DPS
IT FOR FINANCE
AND RETAIL

Vergleich eines Usecases mit Serverless Technologie gegenüber Spring Boot Technologie

am Beispiel von Instant Payments

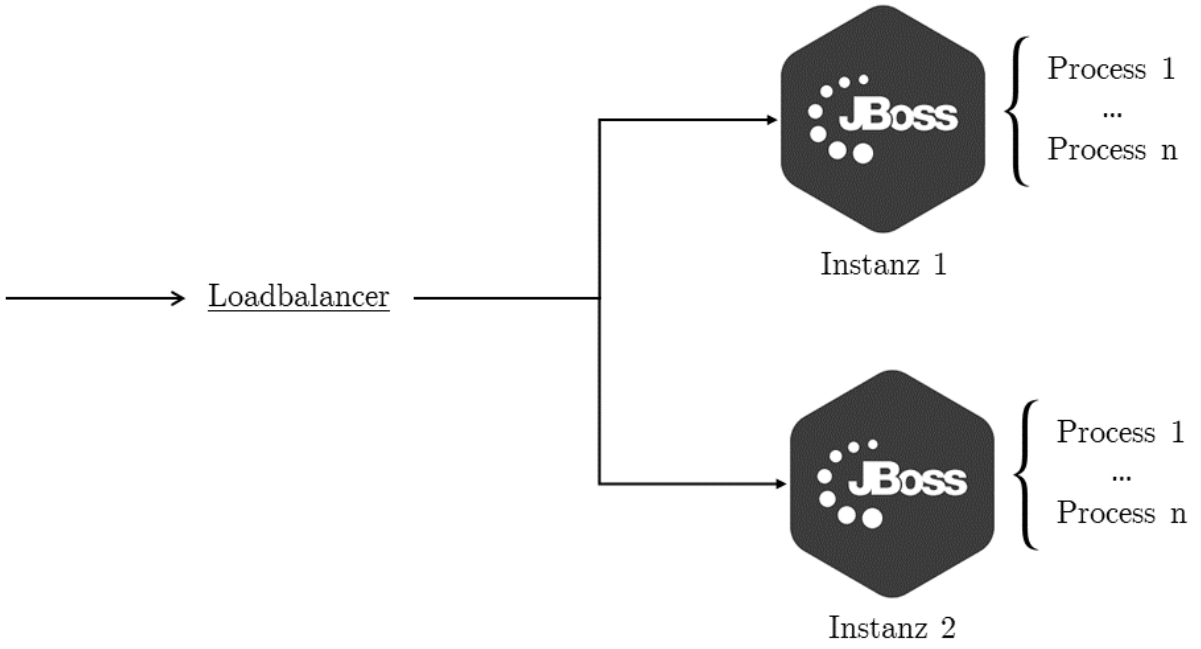
Silas Hoffmann

xy Januar 2021

Ist-Analyse

- Verwendet Application Server
- Aufteilung der Last durch Loadbalancer
- Request-Queue bei Überlauf befüllt
- Dynamische Prozessanzahl

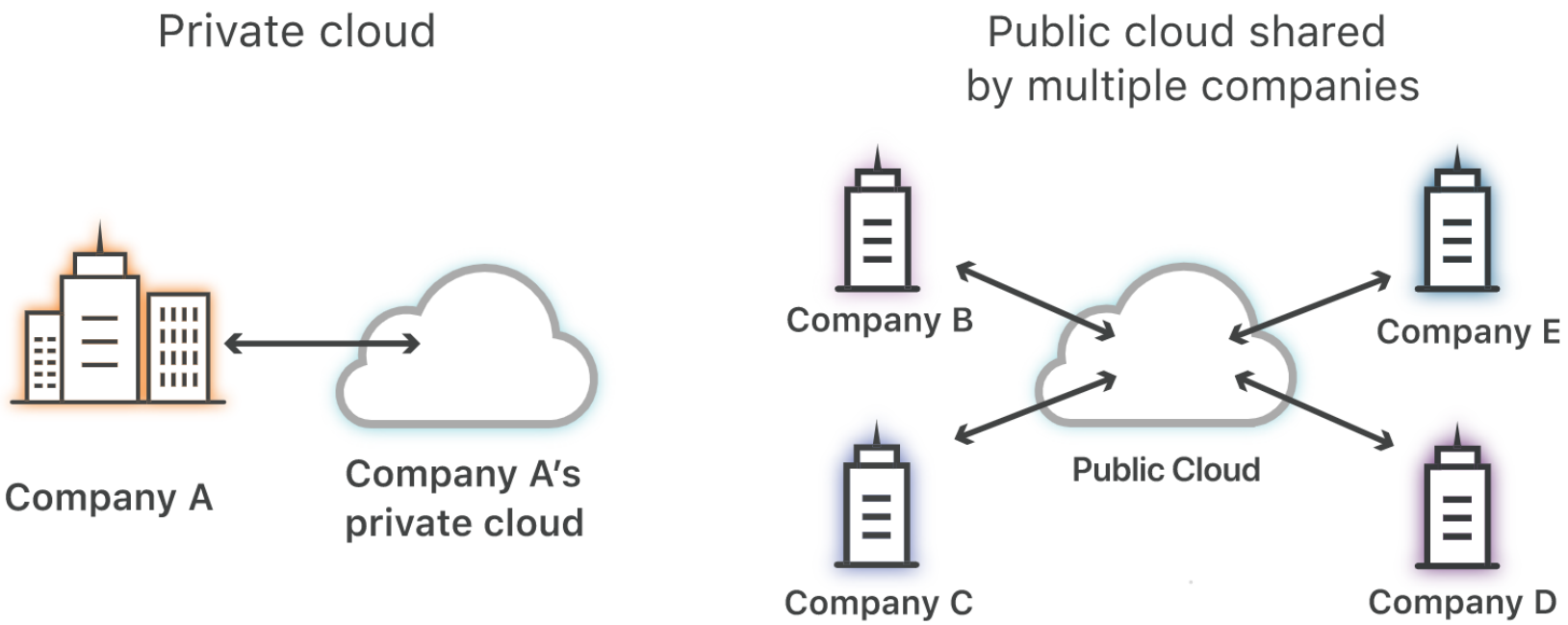
- Monolith: Probleme
  - Skalierte Entwicklung
  - Unabhängiges Deployment
  - Skalierung innerhalb einer Produktivumgebung



Quelle: Hoffmann – Bsc. Thesis

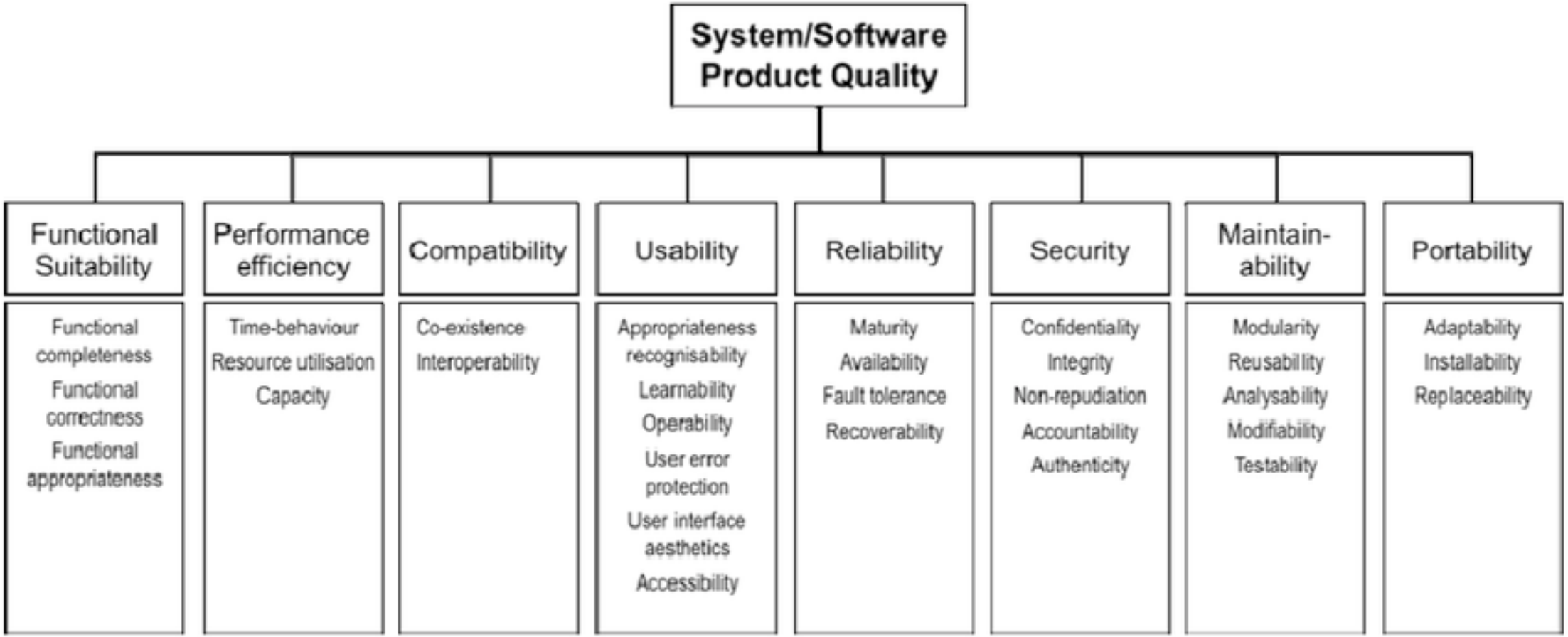
Zielsetzung

- Cloudfähigkeit eines definierten Anwendungsfalls ermitteln
- Cloudfähigkeit zweier moderner Technologien ermitteln



