Einsatz von Java-Komponenten in verteilten Embedded Systems

Uwe Rastofer,
Ulrich Gall, Frank Schinkmann, Bernd Hindel, Jürgen Kleinöder

3SOFT GmbH, Erlangen
Informatik 4, Friedrich-Alexander-Universität Erlangen-Nürnberg

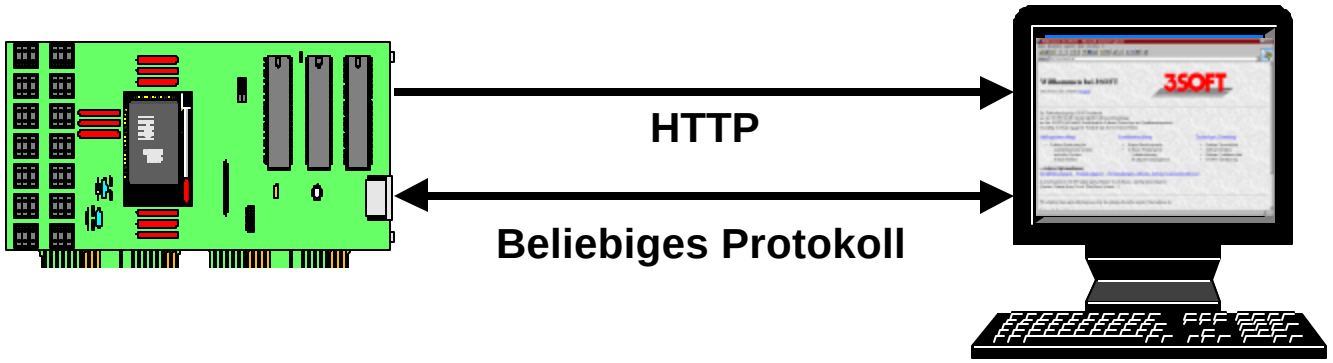
rastofer@3SOFT.de
http://www.3SOFT.de/

Überblick

- Entwicklungstrends bei Embedded Systems
- Embedded Systems als Komponenten
- Stellvertreterkomponenten
- Verteiltes Komponentenmodell
- Zusammenfassung und Ausblick

Bedienen und Beobachten

- Bisher meist proprietäre Softwarelösungen
- Übergang zu Standards (HTML, HTTP, Java)



Embedded System

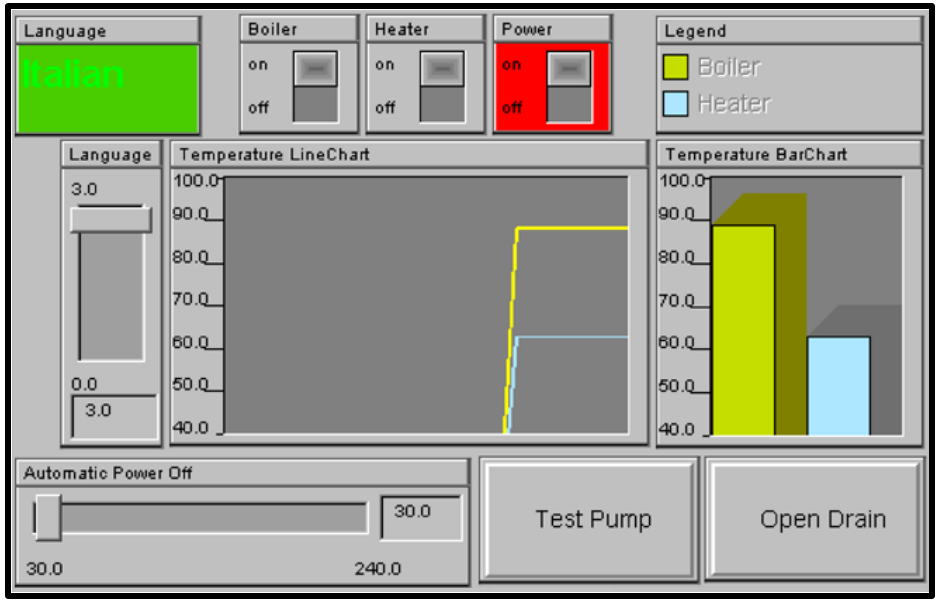
spezielle Service-Tasks
Embedded-HTTP-Server

