Distributed Object Model Environment

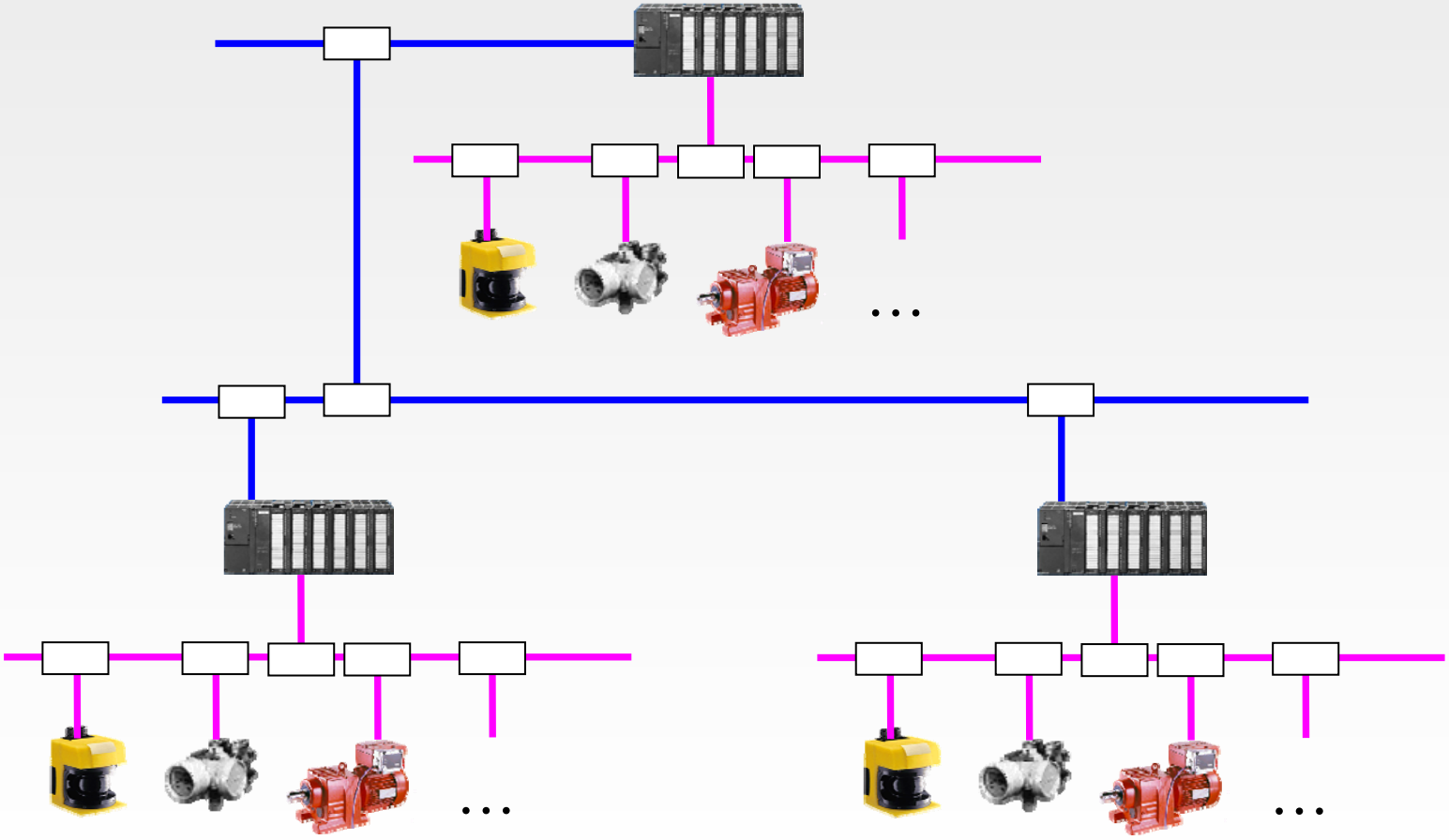
Ein objektorientiertes Softwaremodell für verteilte Automatisierungssysteme

Matthias Riedl
ifak e.V. Magdeburg
Werner-Heisenberg-Str. 1, 39106 Magdeburg
www.ifak.eu
matthias.riedl@ifak.eu

