UNTERNEHMENS INFORMATIONSSYSTEME

MICROSERVICES
DEFINITION, ARCHITEKTUR, ANWENDUNG

Dr.rer.soc.oec. Maxim Vidgof, MSc BSc, Workflow Systems and Technology, Fakultät für Informatik
Folien mit freundlicher Genehmigung von emer. o. Univ.-Prof. Dr.techn Dimitris Karagiannis, Mag. Dr. Wilfrid Utz und Alexander Völz, B.Sc. MSc

Microservice Foundations

1. Was sind Microservices?
2. Monolith vs. Microservice
3. Charakteristika von Microservices
4. Microservices Pattern
5. Vor-/Nachteile von Microservices

Was sind Microservices?

"In short, the microservice architectural style is an approach to developing a single application as a **suite of small services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery. There is a **bare minimum of centralized management** of these services, which may be written in different programming languages and use different data storage technologies."

Martin Fowler & James Lewis

Q: http://martinfowler.com/microservices/

Service Oriented Architecture (SOA) vs. Microservices

- SOA sehr weiter Begriff
  - Experten verstehen unterschiedliche Dinge darunter
- Microservices als Teilmenge von SOA
  - „probably service orientation done right“[1] – Fowler
  - „fine grained SOA“[2] – Netflix
- Microservice-Ansatz bereits seit ~10 Jahren im Umlauf unter dem Begriff SOA (als Teilgebiet dessen)

Q: http://martinfowler.com/bliki/ServiceOrientedAmbiguity.html
[1] http://martinfowler.com/articles/microservices.html#MicroservicesAndSoa
[2] https://twitter.com/adrianco/status/542850261782237184

Microservice-Architekturen in der Praxis

Beispiel Netflix

- Ehemals Monolith für DVD-Verleih, nun Architektur mit hunderten Microservices für Netflix‘ Streamingservice
- Skalierbarkeit enorm wichtig
  - „Peak hours“
- Eigene Tools für Performance- und Zuverlässigkeitsmessungen
  - Teilweise Open Source

Q: https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/
http://techblog.netflix.com/2015/02/a-microscope-on-microservices.html

Was sind Microservices (II)

- Trend hin zu Microservices in den vergangenen Jahren
- Idee: einzelne Services voneinander trennen, um so flexibler zu sein
  - Je Service unterschiedliche Technology-Stacks
    - > Verschiedene Programmiersprachen
    - > Verschiedene Datenbanken (MySQL, Cassandra..)
    - > Verschiedene Frameworks (z.B. User Interface)
  - Services können getauscht werden, ohne Auswirkungen auf gesamtes System/andere Services
  - Individuelles Service Deployment

Q: http://martinfowler.com/articles/microservices.html

Was sind Microservices (III)

- Klein und konzentriert
- „Doing one thing well“
  - Robert Martin: Single Responsibility Principle
    - „Gather together those things that change for the same reasons, and separate those things that change for different reasons.“
- Autonom
  - Separate Entitäten
  - Deployed auf separaten (virtuellen) Maschinen
  - Gesamte Service-Kommunikation wird durch Netzwerkaufrufe realisiert
  - Services bieten Funktionalität nach Außen nur über Interfaces an
  - Key question: „Can you make a change to a service and deploy it by itself without changing anything else?“

Q: Sam Newman. Bulding Microservices, O‘Reilly

Microservices: Key Benefits

- Technology Heterogeneity
  - Each service may implement the most appropriate technology stack (UI, Logic, DB)
- Resilience
  - Total service failure is handled functionality degraded accordingly
- Scaling
  - Scaling can be decided and realized on a service-specific basis
- Ease of Deployment
  - Service-individual deployment enables immediate deployment of fixes/features
- Organizational Alignment
  - Small and efficient teams for each service
- Composability
  - Compose different services for different purposes/devices etc..
- Optimizing for Replaceability
  - In case it gets obsolete (technically) or unnecessary (economically), or both