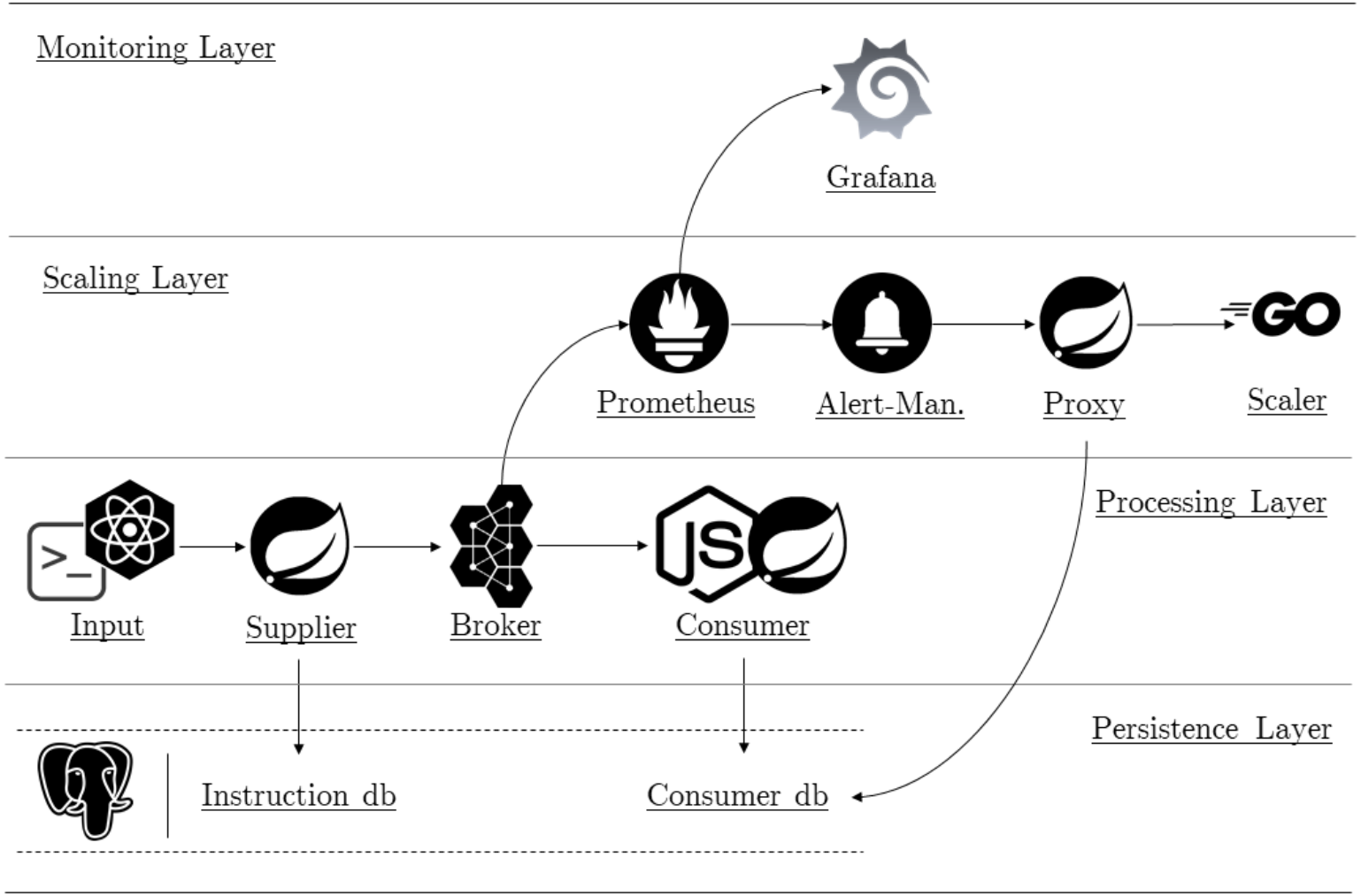
Quelle: https://www.cloudflare.com/de-de/learning/cloud/what-is-a-public-cloud/