Bedienrechner

Standard-Web-Browser
mit Java

B&B-Oberflächen

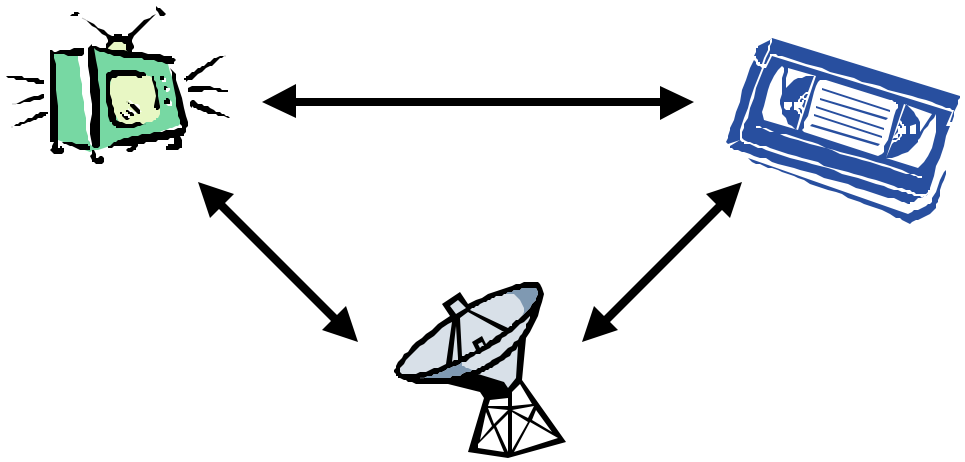
- Anzeige- und Bedienelemente sind wiederverwendbare Komponenten



➔ Wie tauschen Bedienkomponenten und Embedded System Daten aus?

Verteilte Embedded Systems

- Embedded Systems müssen auch untereinander kommunizieren
  - Beispiel: Abstimmung von Fernsehgerät, Videorecorder und Satellitenreceiver



➔ Über welches Protokoll findet die Kommunikation zwischen beliebigen Embedded Systems statt?

Geräte als Komponenten

- Komponentenmodell stellt Mechanismen zum Datenaustausch zur Verfügung

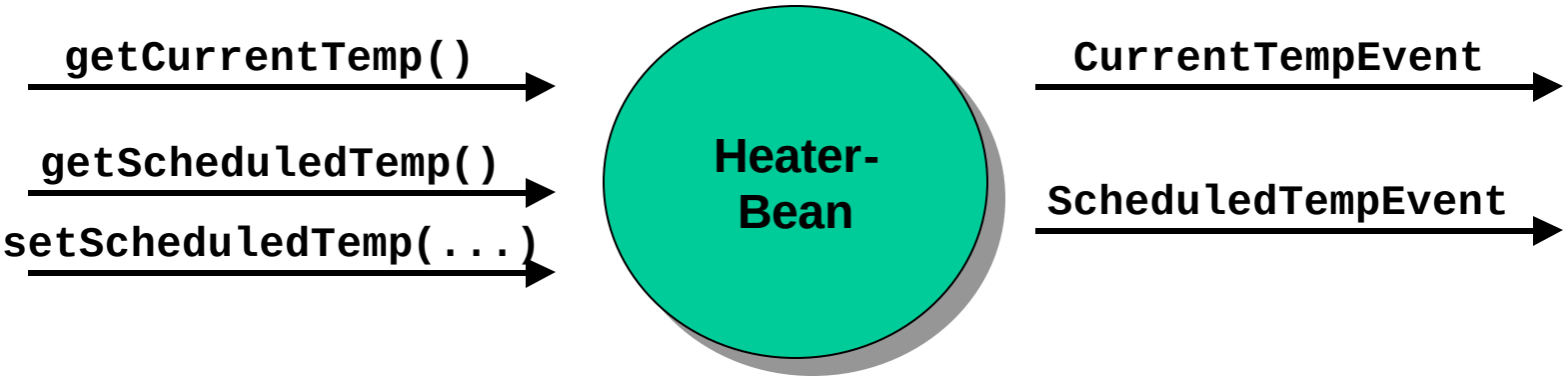
➔ Embedded System ist selbst eine Komponente

