Semesterprojekt Laufzeitumgebung

Christoph Hellmich

Inhalt

1. Anforderungen
2. Architektur
3. Entwurfsentscheidungen
4. Klassifizierungsframework nach Crnkovic
5. Fazit

Anforderungen: LZU 1/2

- Komponente laden, starten, stoppen und entfernen
- Operationen auf Komponente protokollieren und rekonstruieren
- Status von Komponente abfragen
- Komponenten parallel ausführen

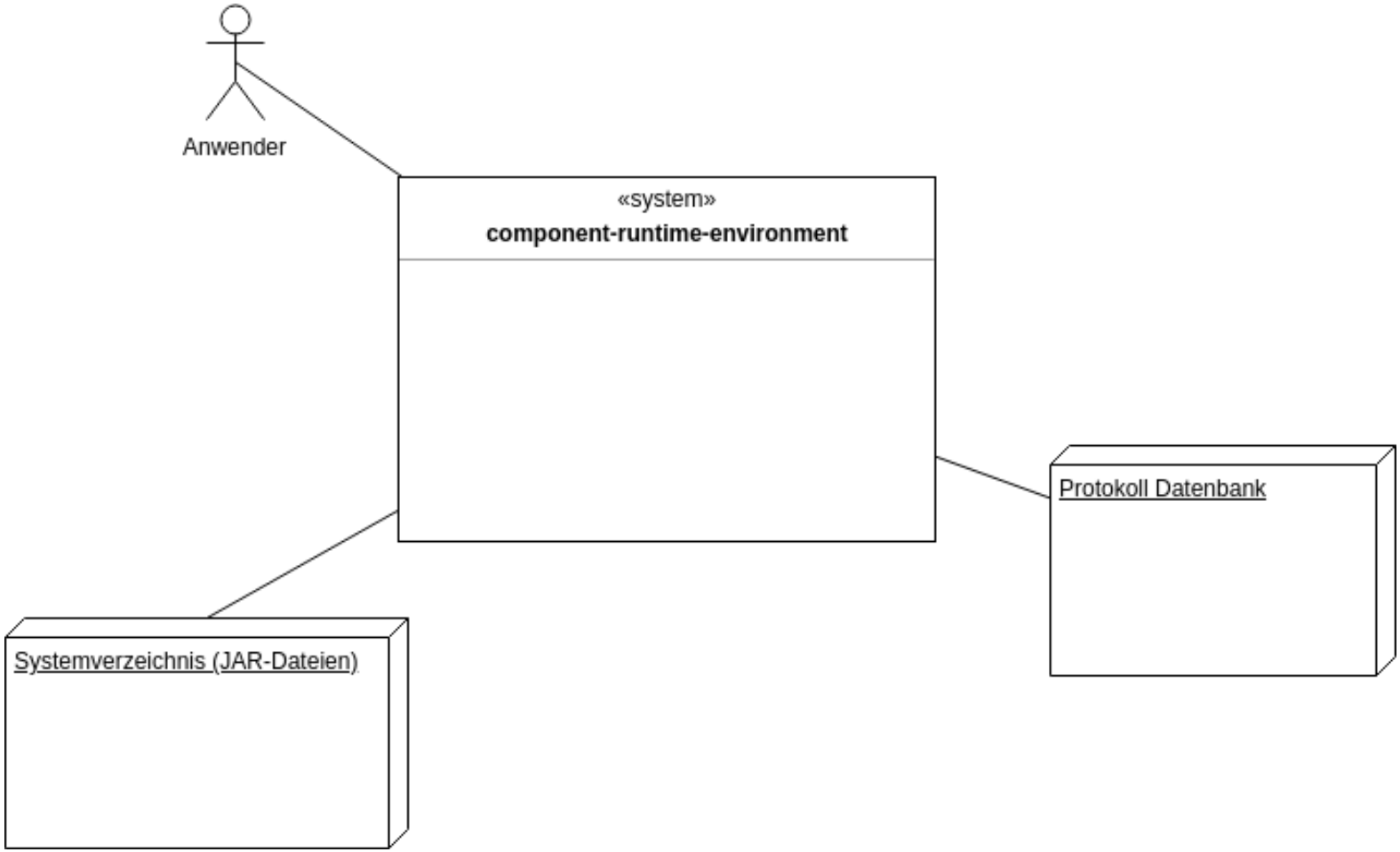
Anforderungen: LZU 2/2

- Logging-Dienst bereitstellen
- Kommunikation mittels Event-System
- Injektion abhängig vom Lebenszyklus
- Steuerung über Web-Frontend