Vorgehensmodell



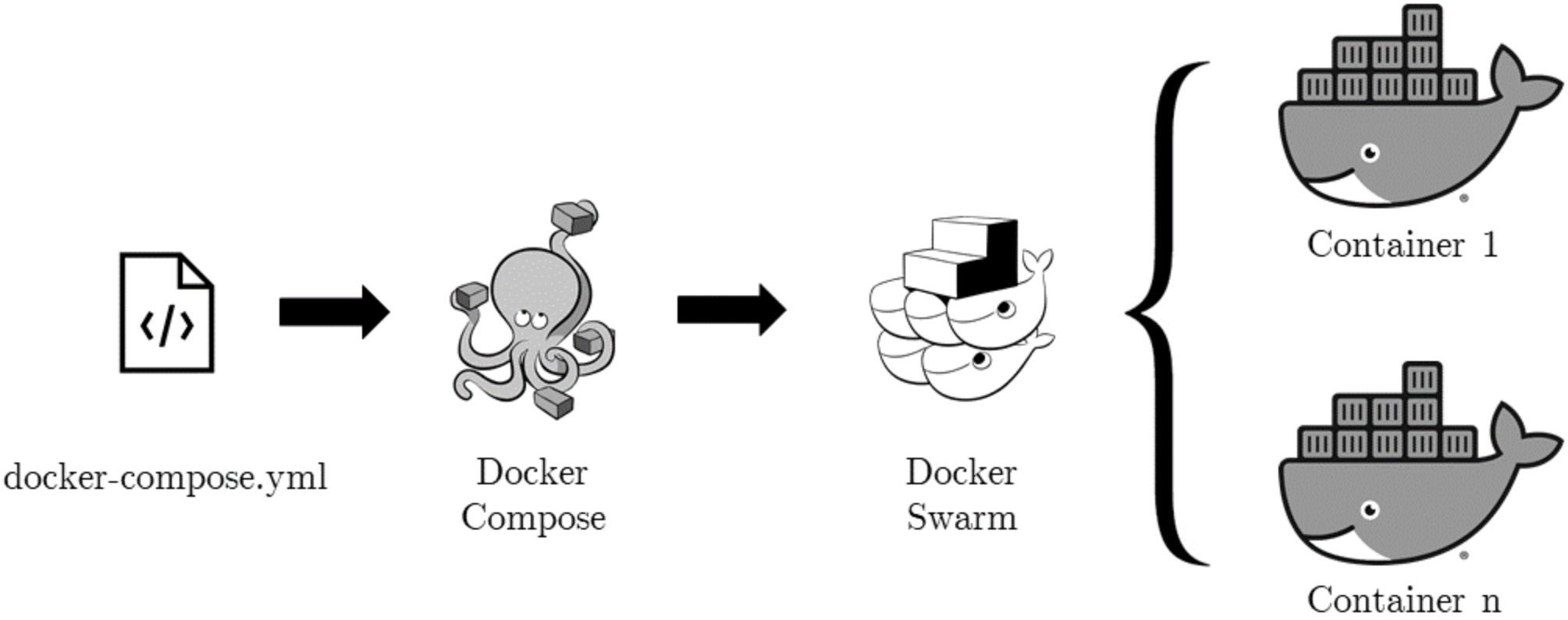
Quelle: ISO/IEC 25010

Implementierung des Prototypen



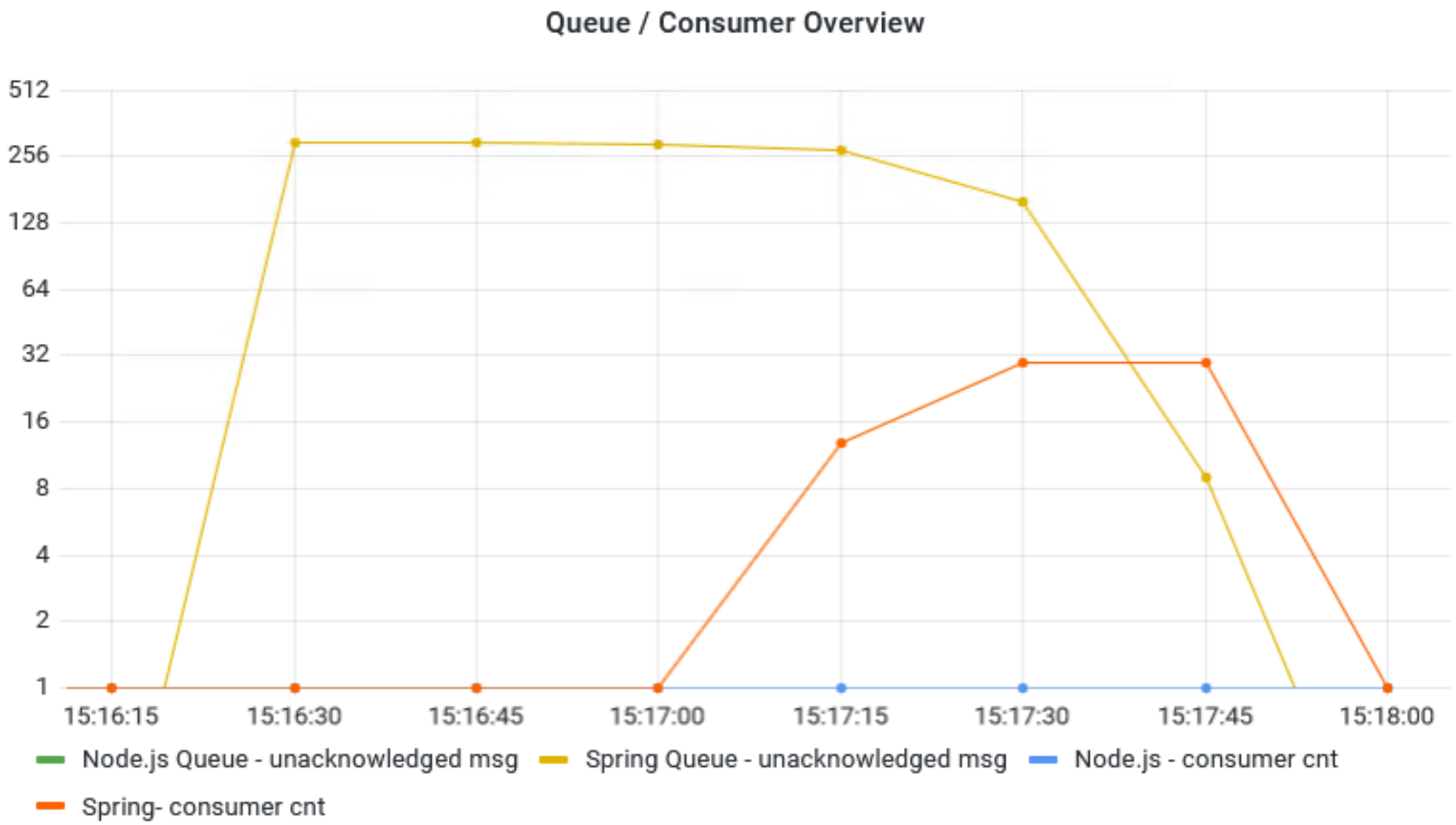
Quelle: Hoffmann – Bsc. Thesis

Implementierung des Prototypen



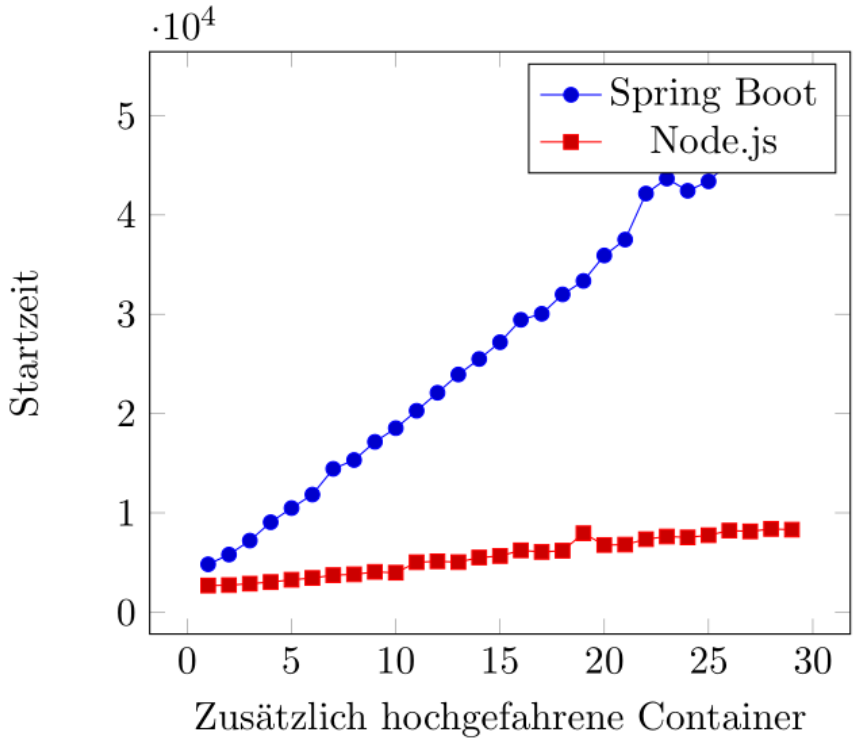
Quelle: Hoffmann – Bsc. Thesis

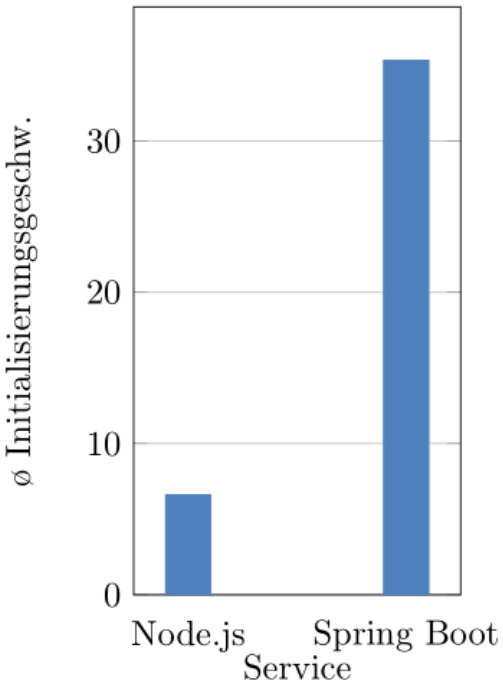
Implementierung des Prototypen

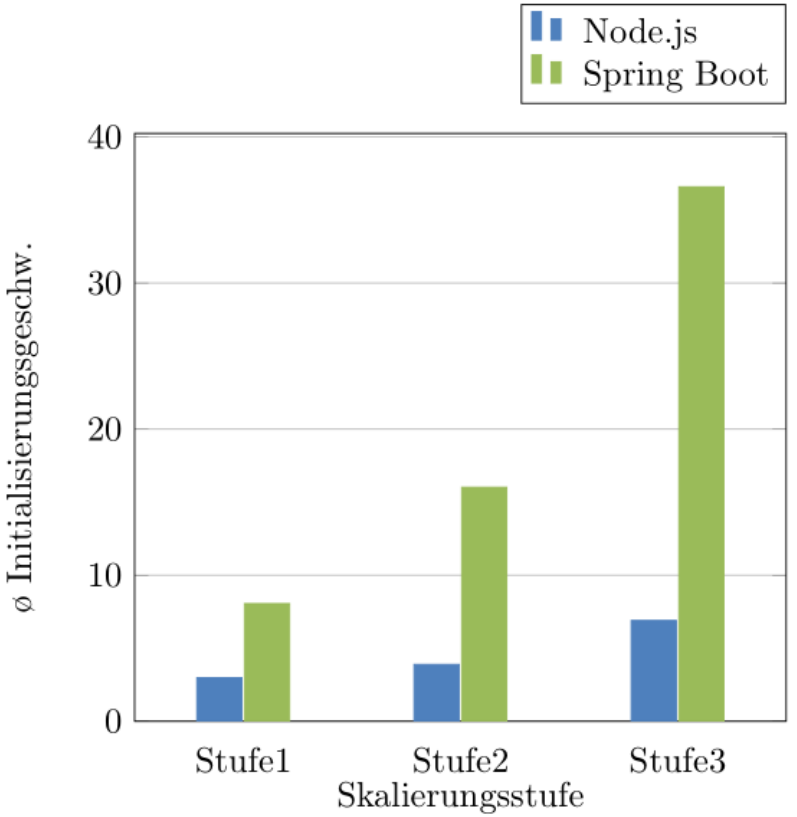


Quelle: Hoffmann – Bsc. Thesis

Ergebnisanalyse / Fazit



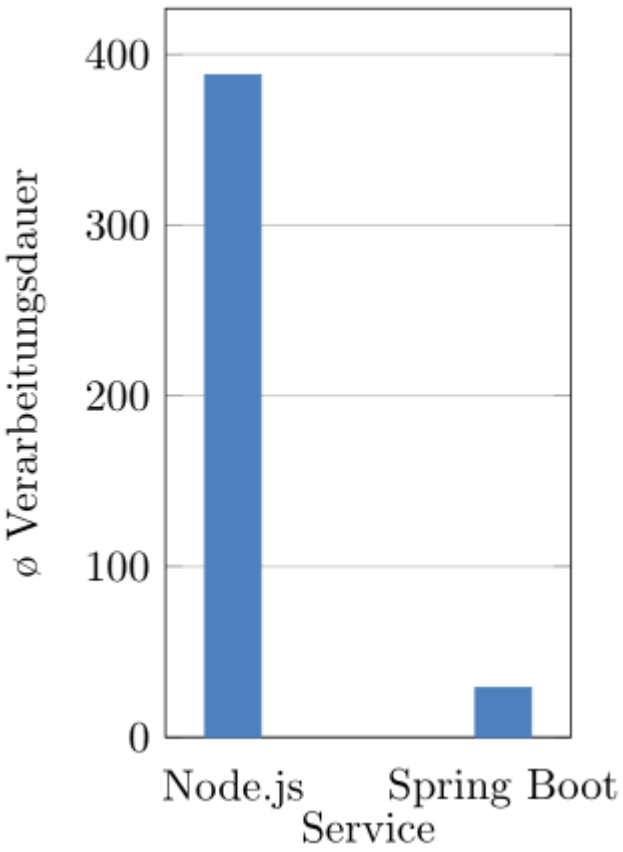


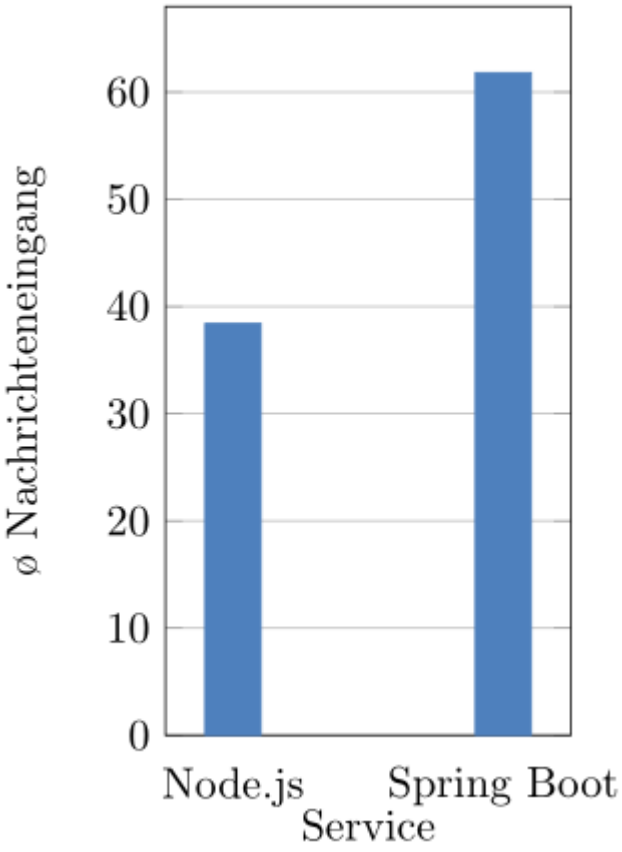


Quelle: Hoffmann – Bsc. Thesis

Ergebnisanalyse / Fazit

- Node.js mit besserem Skalierungsverhalten
- Spring Boot mit besserer Verarbeitungsgeschwindigkeit
- Unterschied beim Nachrichteneingang vernachlässigbar