Q: Sam Newman. Bulding Microservices, O'Reilly

Monolith vs. Microservices

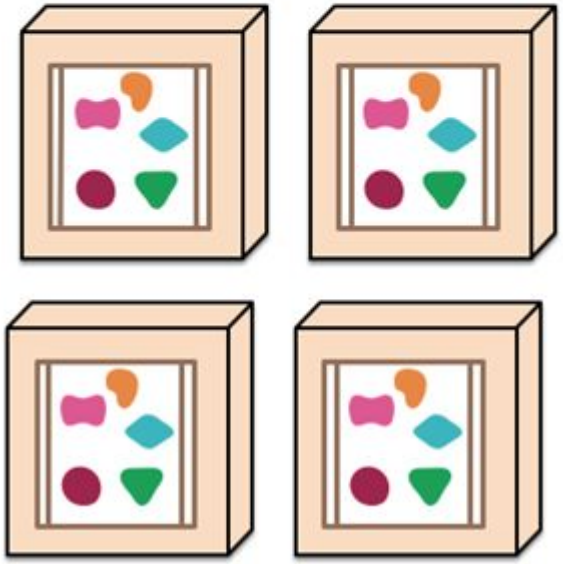
Monolith	Microservices
Einzelne Anwendung	Eine Anwendung = mehrere Services
Oft geteilt in 4 Komponenten: - User Interface - Client-side logic - Server-side logic - Database	Jedes Services läuft als eigener Prozess, keine klare Komponentenstruktur
Skalierung durch vollständige Instanzen	Service-spezifische Skalierung möglich
Zunehmend frustrierender mit Deployment in Cloud	Cloud- und Internet-basierte Technologien als Technologischer Backbone
Meist eine Programmiersprache / 1 Technology Stack	Service-spezifische Technologie Stacks erlauben maximale Freiheit

Q: http://martinfowler.com/microservices/

Monolith vs. Microservices

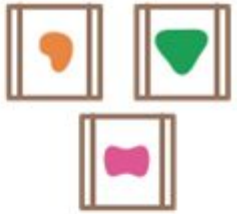
A monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers

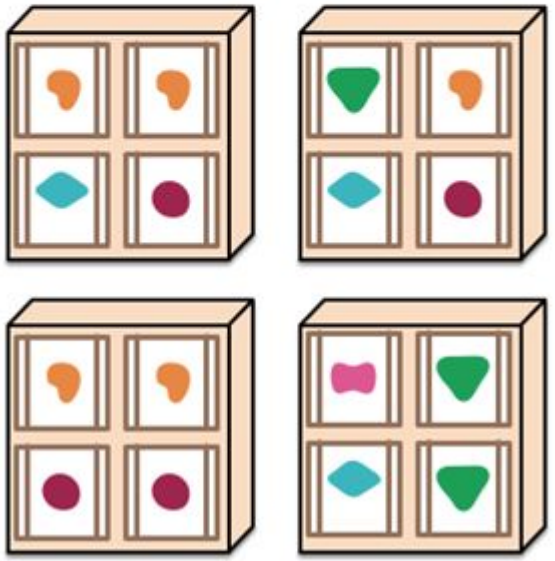


Monolith

A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



Microservice

Q: http://martinfowler.com/articles/microservices.html

Charakteristika von Microservices

- Keine formale Definition vorhanden -> **9** distinkte wiederkehrende Charakteristika in Microservice-Architekturen:

1. Aufteilung der Komponenten als Services
2. Organisation rund um Geschäftsressourcen
   - „Versand, Bestellungen, Katalog“ statt „UI, Server, Datenbanken“
   - Cross-functional Teams: decken alle nötigen Skills von UX, über Datenbanken bis Projektmanagement ab
   - „as full stack as possible“[1] – direkte Kommunikation eines Entwicklerteams mit Endkunden

Q: http://martinfowler.com/articles/microservices.html
[1] https://youtu.be/wgdBVIX9ifA

Charakteristika von Microservices (II)

3. Produkte, nicht Projekte
   - Team „besitzt“ ein Produkt über gesamte Lebensspanne
   - Normale Vorgehensweise: Softwareprojekt gilt als abgeschlossen, Projektteam wird zurückgezogen und Maintenance Team übernimmt
4. Intelligente Endpunkte, „dumme“ Übertragungskanäle
   - favorisierte Alternative zu Enterprise Service Bus (ESB)
   - Microservices sind decoupled, beinhalten deshalb ihre Logik
   - HTTP; lightweight messaging; REST-ähnlich
   - Gutes, effizientes Piping nötig – vgl. Internet