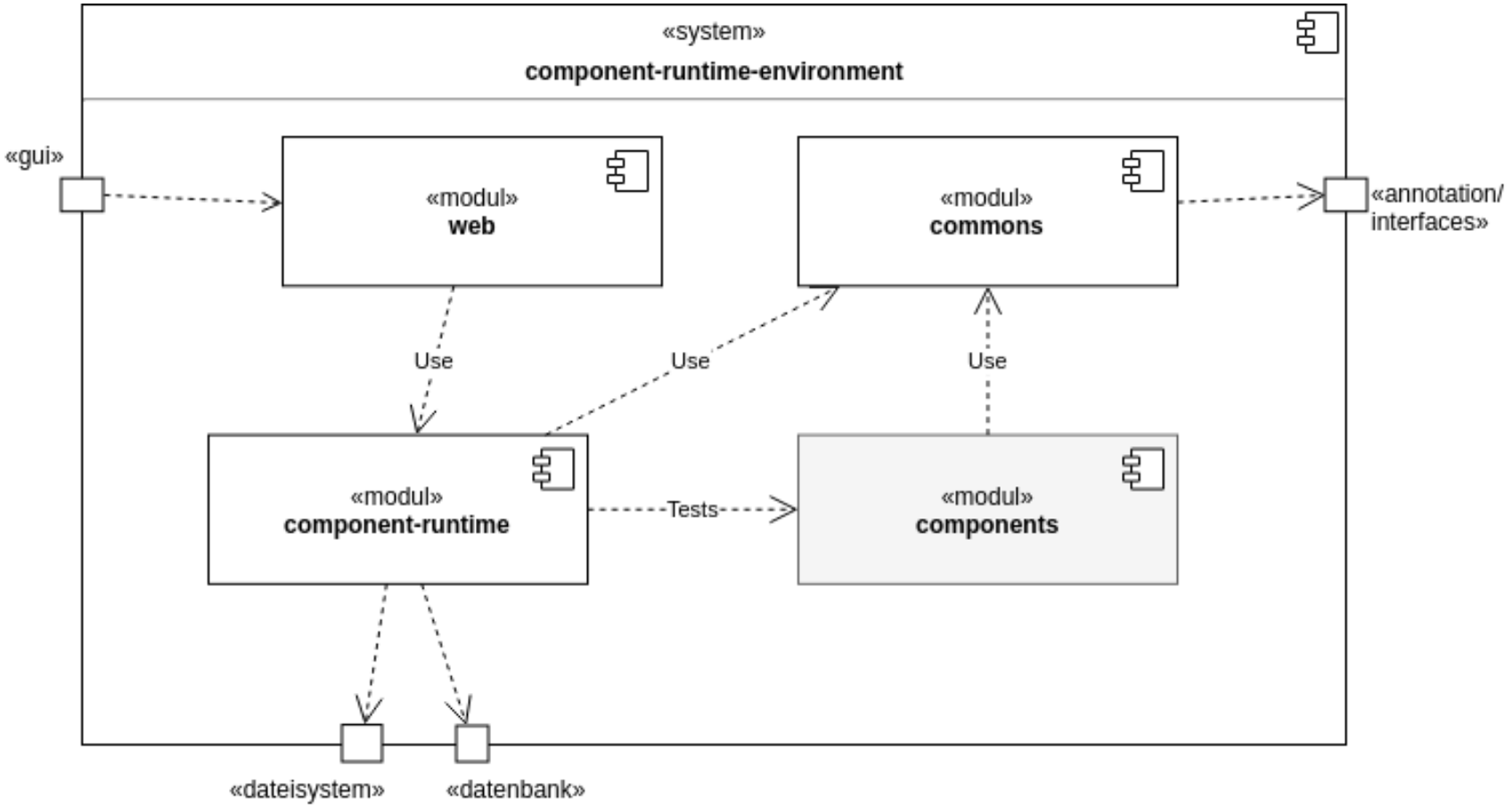
Anforderungen: Komponente

- ausführbare JAR-Datei, d.h.
  - Main-Class im Manifest definiert
  - alle Abhängigkeiten enthalten
- obligatorisch: Start- und Stop-Methode
- optional: Logger, Event, Observer, Injektion