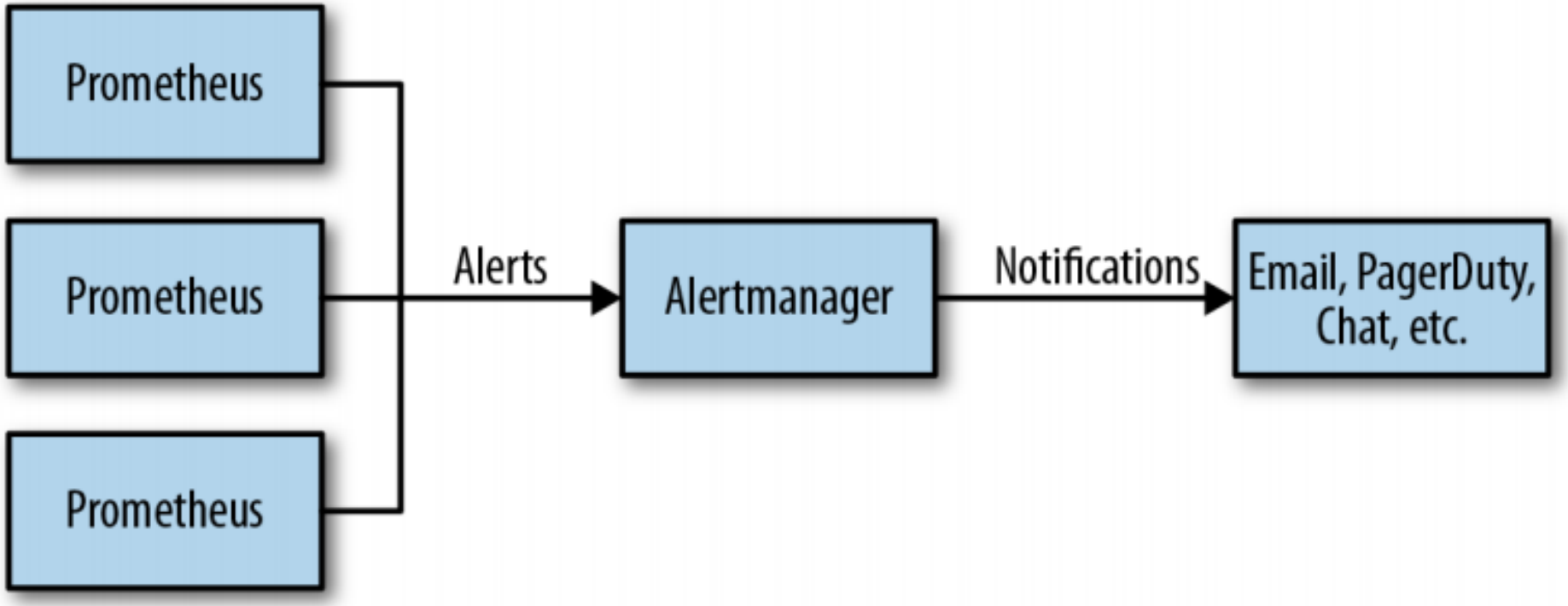
Quelle: Hoffmann – Bsc. Thesis

Skalierung - Regelsatz

QL3 $QB2 < MC$	UP $abs(CB0 - CB3)$	UP $abs(CB1 - CB3)$	UP $abs(CB2 - CB3)$	OK _
QL2 $QB1 < MC \leq QB2$	UP $abs(CB0 - CB2)$	UP $abs(CB1 - CB2)$	OK _	DOWN $abs(CB2 - CB3)$
QL1 $QB0 < MC \leq QB1$	UP $abs(CB0 - CB1)$	OK _	DOWN $abs(CB1 - CB2)$	DOWN $abs(CB1 - CB3)$
QL0 $MC \leq QB0$	OK _	DOWN $abs(CB0 - CB1)$	DOWN $abs(CB0 - CB2)$	DOWN $abs(CB0 - CB3)$
	CL0 $CB0 == CC$	CL1 $CB0 < CC \leq CB1$	CL2 $CB1 < CC \leq CB2$	CL3 $CB2 < CC \leq CB3$

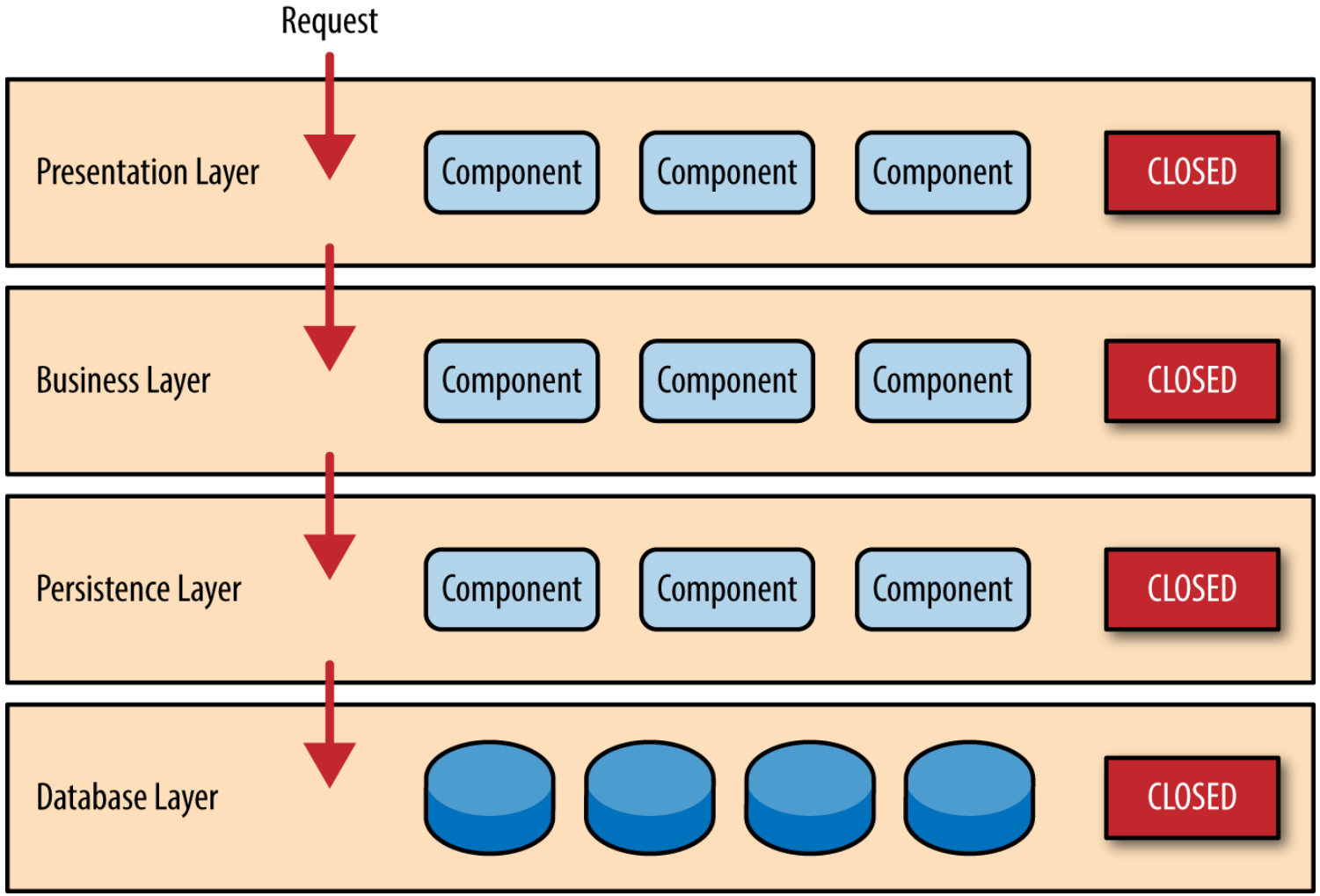
```
CB0=1     CB2=10     QB0=15     QB2=100     CC: Container Count
CB1=5     CB3=30     QB1=30                 MC: Message Count
```

Prometheus / Altermanager



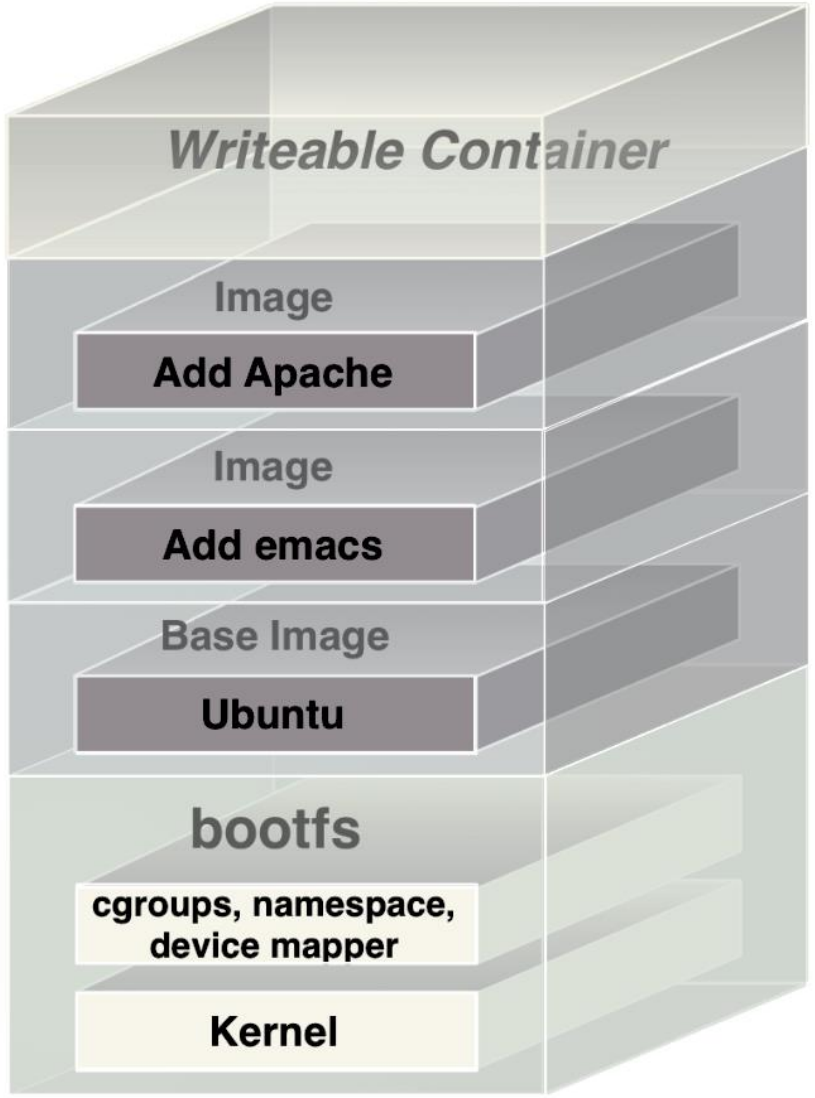
Quelle: Brazil – Prometheus: Up & Running (S. 291)