Q: http://martinfowler.com/articles/microservices.html

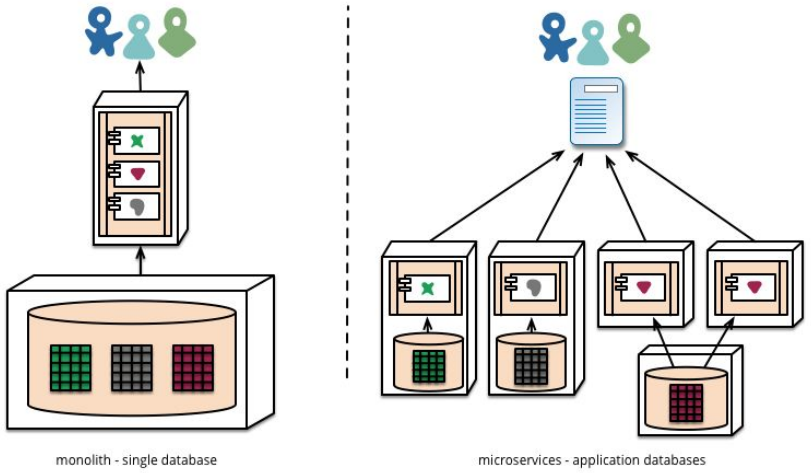
Charakteristika von Microservices (III)

5. Dezentralisierte Governance

- Wahl des richtigen Tools für den richtigen Zweck
  - > möglich durch Komponenten Architektur
- Keine Standardisierung, keine strikt vorgegebenen Methoden

6. Dezentralisiertes Datenmanagement

- In großen Systemen verschiedene Ansichten der Daten
  - > Problem bei Monolith, um alles unter einen Hut zu bekommen
- Möglichkeit verschiedene Datenbanken
  - bei verschiedenen Services zu benutzen
    - > Vgl. Amazon: Service selbst
    - für Daten und Persistenz
    - verantwortlich



Q: http://martinfowler.com/articles/microservices.html#DecentralizedDataManagement

Charakteristika von Microservices (IV)

7. Automatisierung der Infrastruktur

- Automatische Tests der Software
- Automatisches Deployment

8. Design for Failure

- dauerhaftes Monitoring
- rasches (automatisches) wiederherstellen bei Fehler
- Bsp. Netflix: Chaos Monkey „zerstört“ automatisch Nodes, um das System während Ausführung zu testen

9. Evolutionary Design / Continuous Delivery

- Neue Features und Fähigkeiten durch Austauschbarkeit einzelner Services

Q: http://martinfowler.com/articles/microservices.html

Microservices Pattern

- Kontext:
  - Serverseitige Enterprise Anwendung
  - Unterschiedliche Clients
  - Möglicherweise API, Webservices oder Message Broker
- Problem:
  - Welche Deployment-Architektur?
- Einflüsse:
  - Team von Entwicklern (neue Mitglieder müssen schnell produktiv sein)
  - Anwendung soll einfach zu verstehen und zu ändern sein
  - Wiederholendes Deployment
  - Mehrere Kopien auf mehreren Maschinen, um skalierbar und verfügbar zu sein

Q: http://microservices.io/patterns/microservices.html

Vor- und Nachteile von Microservices

Vorteile
Abgrenzung zwischen Modulen
Unabhängiges Deployment einzelner Module
Technologische Diversität
Kleine Services
Bessere Fehlerisolation
Einfachere Skalierung

Nachteil
Gesteigerte Komplexität
- Test - Neuartiges Konzept für Beteiligte - Kommunikationsmechanismus
Operationale Komplexität
Konsistenz über gesamtes System

