

IuK Projekt RAJA
“**Regelbasiertes und echtzeitfähiges
Autorisierungsframework auf Java Basis**”

13. Januar 2009

Dipl. Inf. Ludwig Adam (adaml@pst.ifi.lmu.de)

Prof. Dr. Rolf Hennicker (hennicke@pst.ifi.lmu.de)

Agenda

- (Das asynchrone Java/A Komponentenmodell)
- IuK-Projekt Vorstellung
- Echtzeitanforderungen im Komponentenmodell
 - Projektbezug
 - RT Constraints im Komponentenmodell
 - Zeitliche Aspekte im Komponentenmodell
 - Implementierung
- Unterstützung durch Palladio
 - Verifikation bzgl. RT Constraints
 - Bestimmung der Performance-Kapazität einer Komponente

IuK Projektvorstellung

- IuK Projekt “Entwicklung eines regelbasierten und echtzeitfähigen Autorisierungsframeworks auf Java Basis”
 - Kooperation zwischen LMU München und petaFuel GmbH, Freising
 - Projektlaufzeit 3 Jahre
 - Ziel: Verwendung des Frameworks zur Autorisierung von Kreditkartentransaktionen
- Software-Engineering:
 - Komponentenorientierte Architektur des Systems auf Basis des Java/A Komponentenmodells
 - Implementierung eines Komponentenframeworks in Real Time (RT) Java
- Forschungsrelevante Themen
 - Spezifikation von Echtzeitconstraints für Komponentenverhalten
 - Echtzeitbedingungen bei asynchroner Kommunikation in verteilten Systemen
 - Implementierungsmethodiken für RT Java
 - Verifikation von RT Constraints

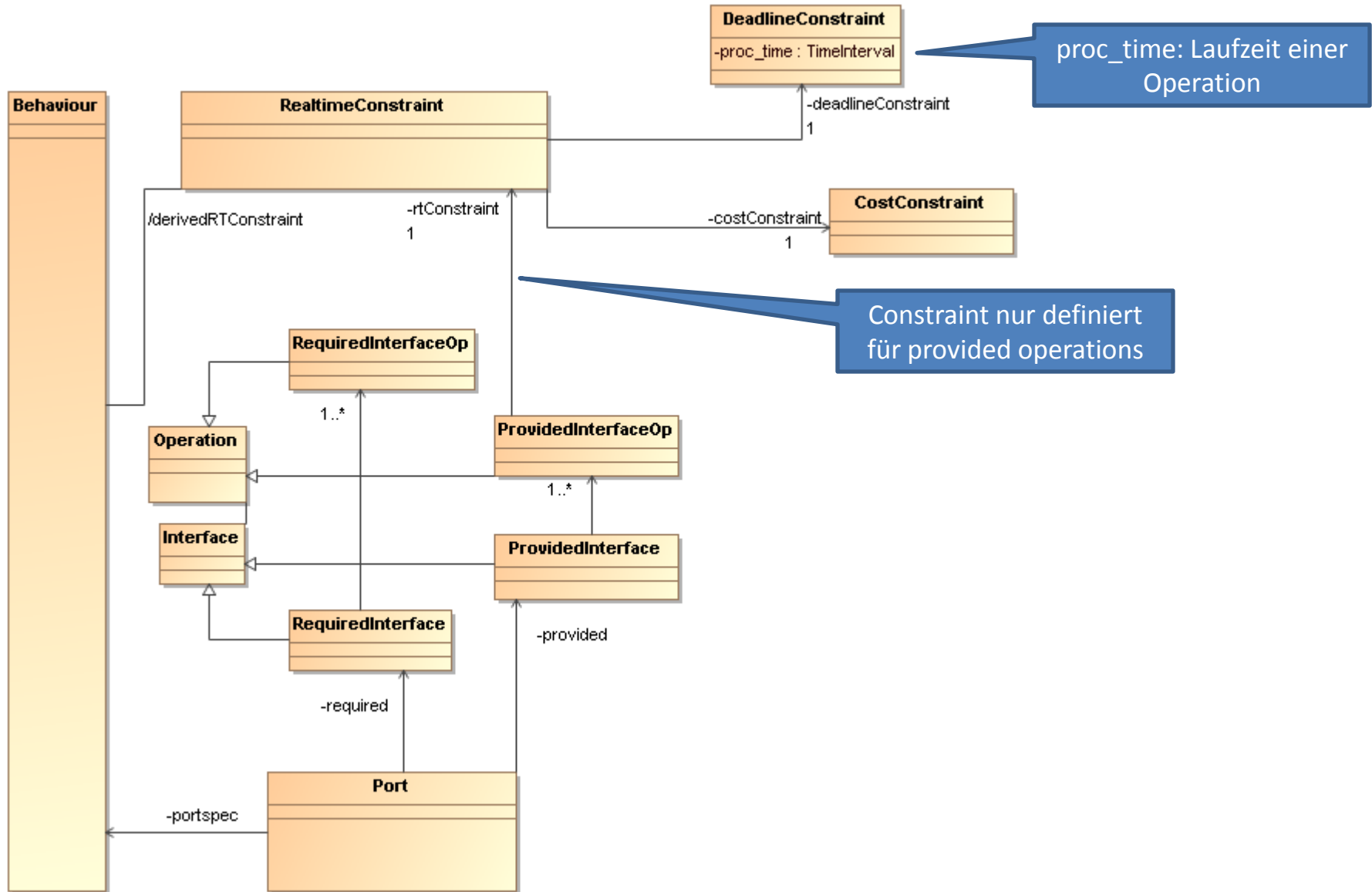
Echtzeitaspekte im Java/A Komponentenmodell