Tier - Modell



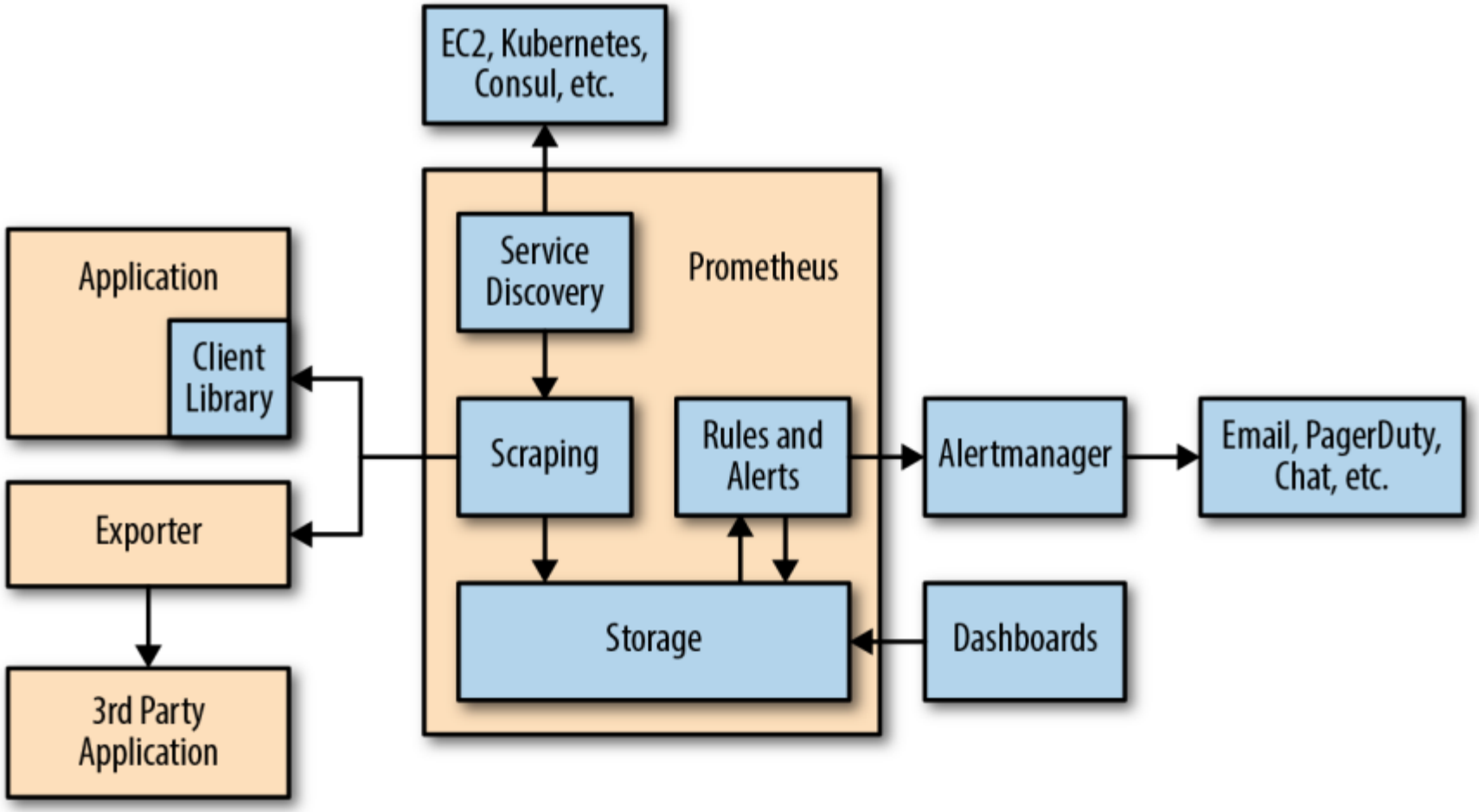
Quelle: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

Docker - Aufbau

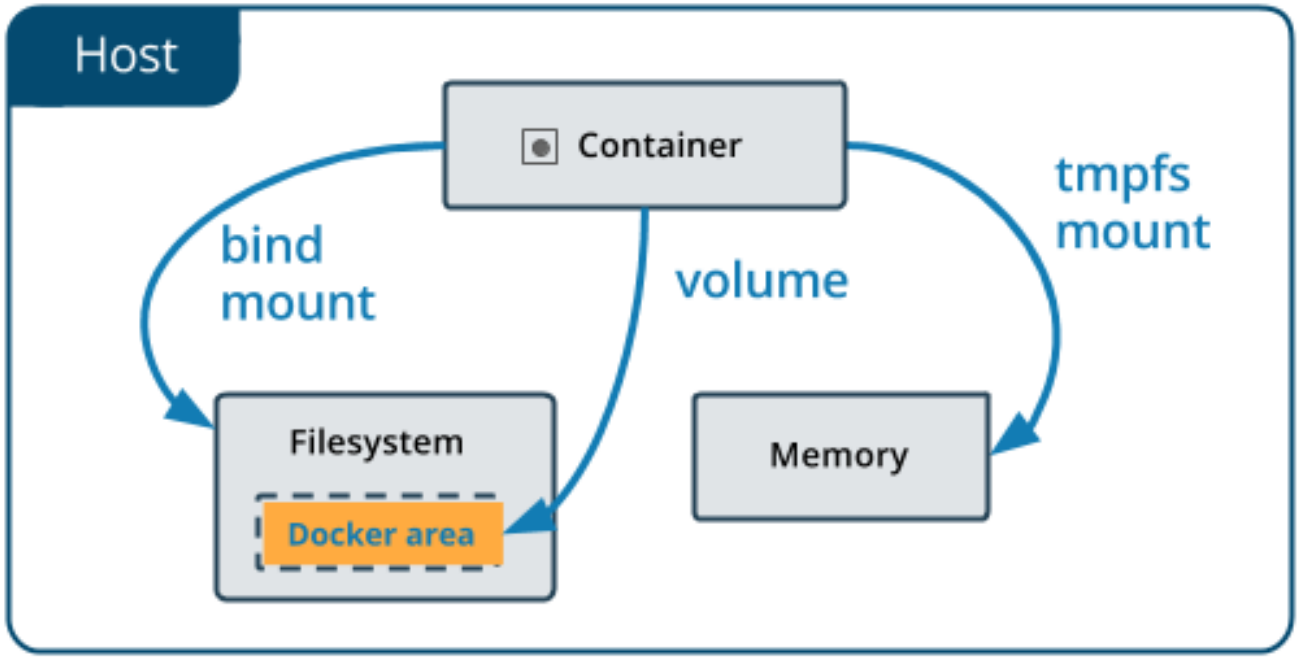


Quelle: J. Turnbull – The Docker Book (S. 72)