Q: http://martinfowler.com/articles/microservice-trade-offs.html

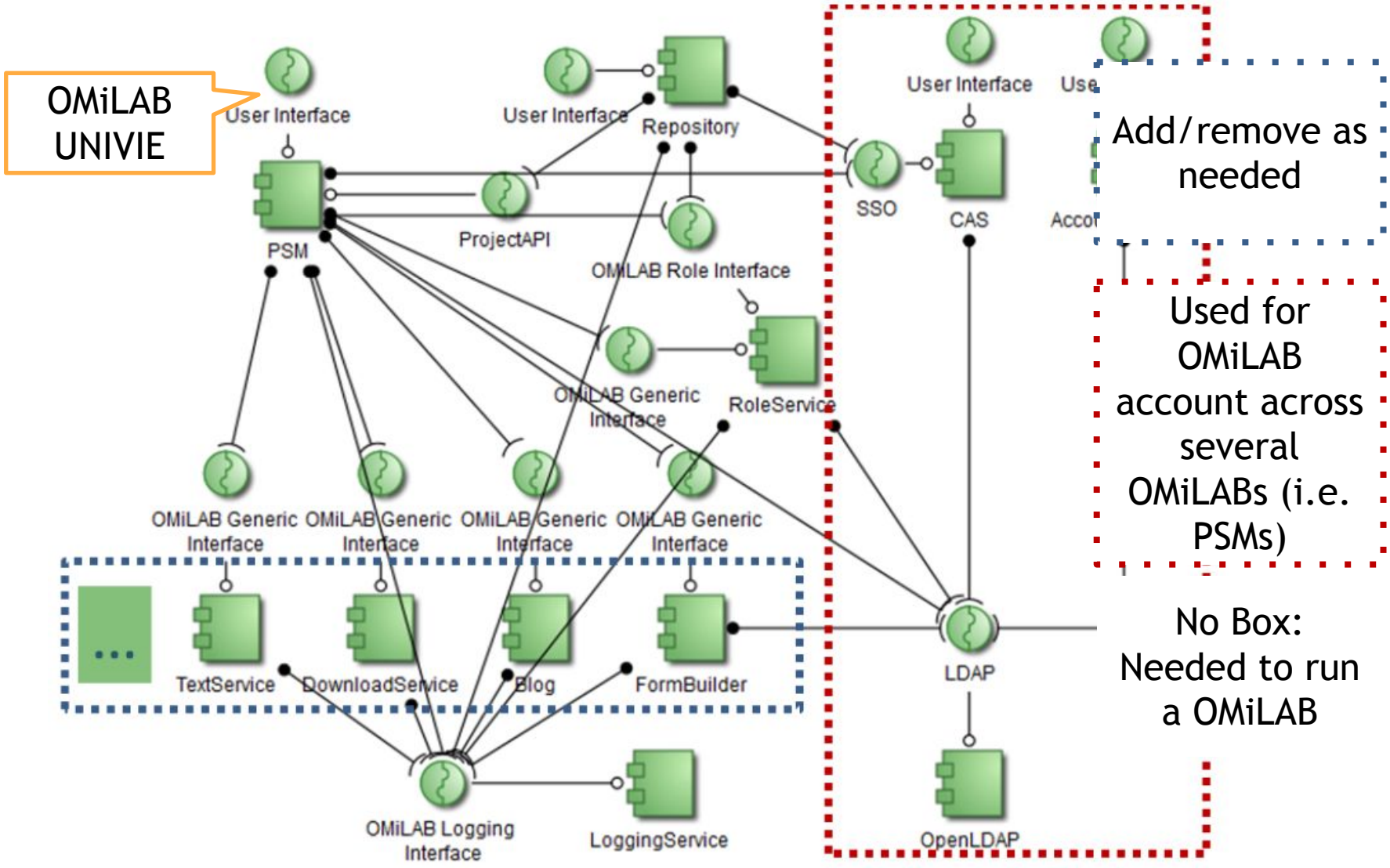
Relevante Literatur

- Martin Fowler & James Lewis:
  - http://martinfowler.com/microservices/
  - http://martinfowler.com/articles/microservices.html
  - http://martinfowler.com/bliki/ServiceOrientedAmbiguity.html
- Microservices Pattern:
  - http://microservices.io/patterns/microservices.html
- Amazon Architektur:
  - http://highscalability.com/blog/2007/9/18/amazon-architecture.html
- Netflix Techblog:
  - http://techblog.netflix.com/2015/02/a-microscope-on-microservices.html

Martin Fowler Keynote über Microservices

Q: https://youtu.be/wgdBVIX9ifA

Example: OMiLAB Portal – Service Architecture (simplified)



