a system into an optimal set of microservices
  • Layered architecture style [2]
  • From a monolithic system towards a microservice-based architecture [1]
  • A group of microservice per development team

  → The conflict among requirements at the system level!
Contribution (continued)

- Another related term: service composition [3]
- Create new services by utilizing existing services.
- Service discovery in requirements elicitation phase [4]–[6]
- These techniques are all based on matching existing services to the requirements
- This work is based on mapping requirements to services.
Conclusion

- Methodology to break monolithic systems into microservice candidates
  - Optimal security and scalability trade off.

- Requirements engineers can mindfully reconcile security and scalability requirements
  - Architectural decisions at requirements engineering phase

- Security and scalability are not the only factors
  - Other factors such as cost, maintainability and structure of development teams

- Future work should explore possible evaluation methods for our methodology.
  - Empirical evaluation with many industrial use cases.
Contact us!

Mohsen

ahmadvan@cs.tum.edu

Amjad

ibrahim@cs.tum.edu
References