Gliederung

1. Motivation
2. Stand der Technik
3. Lösungsansatz / Validation
4. Zusammenfassung / Ausblick

Motivation
Ausgangszustand

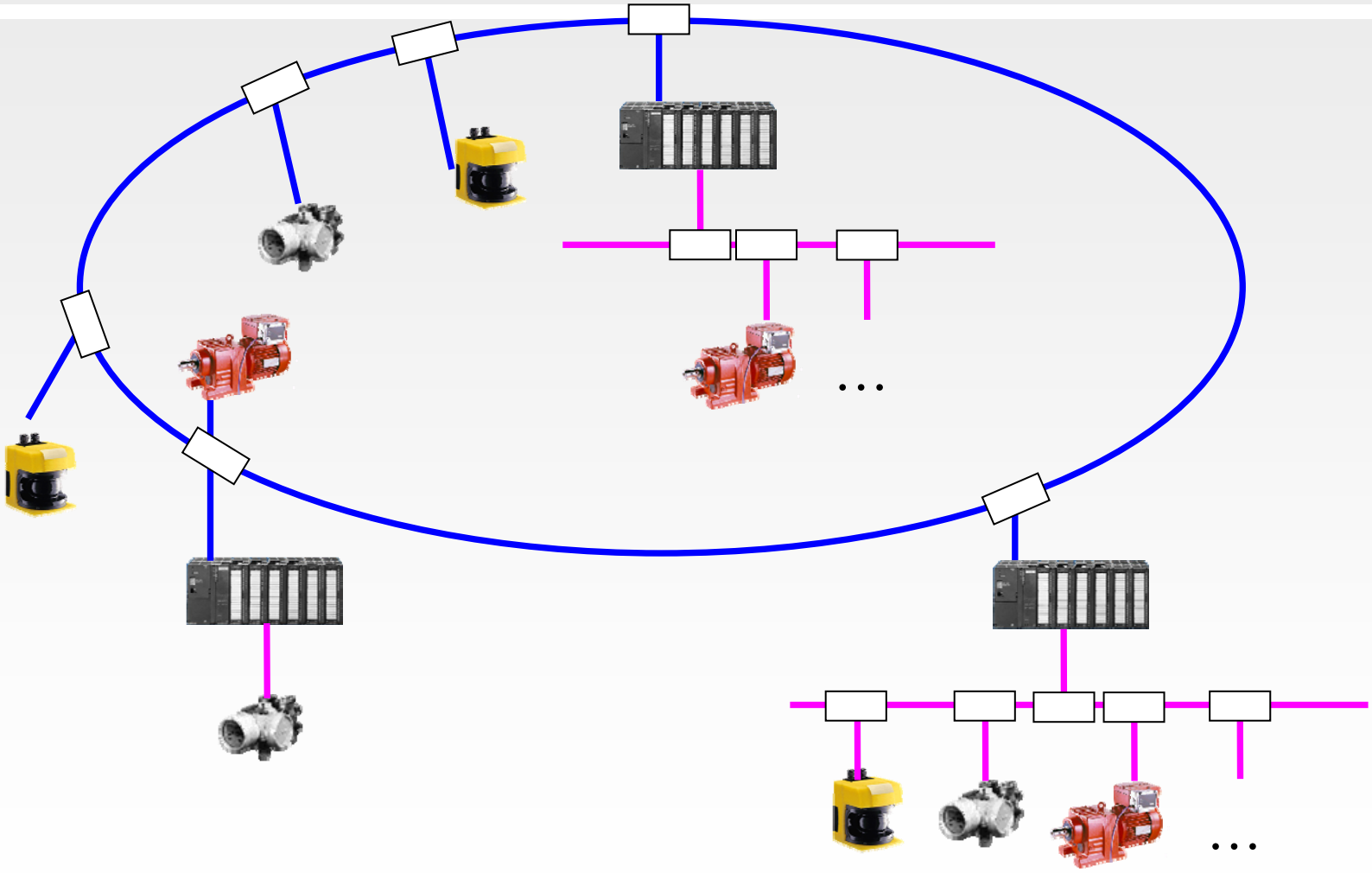


Motivation

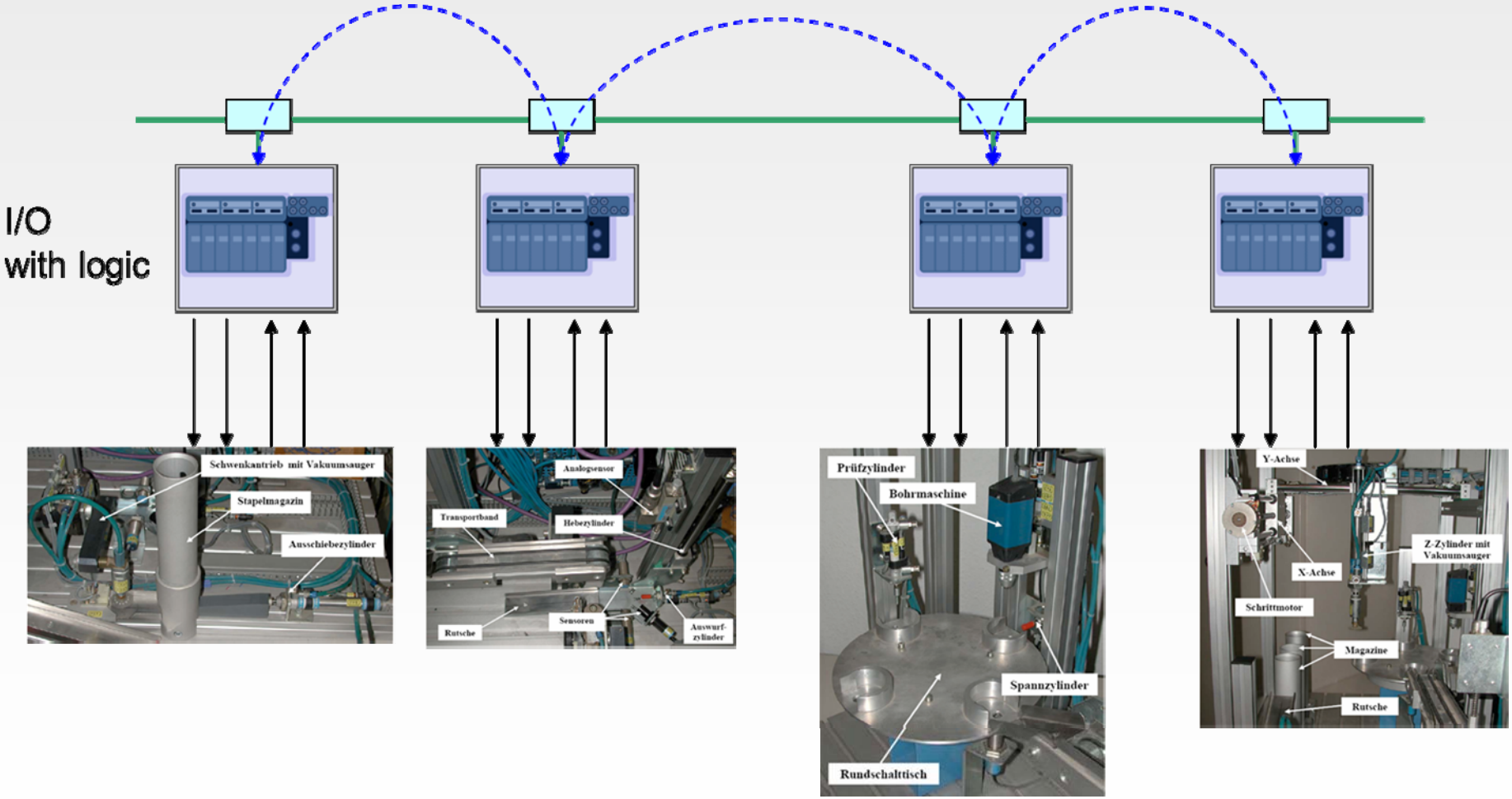
Gründe der Verteilung / Dezentralisierung

- Verarbeitungsleistung in den E/A-Geräten verfügbar (z.B. Antriebe mit Technologiefunktionen, „intelligente“ Ventilinseln)
- Digitale Kopplung zwischen den Automatisierungsgeräten ermöglicht umfangreicheren Datentransport zwischen diesen
- Hierarchiebildung durch Zwischenebenen mit Verarbeitungsleistung (Remote IO)
- Proxykonzept verursacht erhöhtes Datenaufkommen auf dem Bus