Prometheus - Architecture

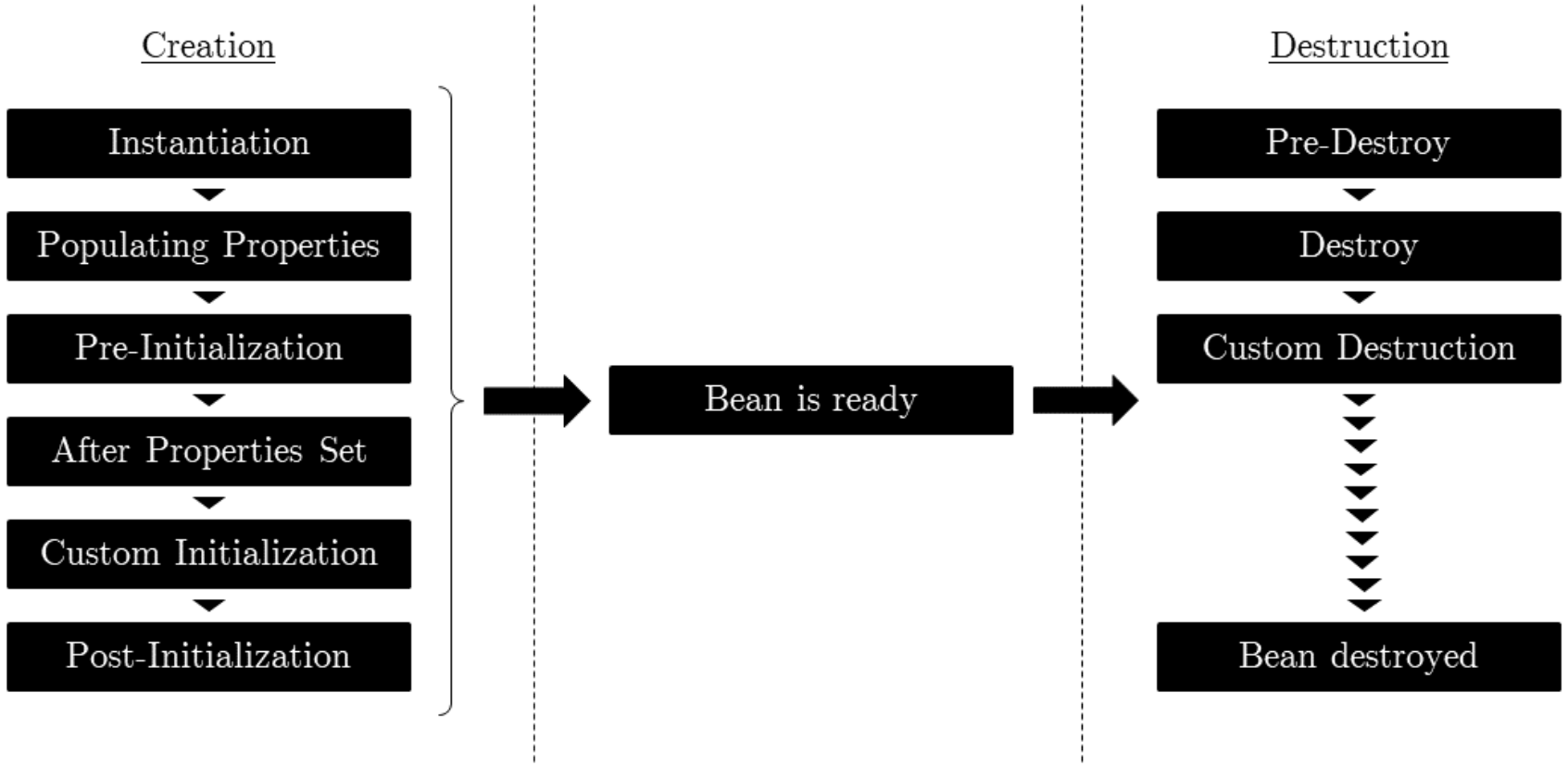


Docker – Types of mounts



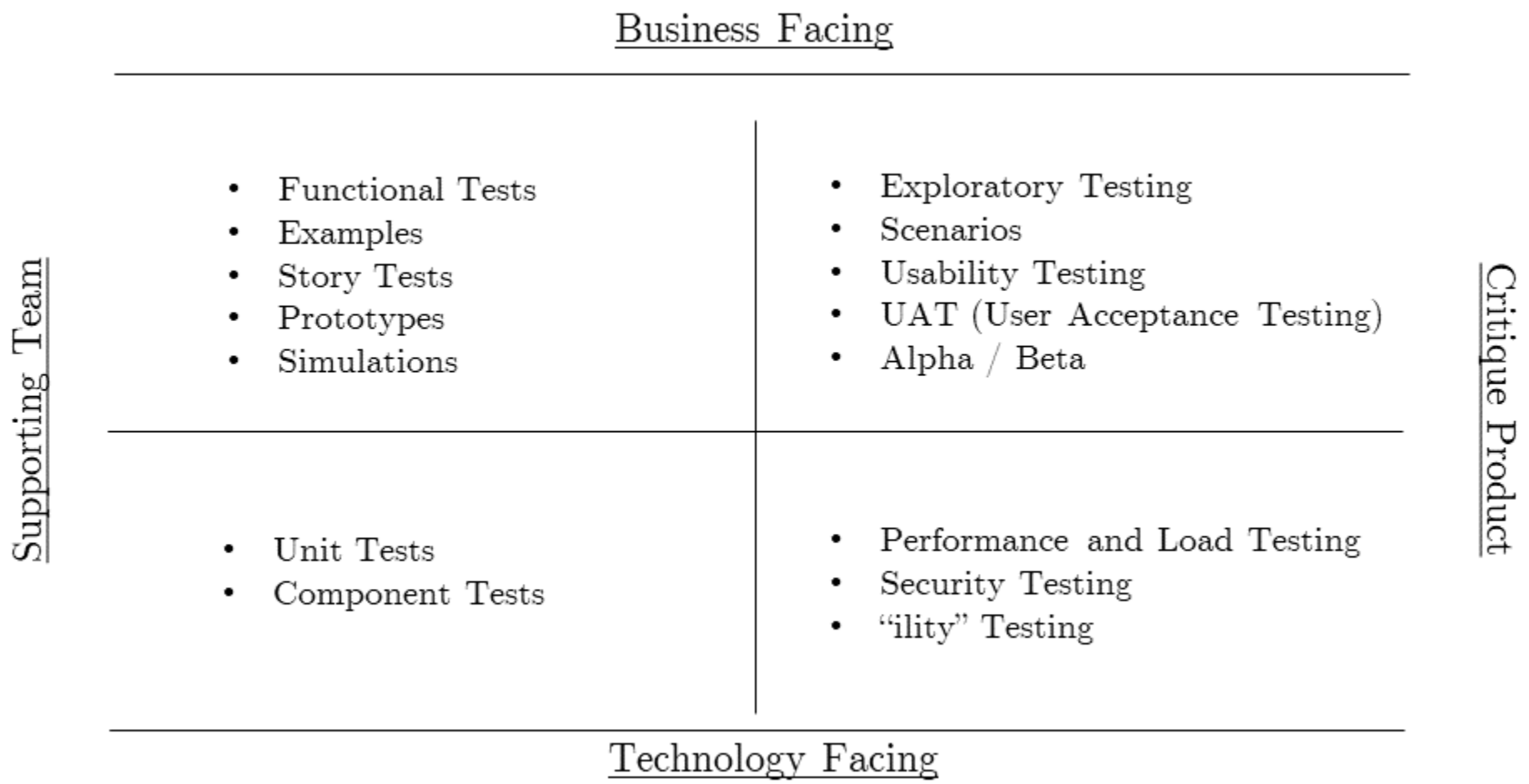
Quelle: Docker Documentation - Kapitel /storage/volumes/