Projektbezug: Zugesicherte Laufzeit einer Autorisierung

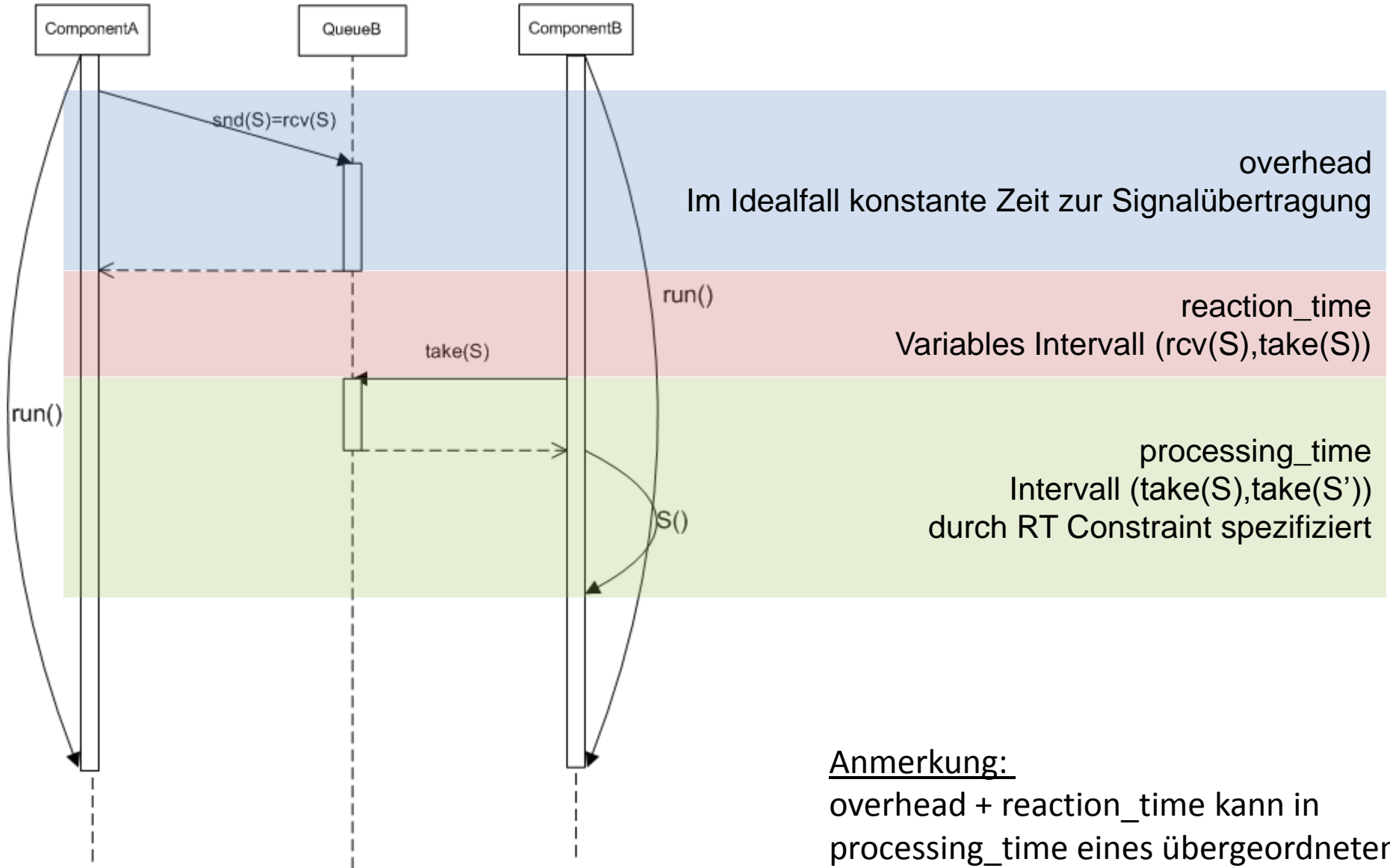
Drei Betrachtungsgebiete:

1. Erweiterung des asynchronen Komponentenmodelles um Real Time (RT) Constraints
2. Betrachtung der Echtzeit-Aspekte im Komponentenverhalten.
3. Implementierung in RT Java

1. Erweiterung des asynchronen Komponentenmodelles um RT Constraints (Ausschnitt)



2. Betrachtung der zeitliche Aspekte im Komponentenverhalten



Anmerkung:

overhead + reaction_time kann in processing_time eines übergeordneten Services enthalten sein

3. Implementierung in RT Java

JSR001: Real-Time Specification for Java (RTSJ):

- Definition von RT Constraints in Form von Deadlines und CPU-Kosten für Threads möglich
 - Definition von `javax.realtime.RealtimeThread`
 - Überwachung der Constraints: Übergabe eines `CostMissHandlers` bzw. `DeadlineMissHandler` zur Behandlung von Überschreitungen
- Spezifikation unterstützt Feasibility-Analysen (d.h. Exception falls Deadline nicht erfüllt werden kann)
- Verwendete Implementierung: JamaicaVM der aicas GmbH, Karlsruhe

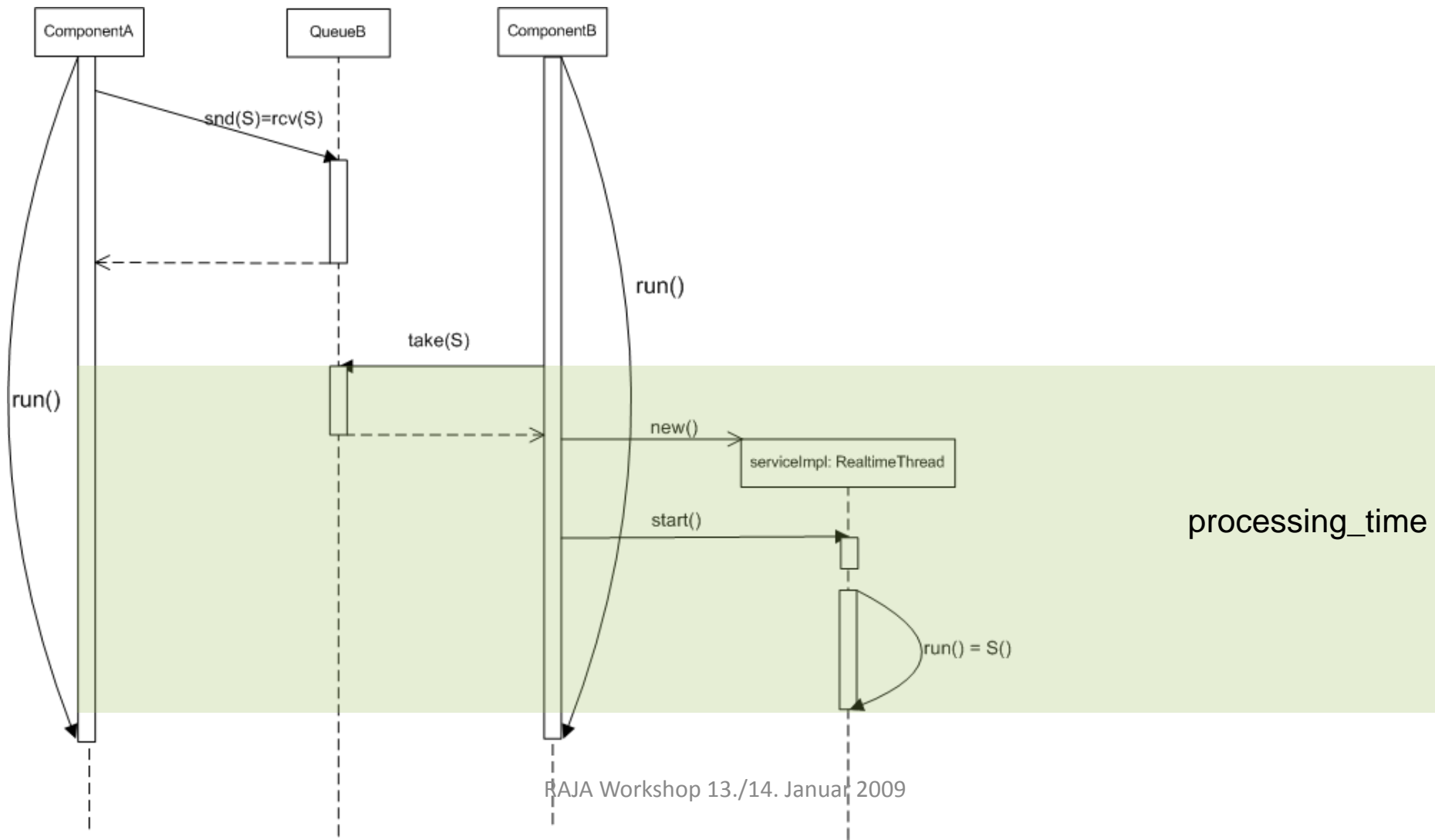
```
SchedulingParameters schedulingParams = new PriorityParameters(RealtimeThread.NORM_PRIORITY + 2);
ReleaseParameters releaseParams = new AperiodicParameters();
releaseParams.setDeadline(new RelativeTime(1000, 0)); // 1 second deadline
releaseParams.setDeadlineMissHandler(new AsyncEventHandler(missHandler));

RealtimeThread myThread= new RealtimeThread(schedulingParams,releaseParams,null,null,null,someRunnable);
```

Problematik der Servicedelegation in RT Java

Problematik: Constraints nur für run()-Methode von `RealtimeThread` nutzbar.

→ Damit Delegation der Ausführung eines Services an eigenen Thread notwendig



Problematik der Servicedelegation (II)

- Durch die Servicedelegation wird das Verhalten der Komponente durch mehrere Threads definiert
- Fragestellungen:
 - Äquivalenz des Verhaltens?
 - Nebenläufigkeit bei mehreren Portinstanzen / mehreren Signalen auf einer PortQueue