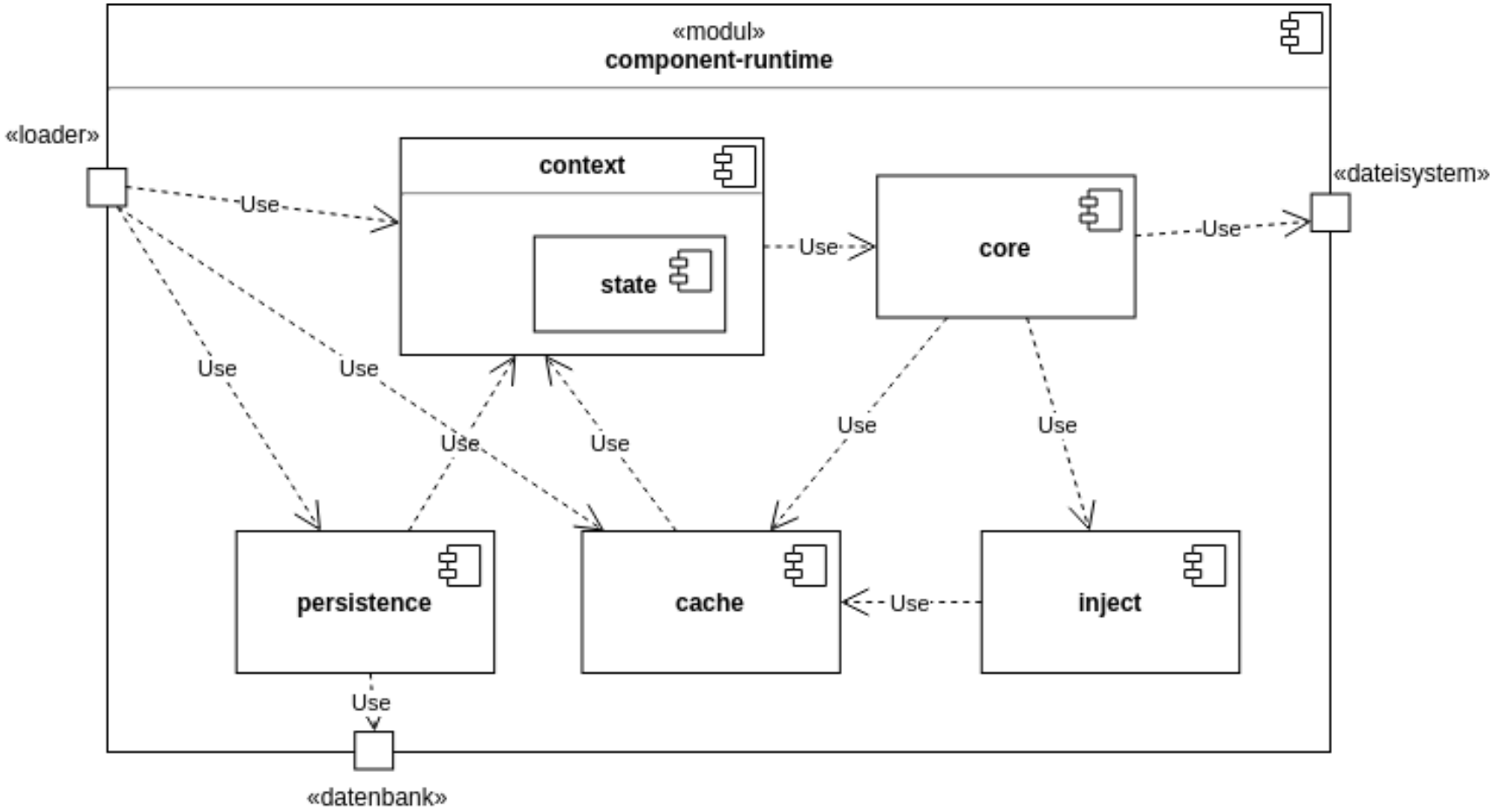
Architektur: Kontextsicht



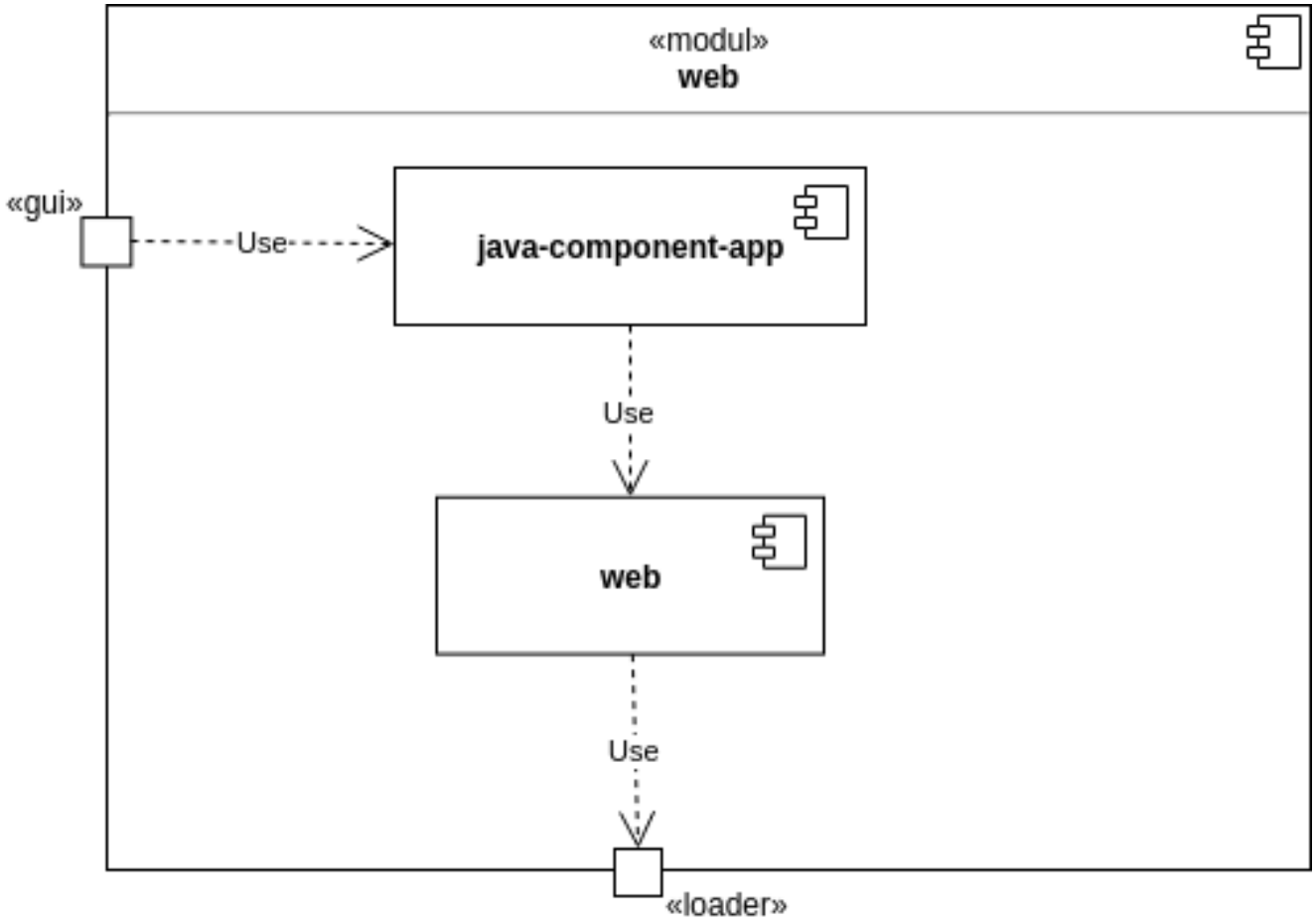