Motivation
Ziel: Verteiltes System

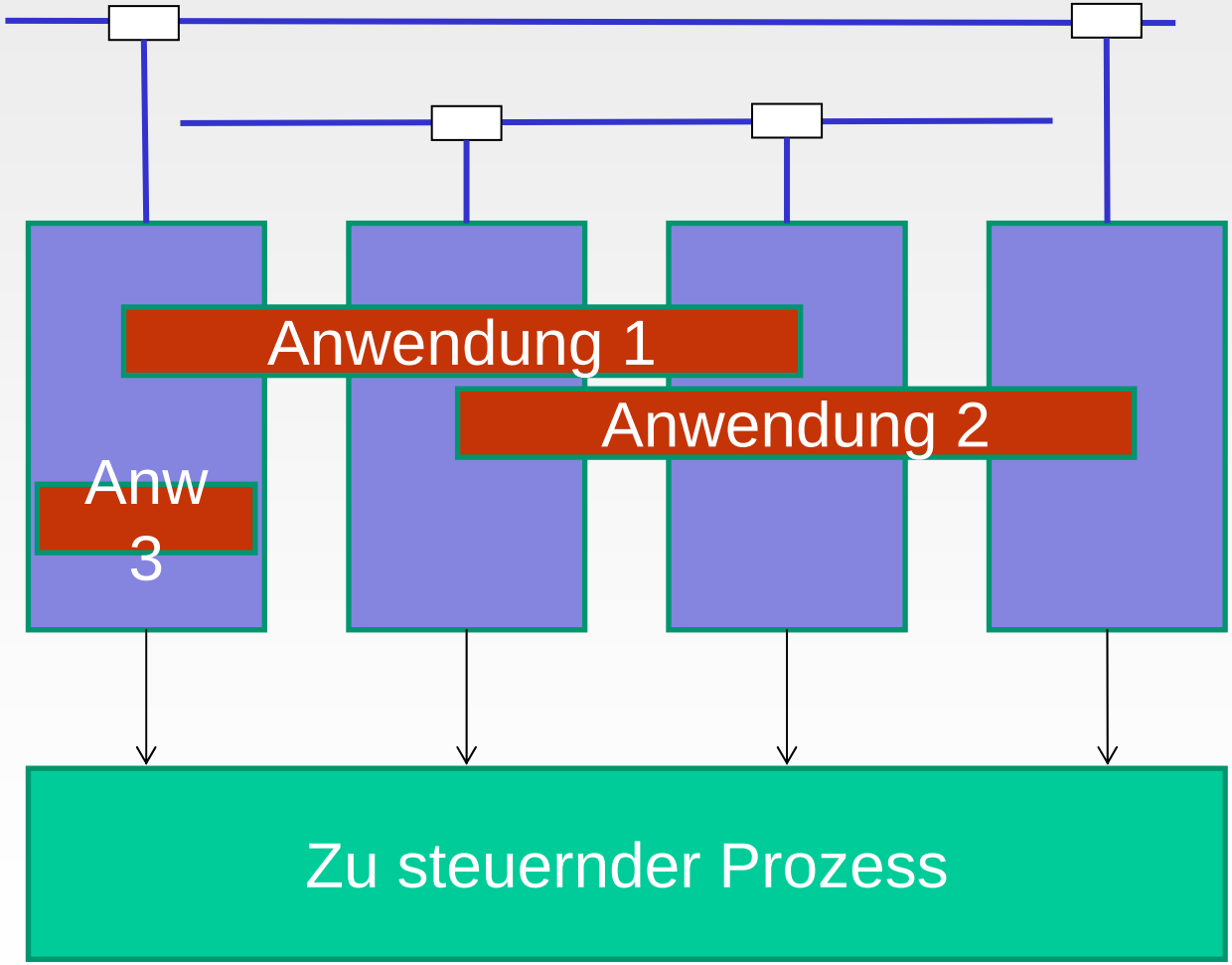


Motivation
Ziel: Verteiltes System