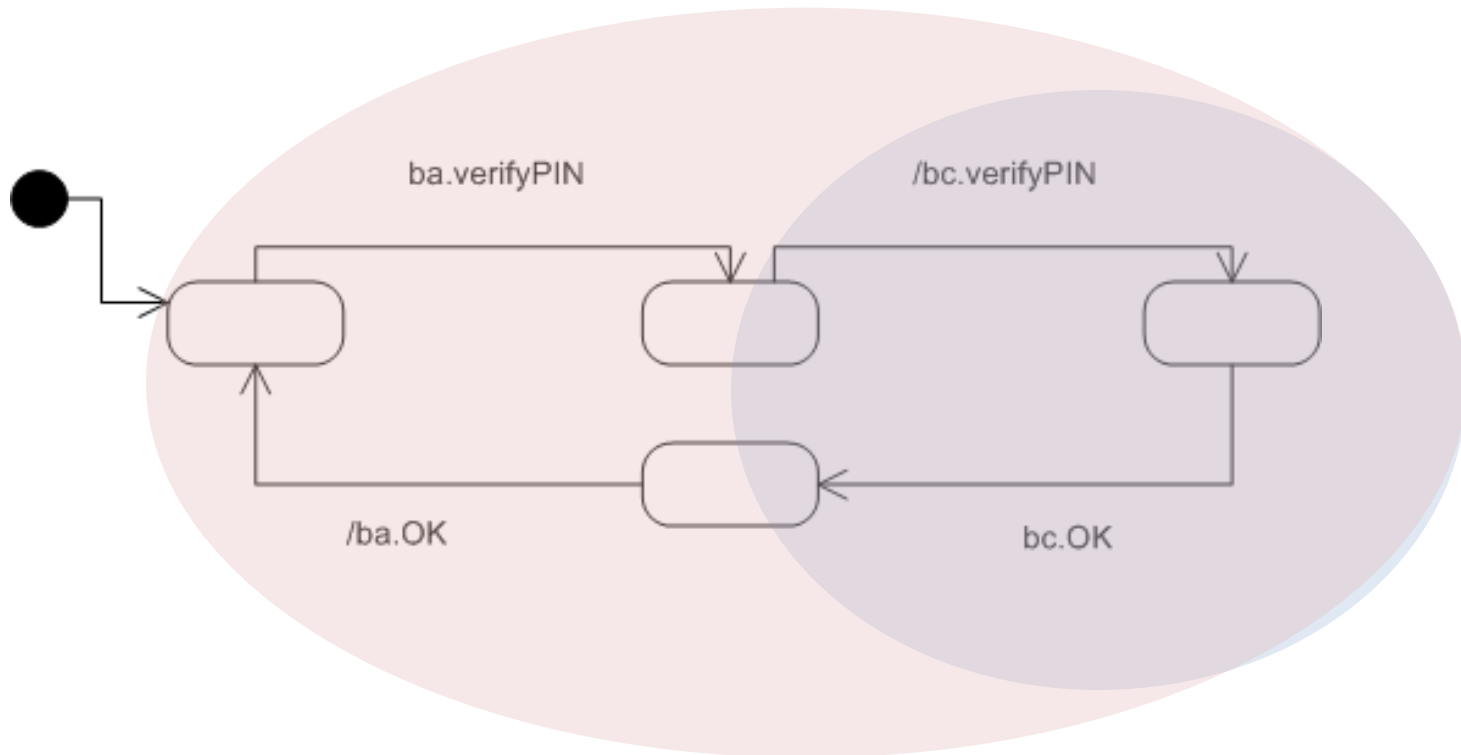
Verifikation: Unterstützung durch Palladio

- Grundlegendes Ziel: Einsatz von Palladio zur Verifikation der RT Constraints in der Implementierung
- Denkansatz
 - Modellierung des Systems im Palladio Model
 - Anwendung von Palladio's Performance Prediction
 - Falls Palladio's Performance Prediction < RT Constraints => Implementierung OK
- Ziele im Speziellen:
 - Verifikation proc_time:

Ist ein RT Constraint für die Prozessierung eines unabhängigen Services überhaupt realisierbar?
 - Bestimmung / Verifikation der lastabhängigen / nutzungsabhängigen reaction_time:

Insbesondere Performance-Kapazität einer Komponente:

Unter welchen Bedingungen wird der RT Constraint eines übergeordneten, abhängigen Services aufgrund von zu hoher reaction_time des untergeordneten Services verletzt?



**RT Constraint für `ba.verifyPIN` schließt RT Constraint `cb.verifyPIN` mit ein
 → Problem, falls `reaction_time (cb.verifyPIN)` zu hoch ist**

Vielen Dank

- Nähere Infos
 - <http://www.pst.ifi.lmu.de/Forschung/laufende-projekte/raja/>
 - <http://www.petaFuel.de>
 - eMail:
 - adaml@pst.ifi.lmu.de
 - hennicke@pst.ifi.lmu.de