- Gerätekomponente kapselt alle Eigenschaften des zu steuernden Embedded Systems
- Gerätekomponente kann mit anderen Komponenten verbunden werden

- Umsetzung auf JavaBeans
  - aktuelle Werte des Geräts ⮜ nicht veränderbare Bean-Properties
  - Regelgrößen des Geräts ⮜ veränderbare Bean-Properties
  - Änderung aktueller Werte oder Regelgrößen ⮜ Events

Geräte als Komponenten (2)

- Einfaches Beispiel: Heizelement
  - aktuelle Temperatur: nicht veränderbare Property **`CurrentTemp`**
  - eingestellte Temperatur: veränderbare Property **`ScheduledTemp`**
  - Zustandsänderungen: Events **`CurrentTempEvent`** und **`ScheduledTempEvent`**

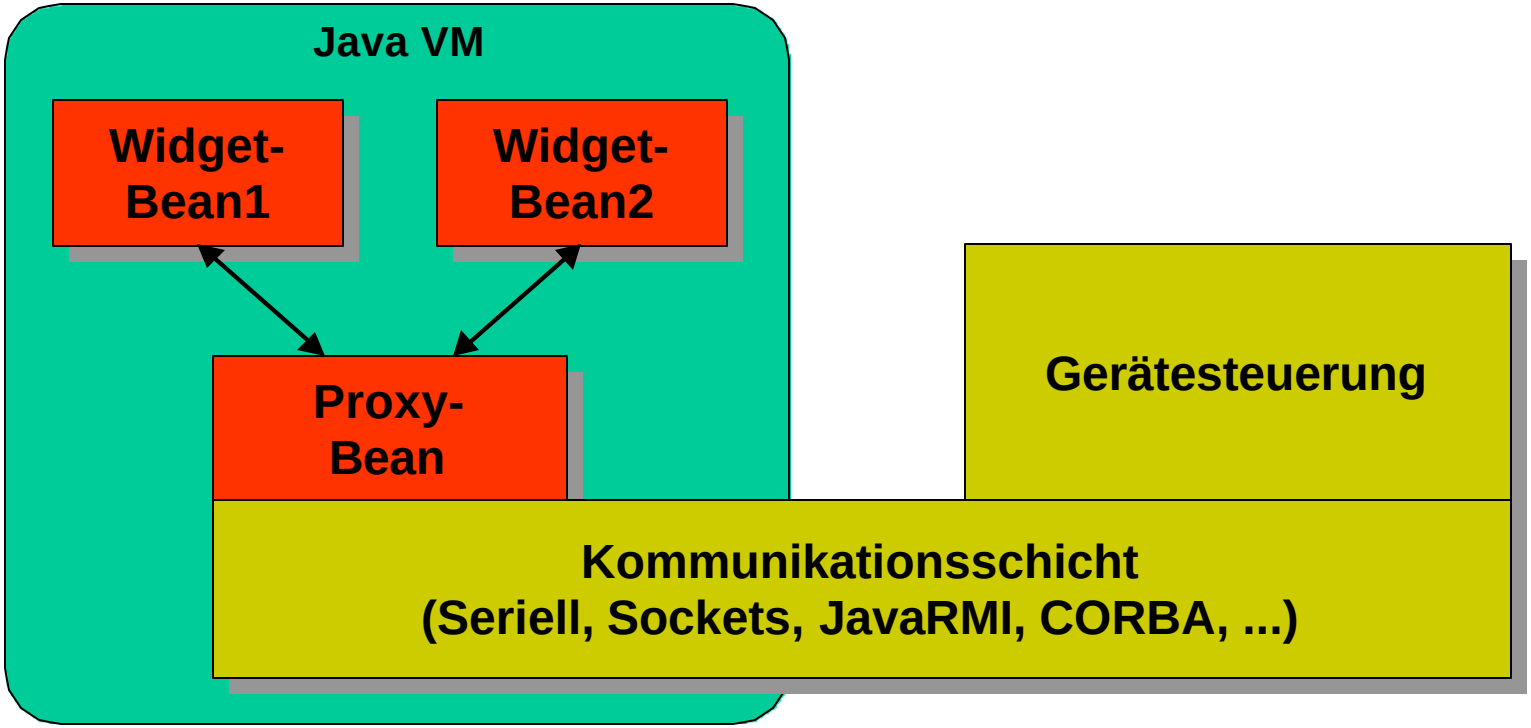


