Software Architecture in Industry

Erwann Wernli
Software Architect
erwann.wernli@sbb.ch
Agenda

- What is Architecture?
- Architecture in Practice: TMS
- Documenting Software Architecture
- Trade-off in Software Architecture
Software Architecture from Textbooks.

**Software architecture** refers the fundamental structures of a software system and the discipline of creating such structures and systems.

Each structure comprises software elements, relations among them, and properties of both elements and relations.

https://en.wikipedia.org/wiki/Software_architecture
Architecture: Abstraction levels and Perspectives
Architecture: Abstraction levels and Perspectives

This lecture

Solution

System

Application

Module

Components

Modularity

Integration

Technology

GUI

Security

...
Example: TMS
TMS – Value Stream and Organisation

Value Stream

E Kapazität managen und Kundeninformation Bereitstellen

E1: Netzzugang und Netzanschluss managen
E2: Netznutzung konzipieren
E3: Netznutzung planen

F: Kapazitätsplan umsetzen und Kundeninformation sicherstellen

F1: Bahnbetrieb durchführen
F2: Abweichungen managen
F3: Kundeninformieren
TMS – Value Stream and Organisation

Value Stream

Agile Release Trains (ART)

ART Kapazitätsmanagement
- TMS-SHOP
- TMS-CapacityPlaner

ART Kapazitätssteuerung
- TMS-CapacityOptimizer
- TMS-OperationalDataGateway
- TMS-IncidentManager

ART Kapazitätslenkung
- TMS-ItlisAdapter
- TMS-ATO-Execution

ART Analytics und DataHub
- TMS-Analytics

ART Topologie Trasse
- TMS-Topo
TMS - Systems

Vision 2028
TMS - Architecture Principles (samples)

Integration
- No shared storage between TMS-Systems
- Asynchronous Communication between TMS-Systems
- Interfaces are idempotent

GUI
- TMS-Systems expose Microfrontend
- Microfrontends integrates in the TMS-Workbench

Technology
- TMS-System leverage the TMS-Technology Stacks

Operations
- Code and Configuration are kept separate
High-available
Low latency / high throughput
(Some safety relevance)

«Office Time» Availability
Medium Latency/Throughput

Low Availability
High Latency / High Throughput
## TMS – Technology Stacks (samples)

<table>
<thead>
<tr>
<th></th>
<th>Analytics</th>
<th>Enterprise</th>
<th>Realtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Interchange</td>
<td>Solace Kafka</td>
<td>Solace Kafka</td>
<td>Solace</td>
</tr>
<tr>
<td>Data Store</td>
<td>Oracle S3</td>
<td>Oracle Hazelcast</td>
<td>Oracle Hazelcast</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neo4j S3</td>
<td>Neo4j S3</td>
</tr>
<tr>
<td>Application Framework</td>
<td>SpringBoot Hadoop</td>
<td>SpringBoot</td>
<td>SpringBoot</td>
</tr>
<tr>
<td>GUI</td>
<td>Angular</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flutter or Native</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td>Splunk</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenTracing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prometheus</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Grafana</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Engineering</td>
<td>Java, Typescript, Kotlin (Mobile)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td>OpenShift</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Linux, iOS, Android</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Architecture Patterns – Microfrontend

We decompose the GUI into microfrontends that integrate in the TMS Workbench

Pattern

Pro:
- No monolithic Frontend
- More team autonomy

Contra:
- More complexity in the integration
Architecture Patterns – High Availability

High available systems must be redundant

**Pattern «FT»**

**Pro:**
- Works with state

**Contra:**
- State Management more complex

**Pattern «DQ»**

**Pro:**
- Easier
- Improve performance too

**Contra:**
- Requires process to be stateless
Architecture Patterns – Decentral Communication

We rely on decentral communication to reduce network footprint

Pattern «Central Communication»
Pro:
• Easy
Contra:
• Broker is SOF
• High Network Footprint

Pattern «Decentral Communication»
Pro:
• Reduced Network Footprint
• Less SOF
Contra:
• Operational Overhead
TMS – Steering the Migration/Development

**Programmmanagement**
- Fachl. Roadmap
- ARUs

**Anwendungsarchitektur**
- TMS Systemarch.
- Gap-Analyse
- Migrationsstrategie

**Technologiearchitektur**
- IT Strategie
- TMS Techstack
- Gap-Analyse
- Migrationsstrategie

**Prinzipien**

**Zielbild 2018**
- «Blache»
- Work-packages

**Zielbild 2020**
- «Blache»
- Work-packages

**Zielbild 2022**
- «Blache»
- Work-packages

**Zielbild 2024**
- «Blache»
- Work-packages

**Ausblick 2026/28**
- ....
- ....
- ...

**Motive**

**Migrationsplan**
Documenting Software Architecture
Software Architecture from Textbooks

Presentation Layer
- Customer Screen
  - Customer Delegate

Business Layer
- Customer Object

Persistence Layer
- Customer dao
  - Order dao

Database Layer
- Database
Software Architecture in Practice

→ Diagrams are a mix of various concerns
Software Architecture in Practice

→ Diagrams are a mix of various concerns
Software Architecture in Practice

→ Diagrams are a mix of various concerns

Functional aspects

Technical aspects
Documenting Software Architecture

→ There exist various template to help documenting software architecture
  - RUP
  - C4
  - Arc42
  - ...

→ They help identify the level of abstractions and perspectives.
Architecture and Trade off
Architecture is about tradeoff

1. There are many ways to decompose a system functionally
2. There are many technologies to choose from
3. There are many NFR that comes into play
Example: TMS

- Availability
- Performance
- Maintainability

Software Architecture

- Shop, Planner, Optimizer, etc.
- Microfrontend
- FT vs DQ Services

Messaging Leader Election
Product NFR
Customer NFR
Functional Decomposition

- Self-contained services
- Microservices
- Monolith
- Layers
- Pipe and Filter
- Workflow
- Event-Driven
- Actors
- Microfrontend
- Fat Client
- Thin Client
- Backend for Frontend
- RESTful Services
- Orchestration
- Choreography
- Pub/Sub
- RPC
- Event Sourcing
- 3-Tier Architecture
- …
Technologies

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

And many many more…
Tradeoff = Fit for Purpose

The simple house example also highlights another important property of architectures: rarely is an architecture simply "good" or "bad". Rather, architecture tends to be fit or unfit for purpose.

https://www.enterpriseintegrationpatterns.com/ramblings/86_isthisarchitecture.html