Spring Bean - Lifecycle



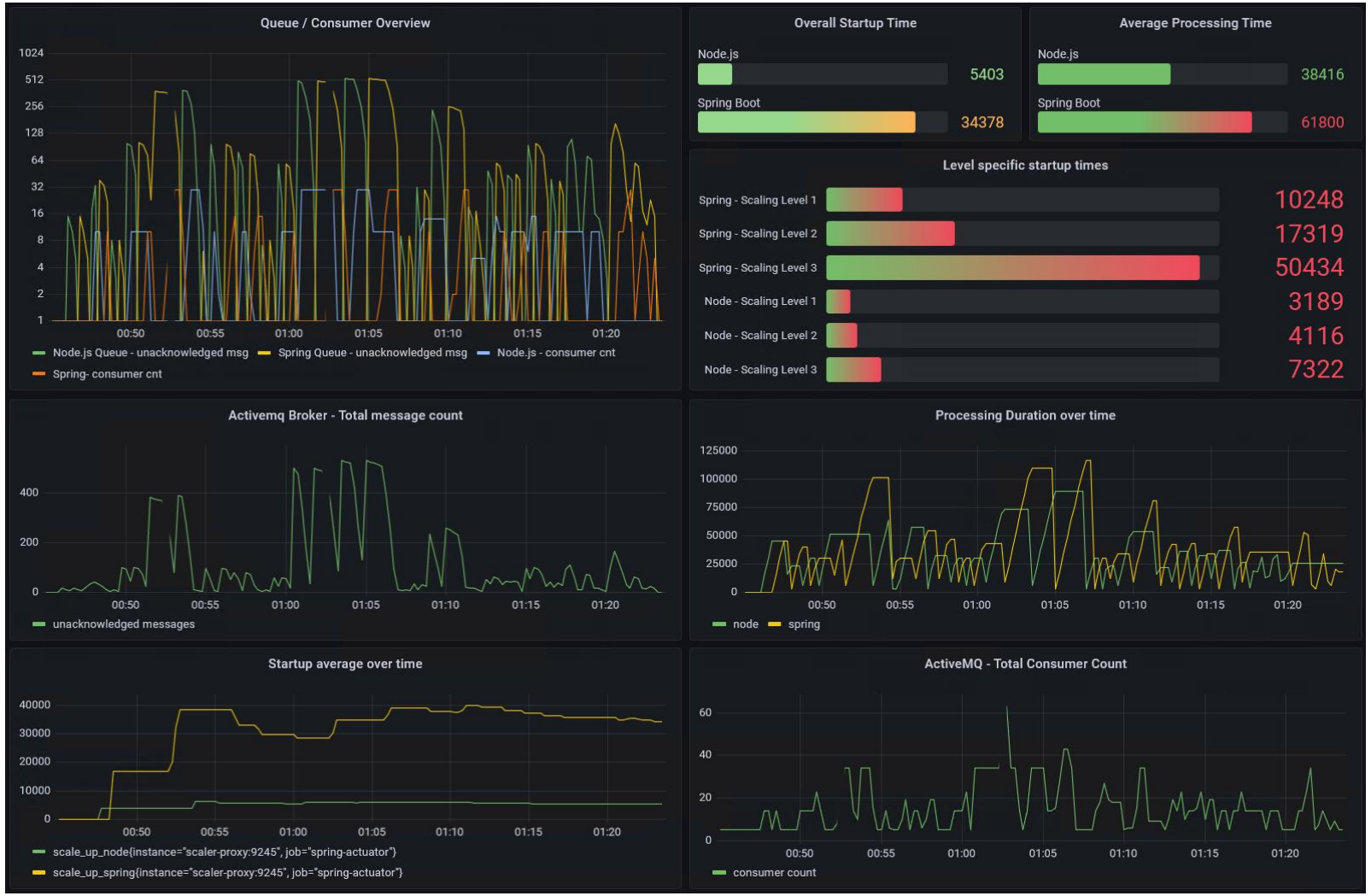
Quelle: Hoffmann – Bsc. Thesis

Agile Testing Quadrants



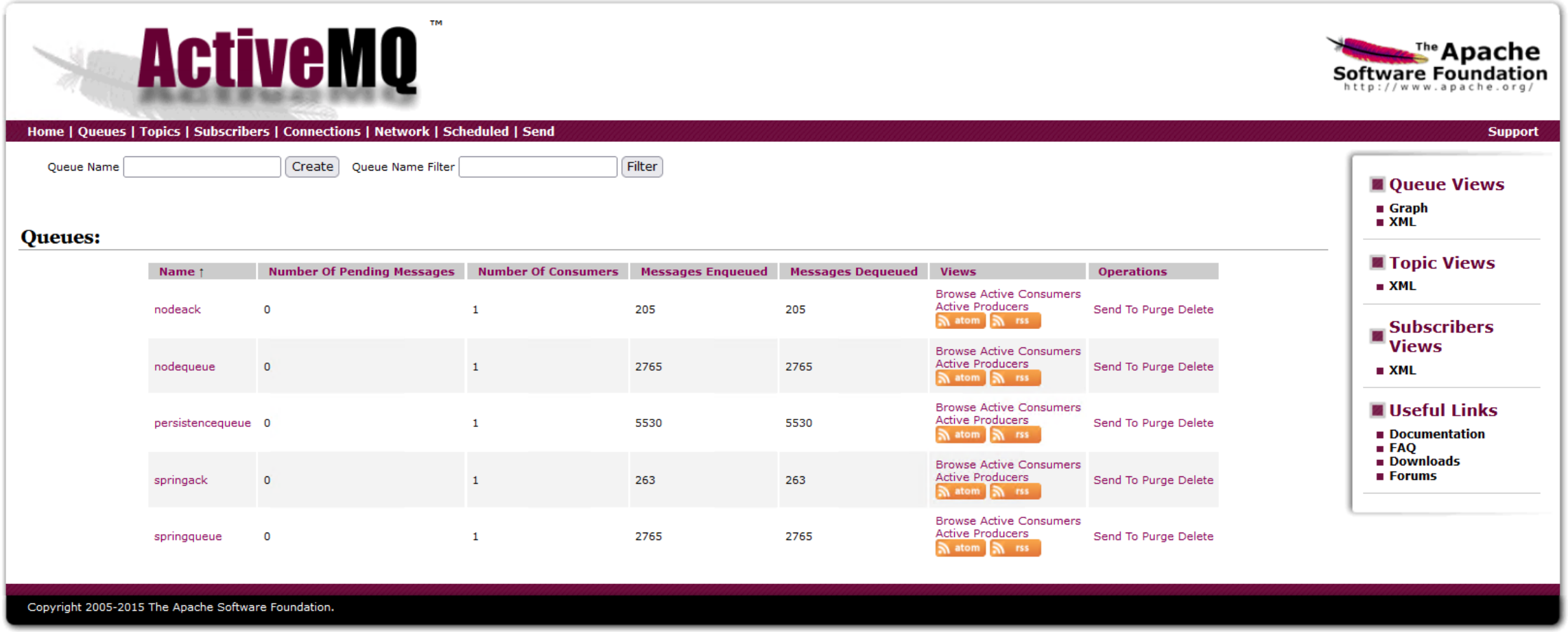
Quelle: Hoffmann – Bsc. Thesis

Implementierung des Prototypen



Quelle: Hoffmann – Bsc. Thesis

Activemq - Dashboard

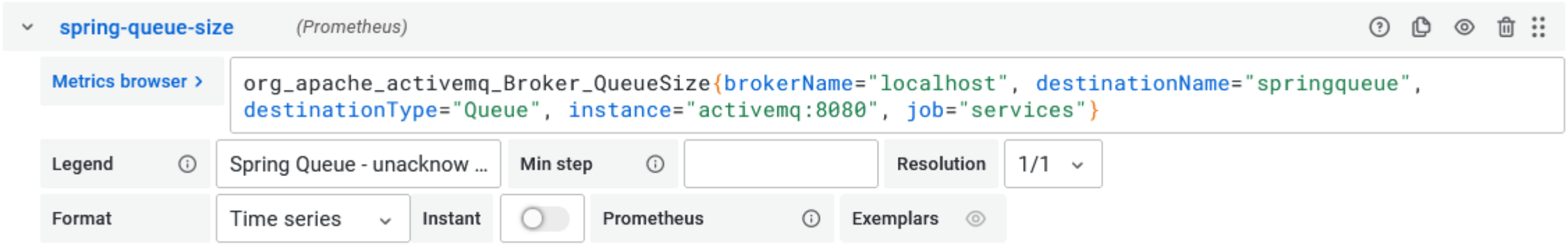


Quelle: Hoffmann – Bsc. Thesis

Input UI

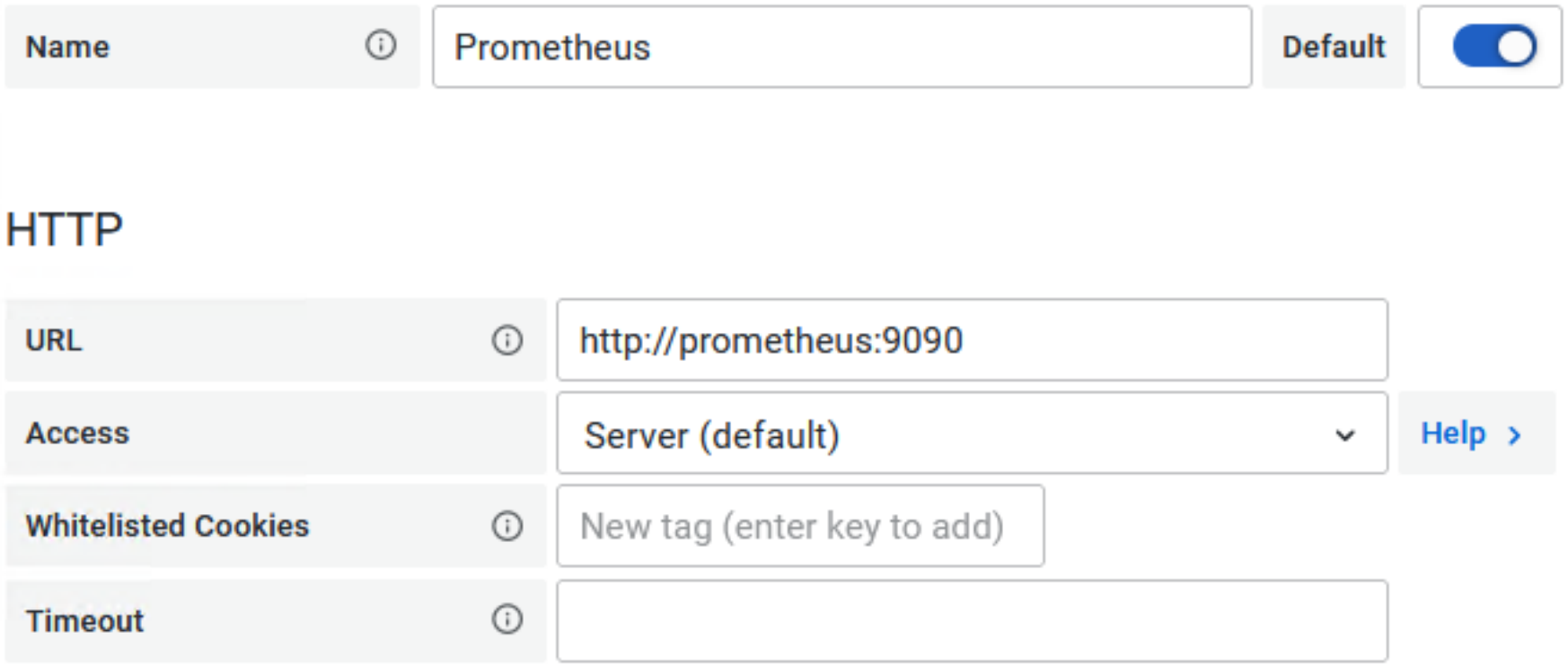
Quelle: Hoffmann – Bsc. Thesis

Grafan - PromQL