Verteilte Embedded Systems

- Beschränkung von JavaBeans
  - alle miteinander kommunizierenden Beans laufen in einer Java-Maschine ab
  - ➔ lokale Event-Behandlung
- Probleme bei der Anbindung der Gerätesteuerung
  - Oberflächenkomponenten laufen in Applets beim Benutzer ab
  - Steuerungskomponenten liegen nicht auf dem Embedded System
  - Gerätesteuerung muß auf dem Embedded System sein

➔ Lokale Kommunikation nicht ausreichend

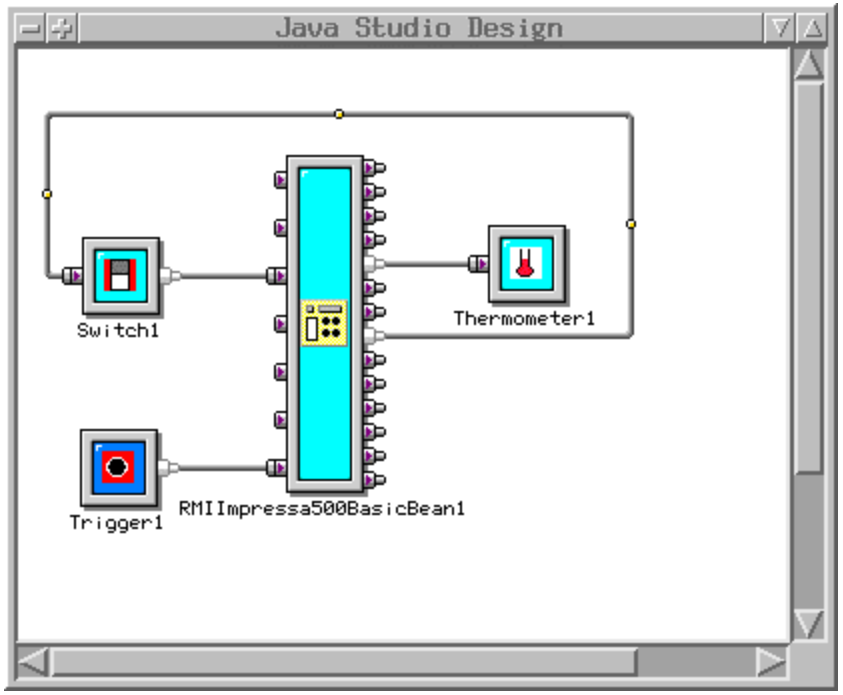
Proxy-Beans

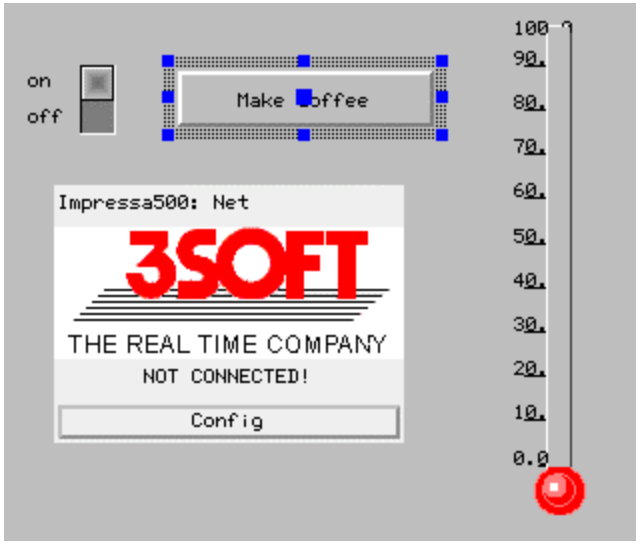


- Lokale Stellvertreter in JavaBeans = Proxy-Beans
- Verwenden beliebiges Protokoll zur Kommunikation
- Kommunikation für andere Beans unsichtbar