Architektur: Bausteinsicht - Level 1



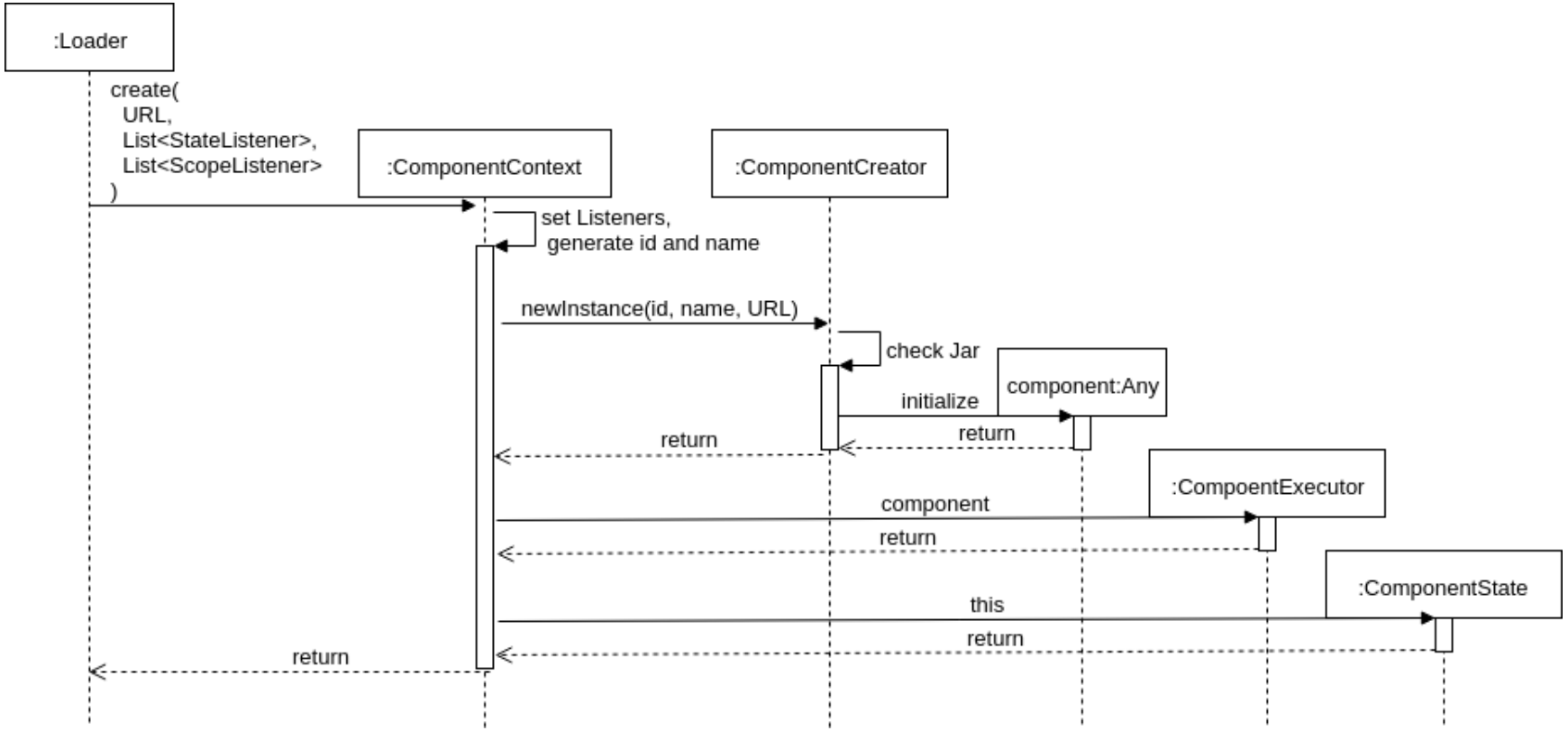
Architektur: Bausteinsicht - Level 2.cre