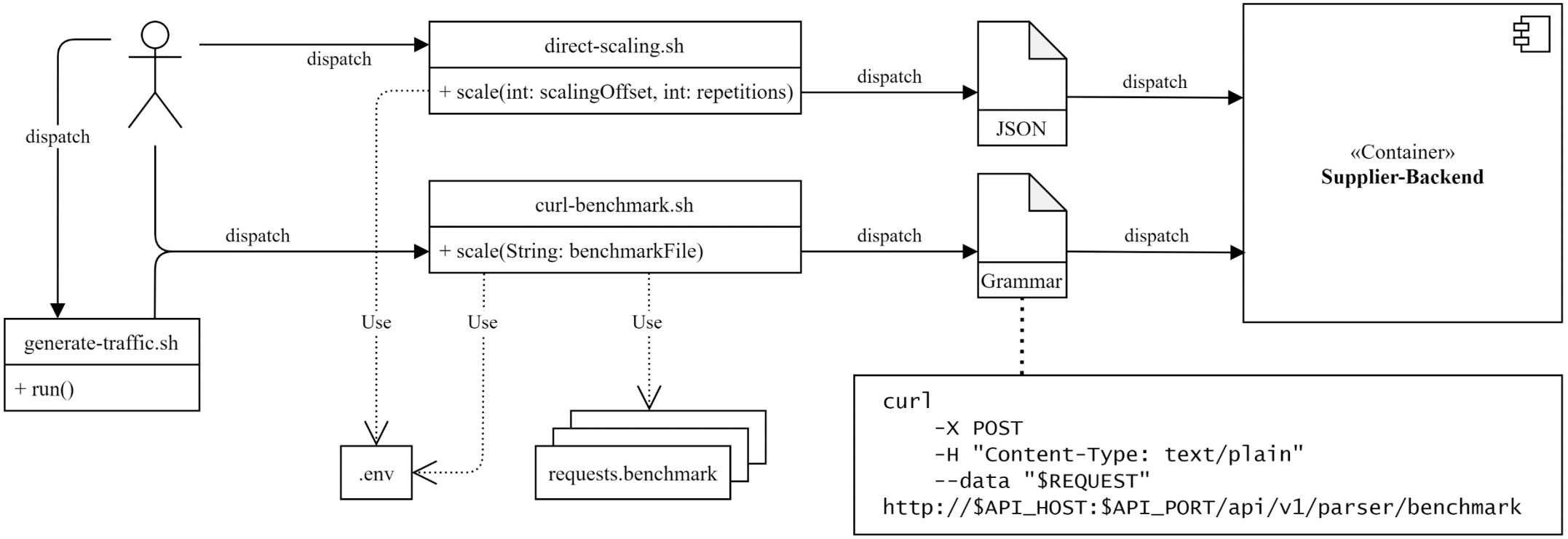
Quelle: Hoffmann – Bsc. Thesis

Prometheus - Datasource



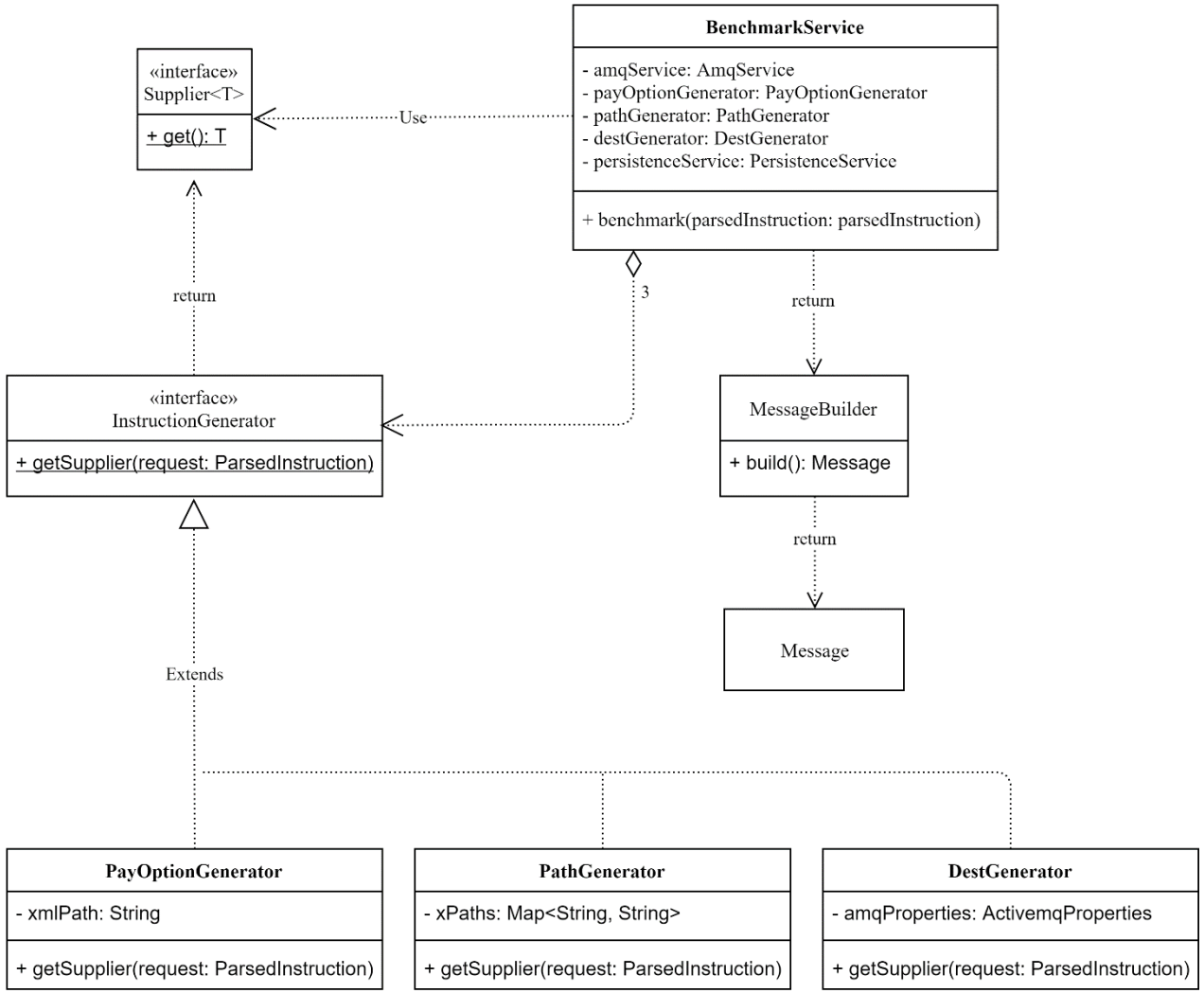
Quelle: Hoffmann – Bsc. Thesis

Input - UML



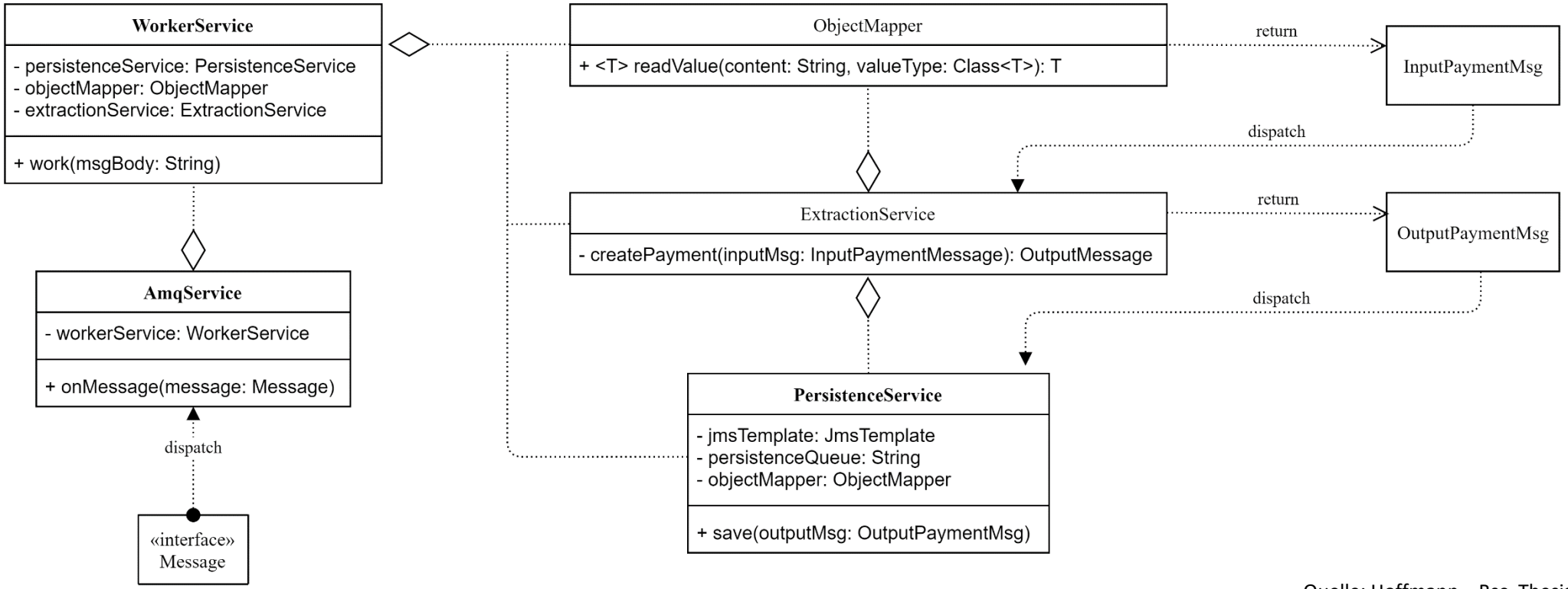
Quelle: Hoffmann – Bsc. Thesis

Supplier - UML



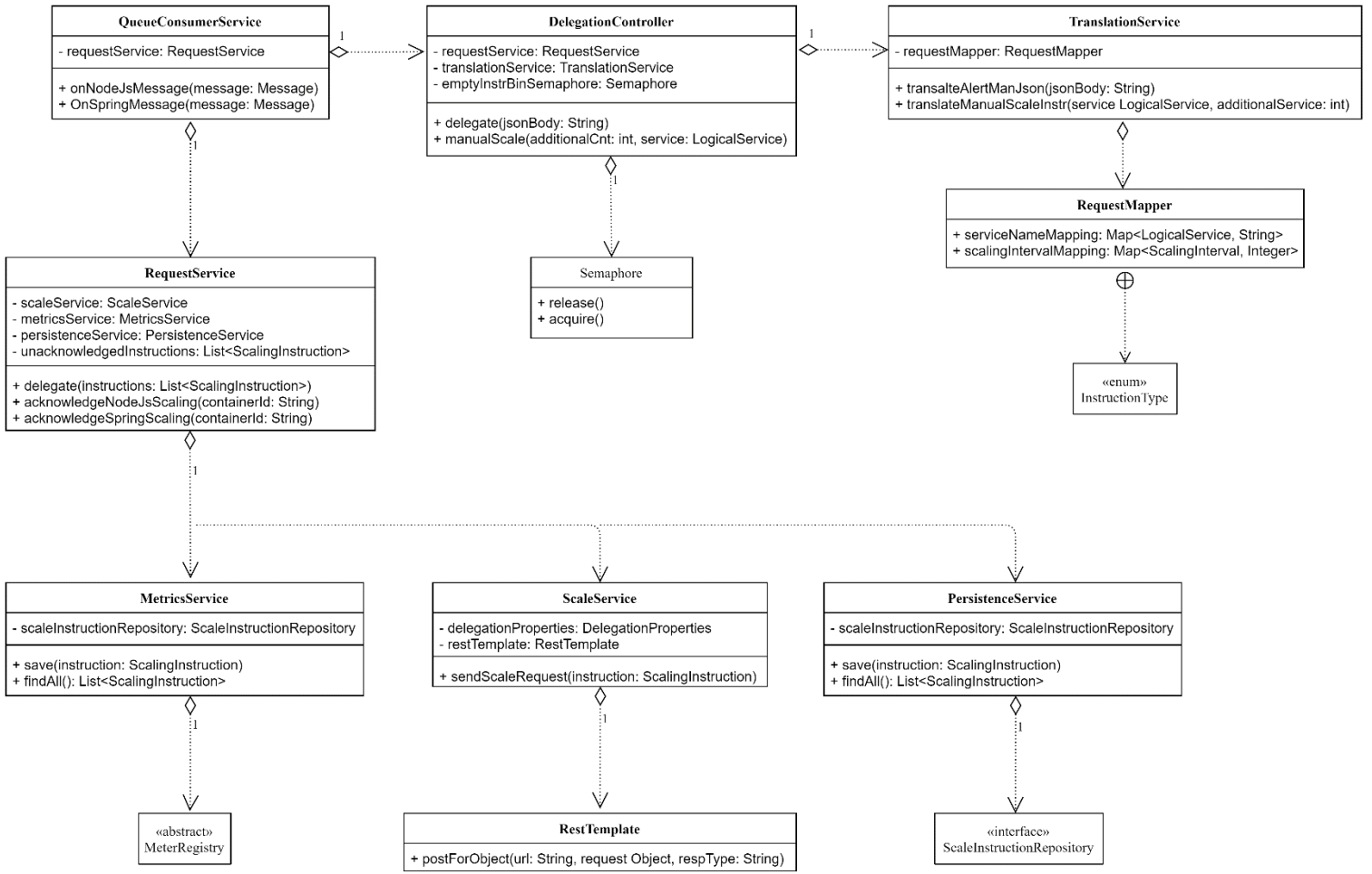
Quelle: Hoffmann – Bsc. Thesis

Consumer - UML



Quelle: Hoffmann – Bsc. Thesis

Scaler Proxy - UML



Quelle: Hoffmann – Bsc. Thesis