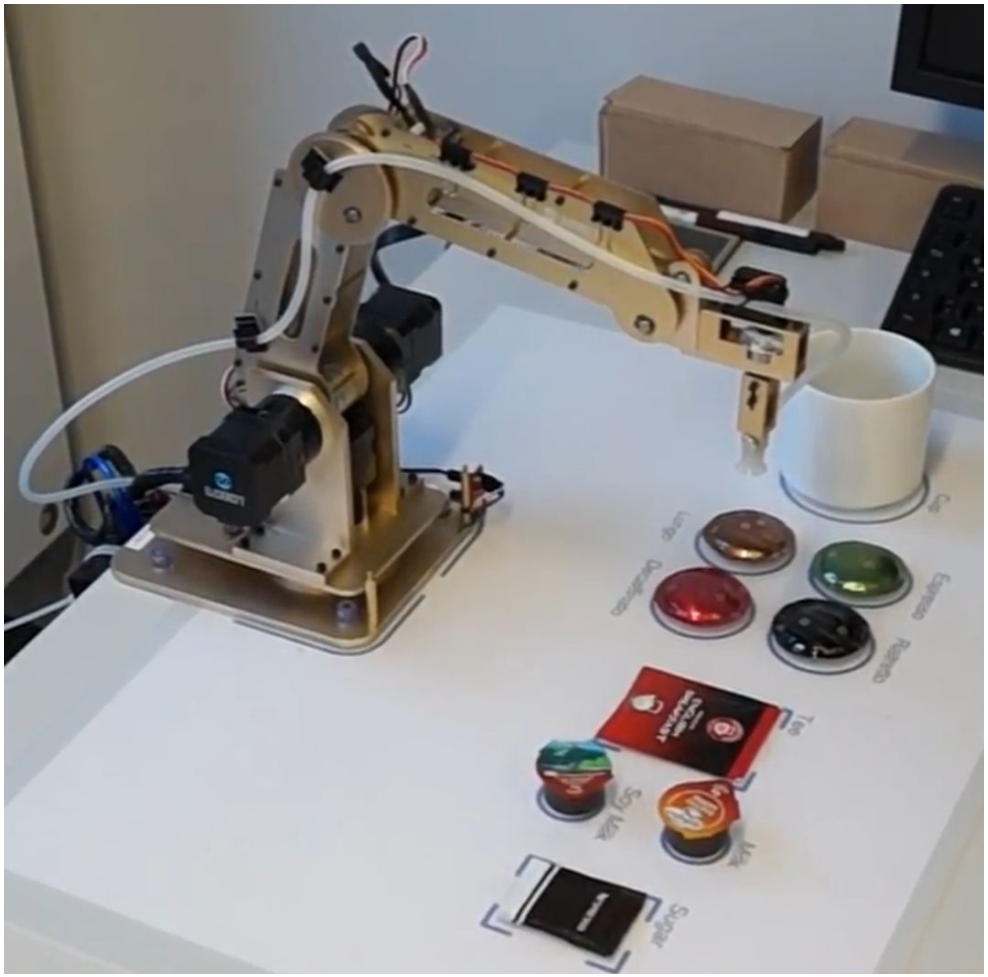
Example: OMiLAB CPS Interaction
Case: Robotic Arm

- The idea is to prepare coffee with the help of a robotic arm and a model describing the process and the different possibilities for creating the coffee.

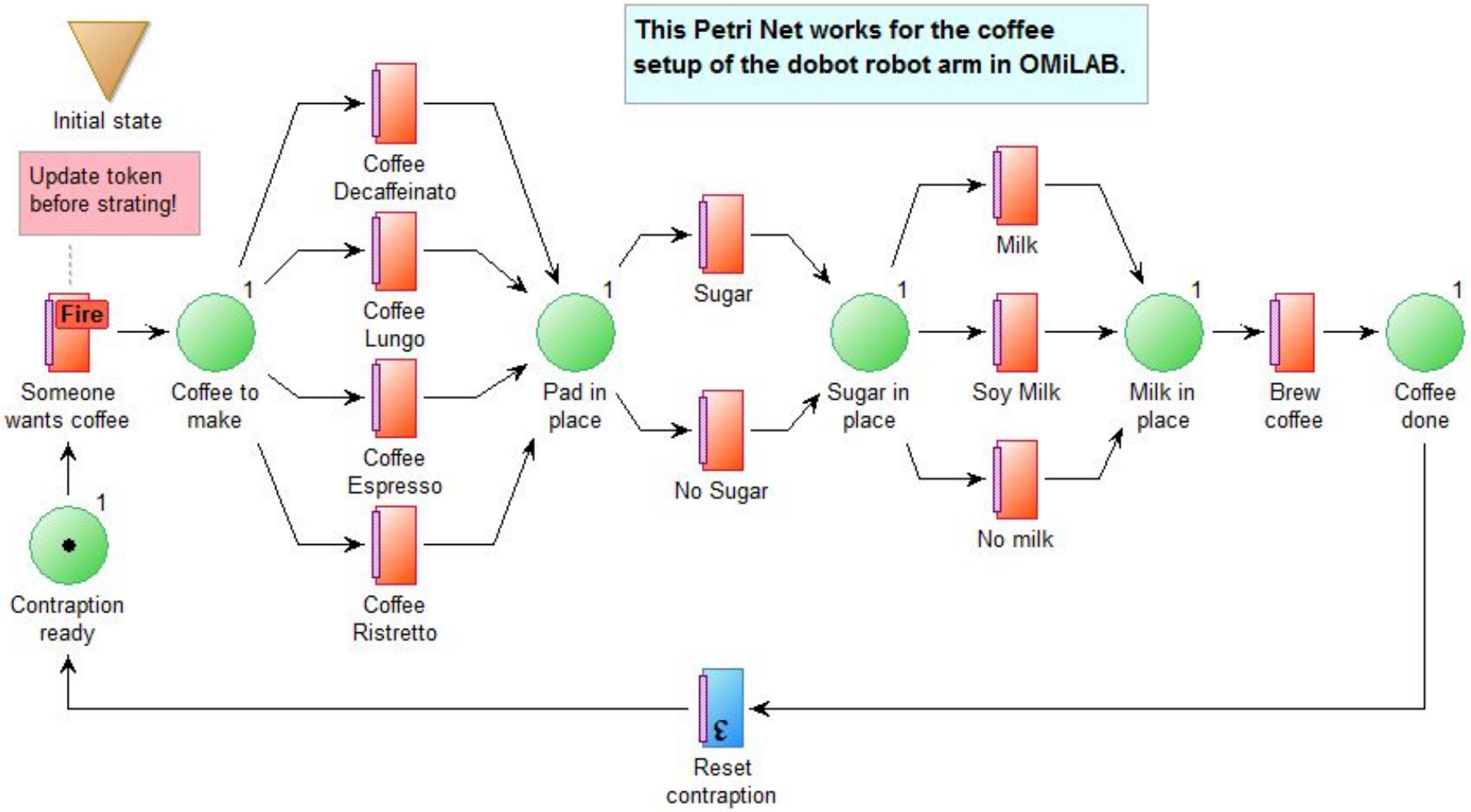
- The robotic arm used is omiarm1, which provides a set of REST services to control the movement and operations of the arm, uses a pump to suck onto and handle light objects and has a coffee setup prepared.