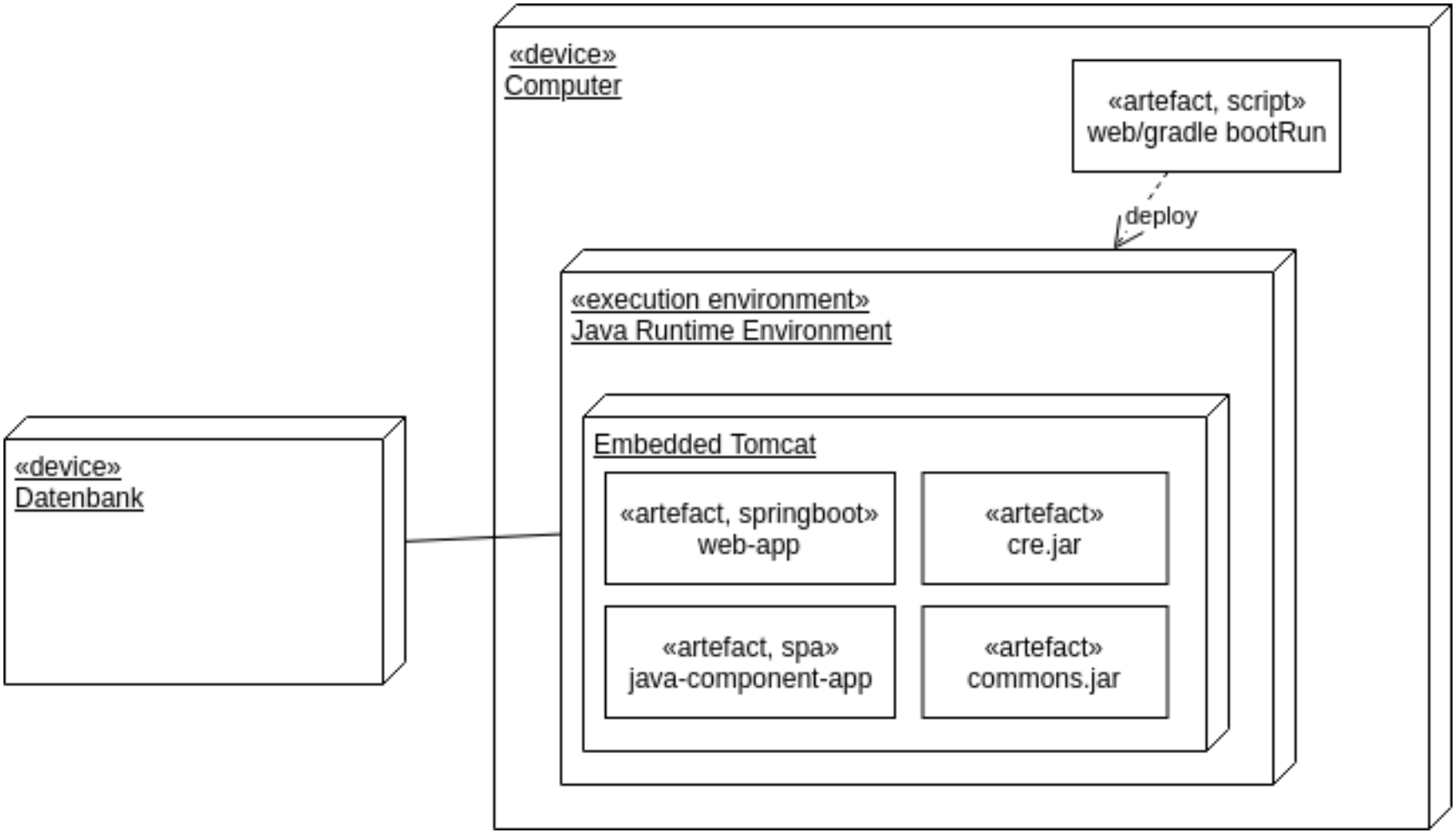
Architektur: Bausteinsicht - Level 2.web