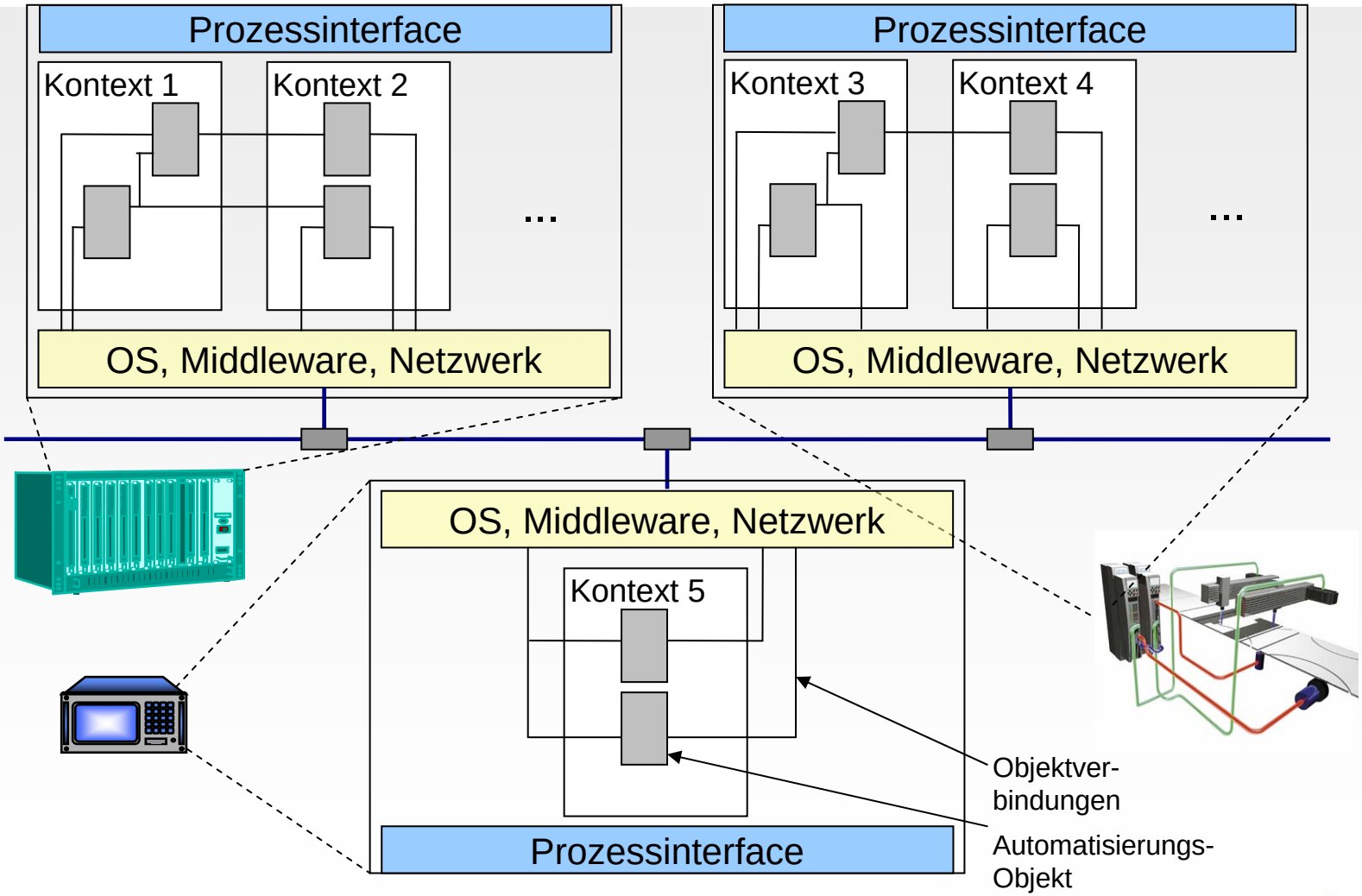


Motivation

Ziel: Verteiltes System (IEC 61499)