- Petri Nets have been chosen as the modelling language, and the case was described using the Bee-Up modelling tool, which has different capabilities for executing/simulating Petri Nets, provides functionalities to execute code when firing transitions and can also perform HTTP calls.

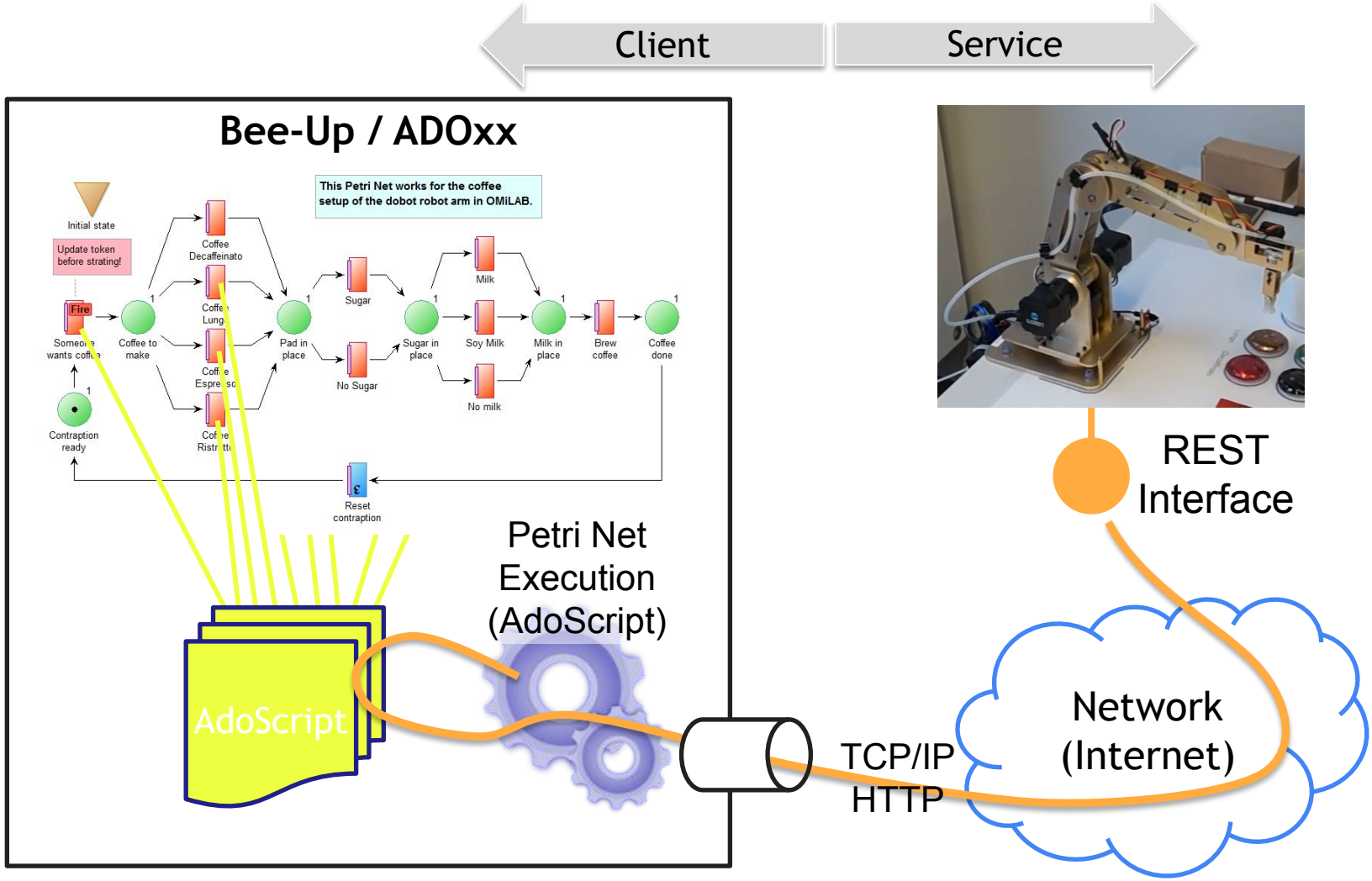
OMIARM1: Device + Microservice

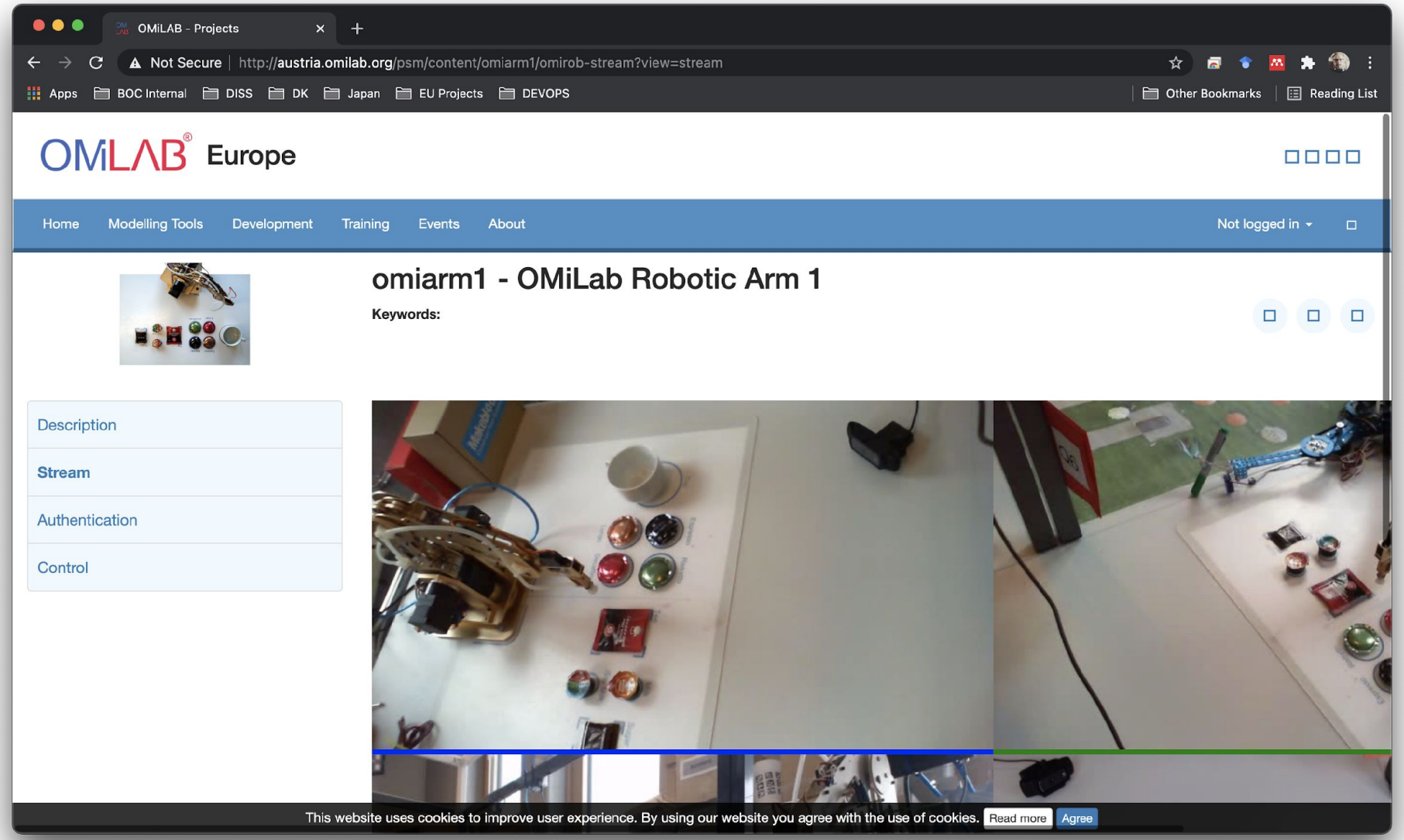


Petri Net Model



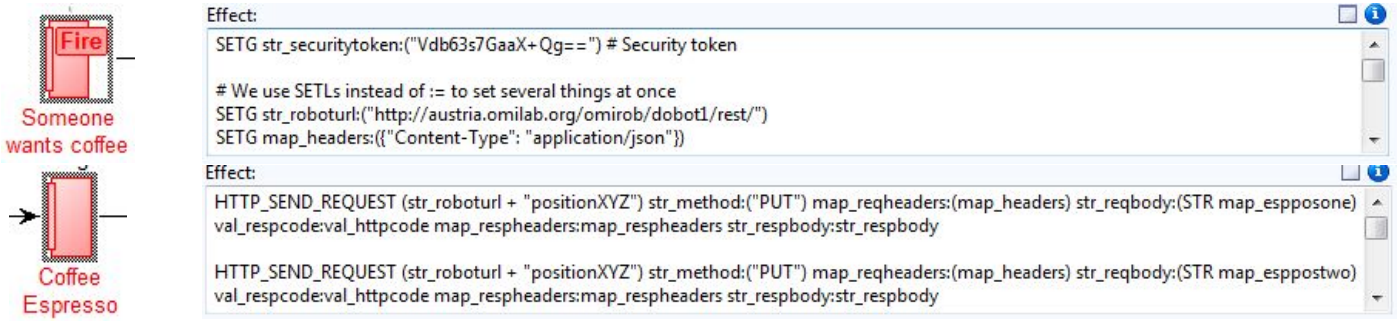
Microservice Architecture




OMiLAB - Projects
Not Secure
http://austria.omilab.org/psm/content/omiarm1/omirob-stream?view=stream
Apps
BOC Internal
DISS
DK
Japan
EU Projects
DEVOPS
Other Bookmarks
Reading List
OMiLAB Europe
Home
Modelling Tools
Development
Training
Events
About
Not logged in
omiarm1 - OMiLab Robotic Arm 1
Keywords:
Description
Stream
Authentication
Control
This website uses cookies to improve user experience. By using our website you agree with the use of cookies.
Read more
Agree

Details: Service Client

- The transitions contain a special „Effect“ code which is executed when they fire.