Architektur: Laufzeitsicht



Architektur: Verteilungssicht



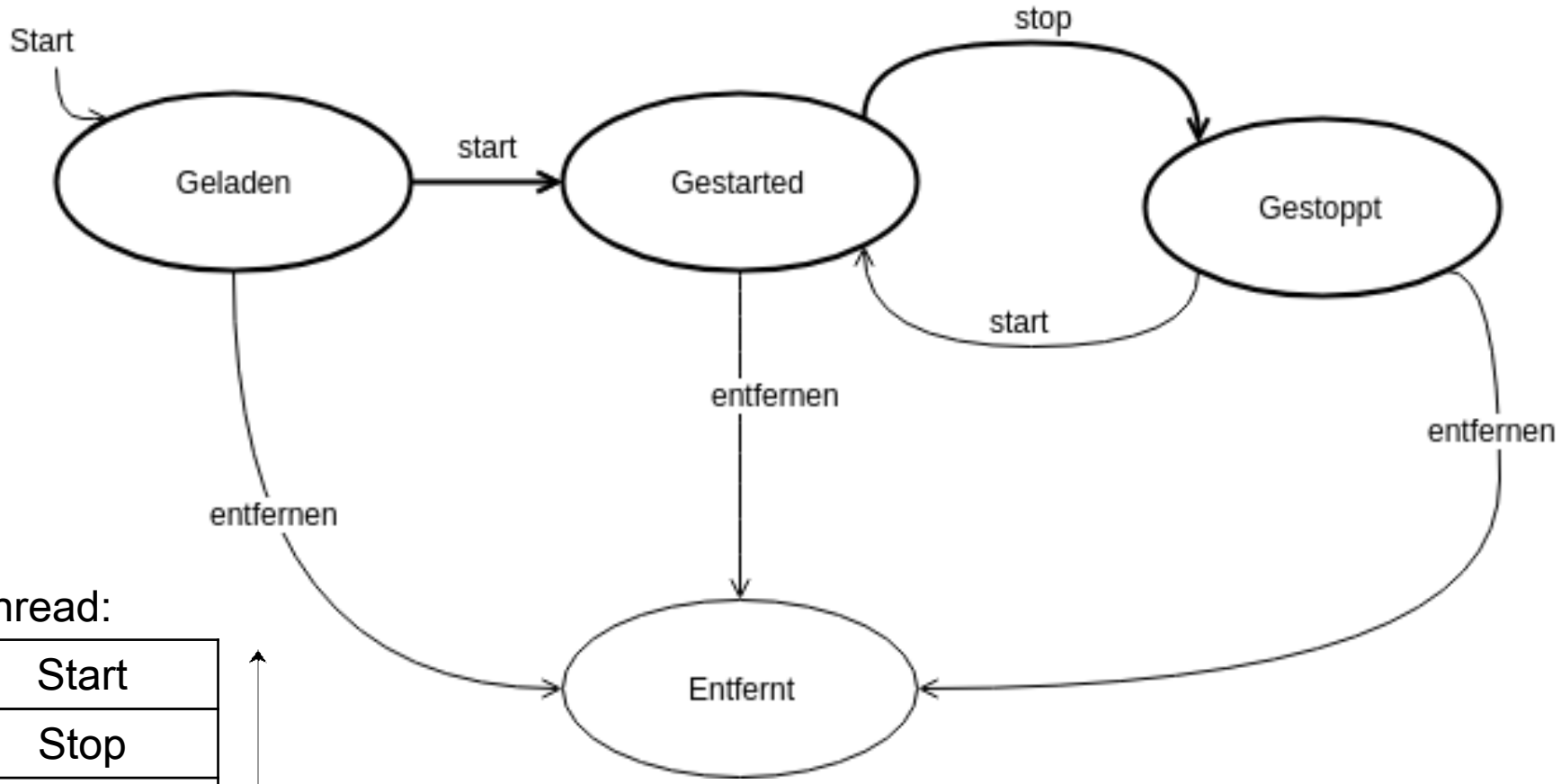
Entscheidung: Nebenläufigkeit

Komponentenentwickler entscheidet über Nebenläufigkeit innerhalb einer Komponente

» Start-, Stop-, Observer-Methode **einer** Komponente werden in **einem** Thread und **vollständig*** ausgeführt

*) Ausnahme beim Entfernen einer Komponente

Single-Thread



Thread:

Start
Stop
Event
Start

Entscheidung: Class-Loader

Bedingungen bei der Komponentenentwicklung möglichst identisch zu denen in der LZU