Motivation
Verteiltes Automatisierungssystem



DOME - Distributed Object Model Environment

Motivation

Konsequenzen der Verteilung

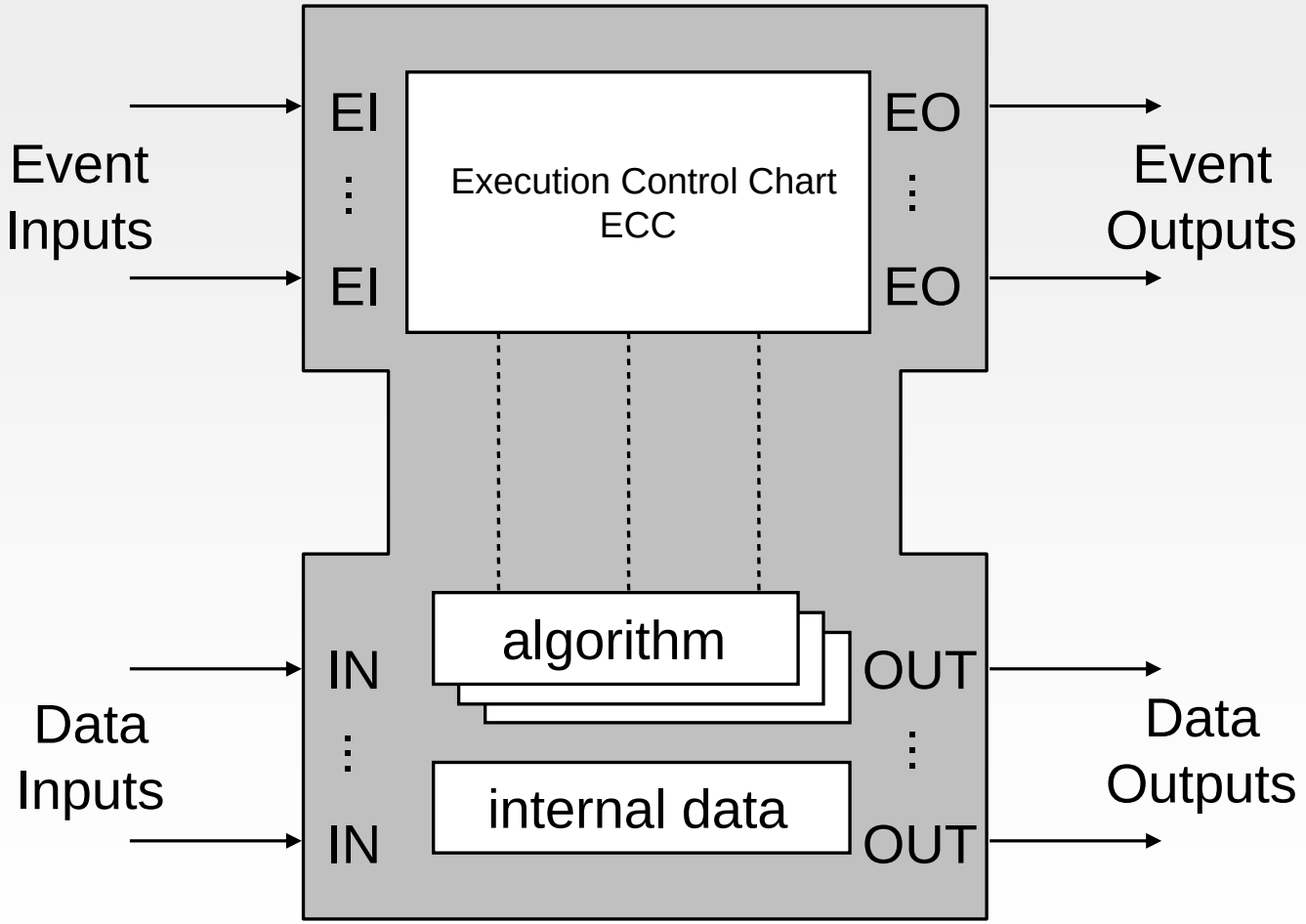
- Veränderung des Engineering Konzeptes, Funktionsverteilung mit zurzeit verschiedenen Werkzeugen
- Offene Systeme sind schwerer zu beherrschen als geschlossene
  - Gewährleistung der Verfügbarkeit und der Zuverlässigkeit
  - Beschreibung semantisch eindeutiger Schnittstellen für alle Funktionalitäten ist zurzeit nicht möglich
  - Definition der technischen und organisatorischen Verantwortung (viele Partner und viele Fehlerursachen)
  - Definition von herstellerspezifischen Features als Wettbewerbs-faktor wird erheblich erschwert (Vergleichbarkeit der Produkte wird wesentlich erleichtert)