Beispiel - Kaffeemaschine

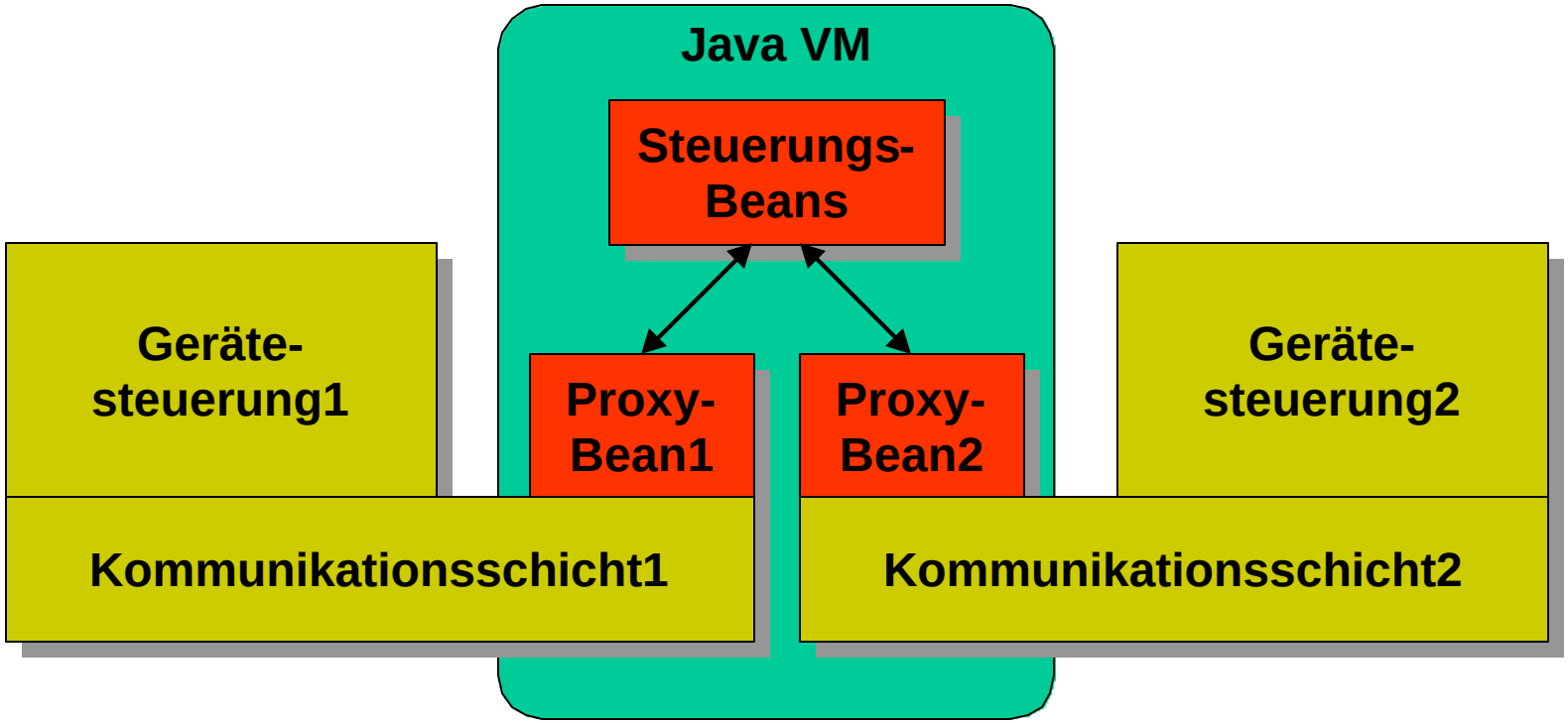
- Proxy-Bean zur Kommunikation mit einer Kaffeemaschine
  - über Embedded HTTP-Server direkt ladbar
  - alternativ: über WWW-Server des Herstellers





Komplexe Steuerungen

- Autonome Steuerungs- und Regelungsprogramme
- Steuerungs-Beans implementieren Steuerungslogik