» Keine* Überdeckung von Abhängigkeiten

» Jede Komponente lädt ihre Klassen mit eigenen* Class-Loader

*) Ausnahme bei Injektion

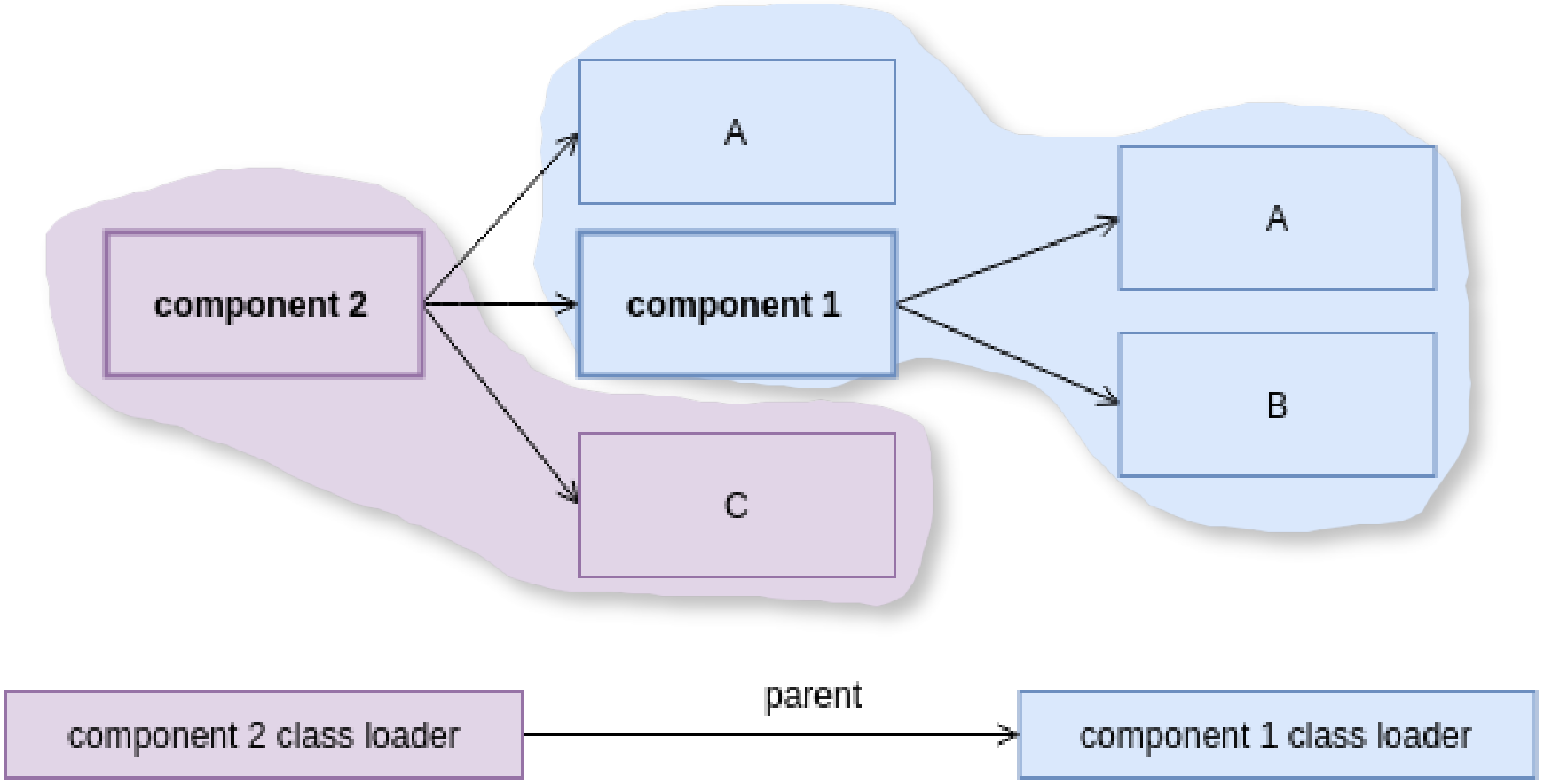
Entscheidung: Kompositionen

Eigner Class-Loader

» Referenzierte Komponenten müssen bereits geladen sein

» Referenzierte Komponenten müssen eindeutig sein

Injection - Class-Loading

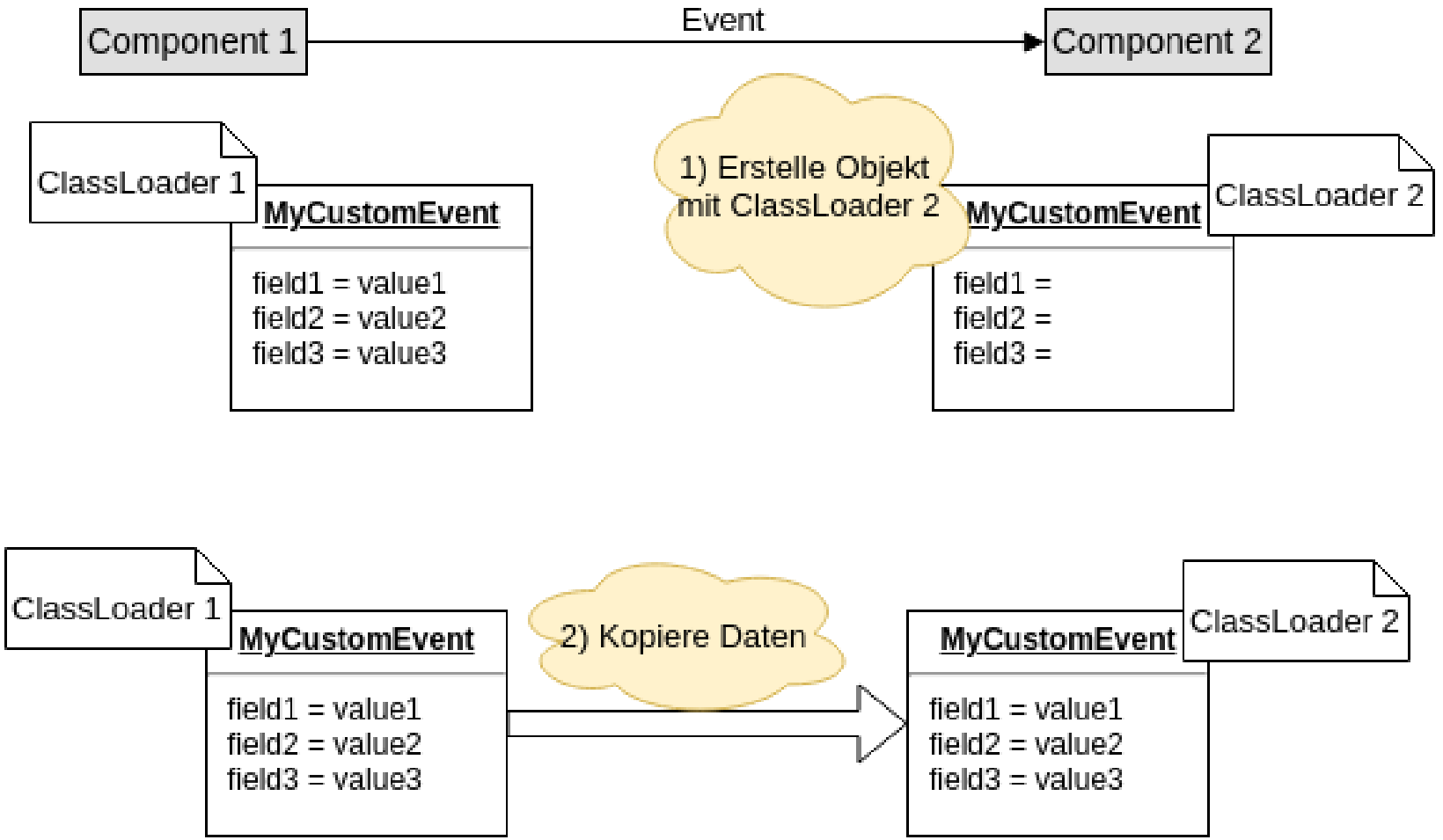


Entscheidung: Events

Events übertragen Daten/Informationen

» Übertrage Objekte vom gleichen Typ, die mit unterschiedlichen Class-Loader geladen wurden

Events


Component 1
Event
Component 2
ClassLoader 1
MyCustomEvent
field1 = value1
field2 = value2
field3 = value3
1) Erstelle Objekt
mit ClassLoader 2
MyCustomEvent
field1 =
field2 =
field3 =
ClassLoader 2
ClassLoader 1
MyCustomEvent
field1 = value1
field2 = value2
field3 = value3
2) Kopiere Daten
MyCustomEvent
field1 = value1
field2 = value2
field3 = value3
ClassLoader 2

Klassifizierungsframework

- Lifecycle
  - Implementation: Kotlin
  - Packaging: Jar-Package
  - Deployment: Run-time
- Construction - Interface Specification
  - Interface Type: Operation-based
  - Interface Language: Kotlin + Annotations
  - Interface Levels: Syntactic

Klassifizierungsframework

- Construction - Binding
  - Type: Vertical
  - Composition: Endogenous
- Interactions
  - Interaction Style: Sender-Receiver
  - Communication Type: Asynchronous

Fazit

- „Lessons Learned“
  - Arbeiten mit Modulen
  - Java Class-Loading
- Ausblick
  - Front-End erweitern
- Kritisch Betrachtung
  - Injektion / Class-Loading