- This code sets up the basic parameters and sends requests to the robot arm to perform certain operations (move to position, turn on valve etc.)
- Additionally an extension is used which allows and simplifies HTTP calls.

Details: Service Client

- The details about the position of things (e.g. where is the espresso pad) have been determined beforehand and specified in the code. Therefore the positioning template is necessary for this experiment.
- The transitions can be executed manually (by clicking on their “Fire” button) or using one of the simulation capabilities for Petri Nets available in Bee-Up.
- Upon firing the code specified in the transition is executed, which in turn calls the REST service provided by the arm to perform different operations.
- In the model one transition actually sends several calls to the robotic arm service to perform certain operations
  - e.g. the ”Coffee Espresso” transition moves the arm over the pad, picks it up, moves to the cup, drops the pad and then returns to a “neutral” position

Details: Service

- The microservice is implemented in Java and exposes its functionality as a ReST interface per possible operation:
  - ResetOperation: input: (null), output: (success_message)
    Capability: Calibration of the arm, reset to default position (0,0,0)
  - MoveToPosition: input: coordinates, output: (success_message)
    Capability: move to any location within the 3D space reachable by the arm
  - TurnOnSuctionCup: input (null), output (success_message)
    Capability: Turn on the suction cup/compressor
  - TurnOffSuctionCup: input (null), output (success_message)
    Capability: Turn off the suction cup/compressor
  - GetPosition: input (null), output (coordinates)
    Capability: read the current position of the arm
- The service functionality (in Java) is responsible to coordinate the communication to the device (via USB Serial) and utilizes the python endpoint of the device.
- Challenge: synchronous execution, coordination, state persistence

Service UI: Operation of the Robot Arm