Verteiltes Komponentenmodell

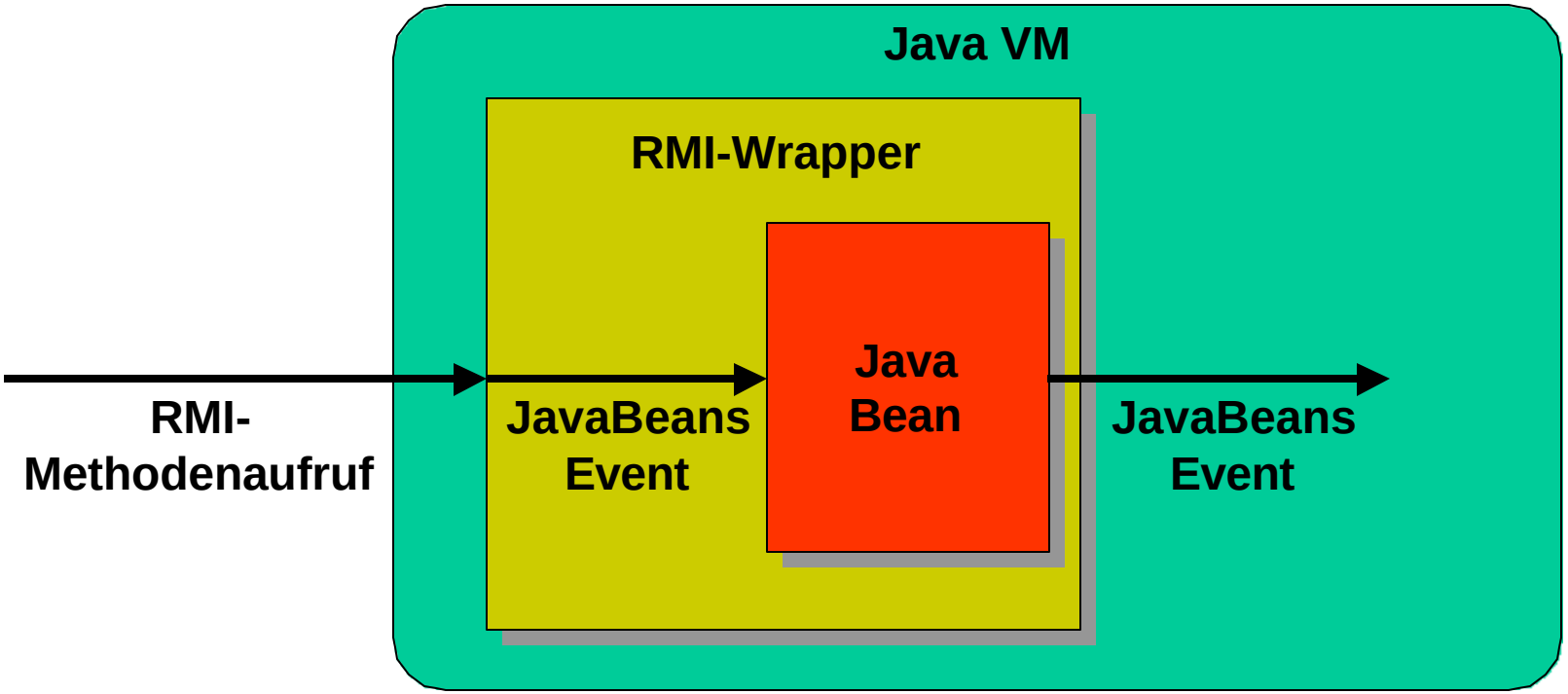
- Nachteile von Stellvertreterkomponenten
  - alle Proxy-Beans laufen in einer zentralen Java-Maschine ab
  - mehrere Proxy-Beans für dasselbe Embedded System müssen Zustand austauschen
  - keine direkte Kommunikation zwischen den Gerätesteuerungen

➔ Alternativer Ansatz: Verteiltes Komponentenmodell

- Geräte-Komponenten liegen auf dem Embedded System selbst
  - Embedded Java
  - Java-Prozessoren (?)
- Zustellung der Events über JavaRMI-Aufrufe
- Kompatibilität zu JavaBeans

RMI-Beans

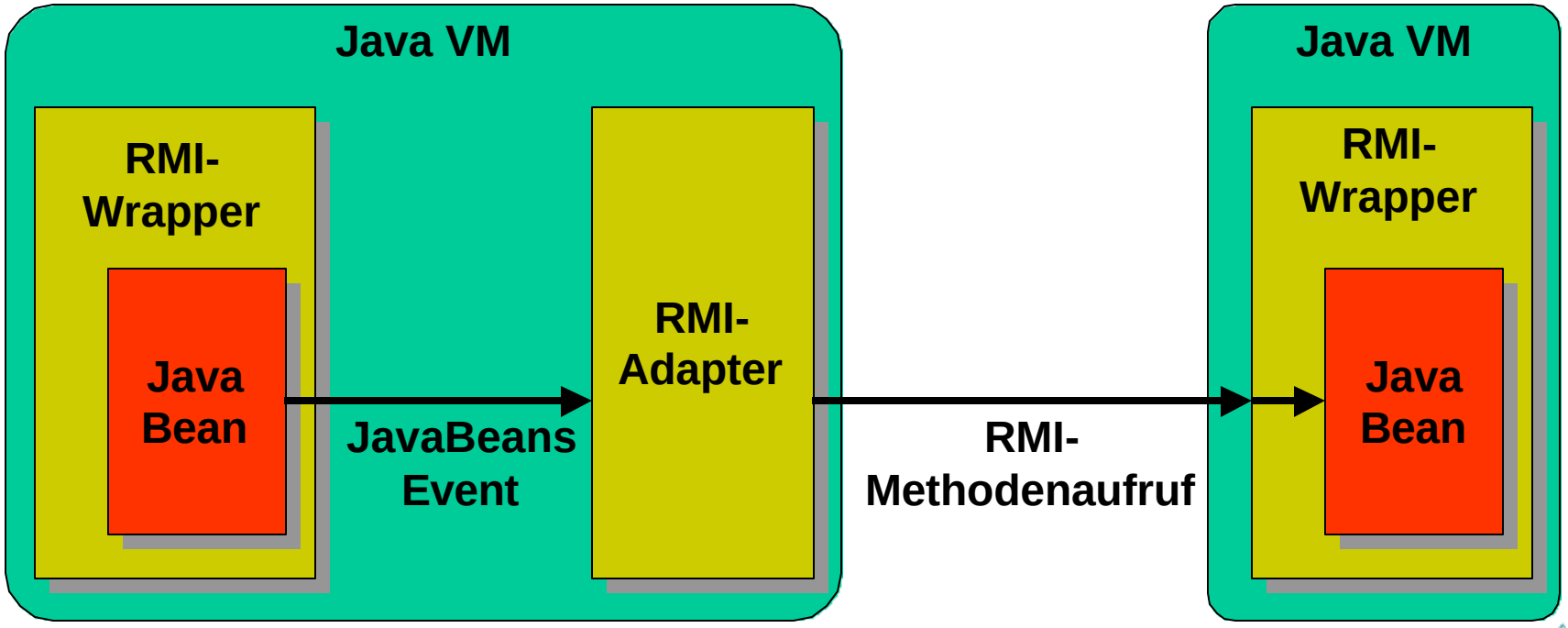
- Verpacken von JavaBeans in RMI-Wrapper



- Automatische Erzeugung des Wrappers aus der JavaBean

RMI-Adapter

- Finden der Ziel-Bean
- Durchführung der RMI-Aufrufe
- Behandlung der RMI-Exceptions



Weitere Dienste

- Bean-Factory
  - Erzeugung von RMI-Beans auf den beteiligten Rechnern
  - Erzeugung der RMI-Adapter bei den RMI-Beans
- Verteilter Namensdienst
  - RMI-Registry nicht hierarchisch
  - eigener Namensdienst, ähnlich zum Naming Service in CORBA
- Entwicklungswerkzeug (Builder-Tool)
  - verfügbare Werkzeuge ungeeignet
  - können keine RMI-Wrapper und RMI-Adapter erzeugen
  - daher Entwicklung eines eigenen Builder-Tools

Zusammenfassung u. Ausblick

- Embedded Systems als Komponenten modellierbar
  - Beispiel: JavaBeans
- Kommunikation mit dem physischen Gerät
- Stellvertreterkomponenten
  - Verbergen der Kommunikation
  - Umsetzung auf JavaBeans-Events
- Verteiltes Komponentenmodell
  - Übertragung von JavaBeans-Events mit JavaRMI
  - Entwicklung eines geeigneten Builder-Tools
- Aktuelle Entwicklungen:
  - CORBA Component Model
  - Suns Jini-Architektur