Gliederung

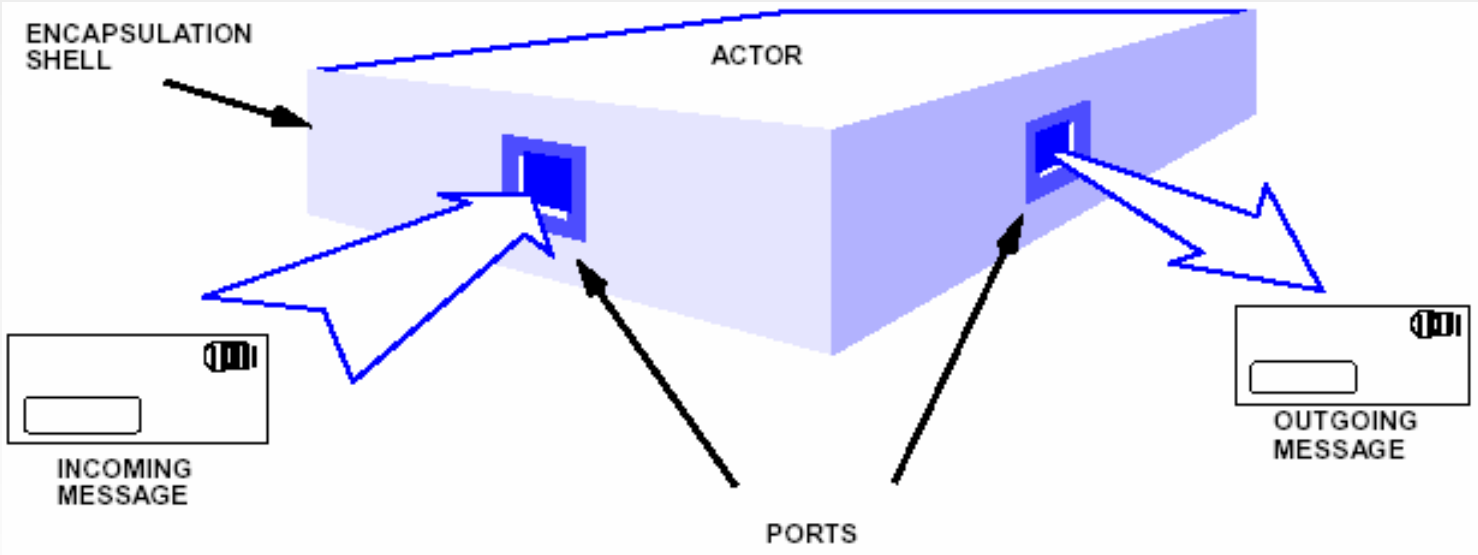
1. Motivation
2. Stand der Technik
3. Lösungsansatz / Validation
4. Zusammenfassung / Ausblick

Stand der Technik

Automatisierungstechnik – IEC 61499



Stand der Technik
Objektorientierung – ROOM

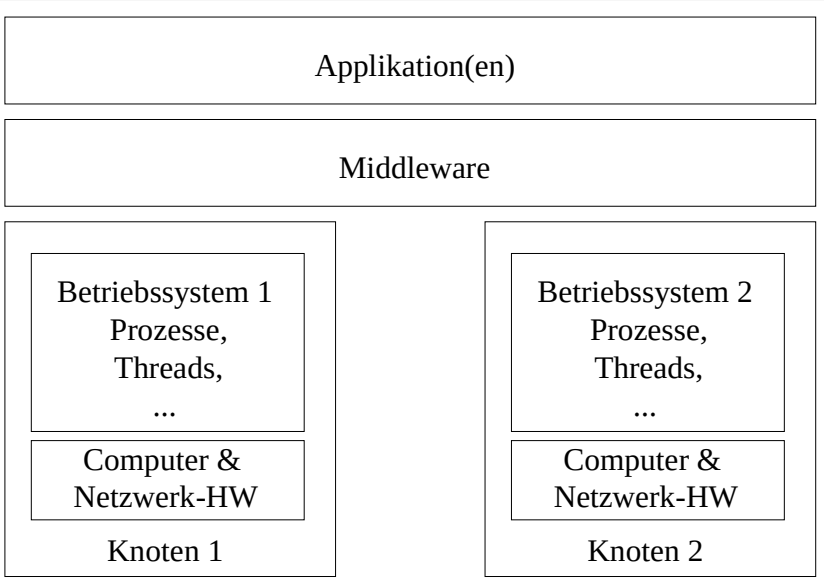


Selic, B.; Gullekson, G.; Ward, P.: “Real-Time Object-Oriented Modeling”, John Wiley & Sons, New York, NY, 1994.

Stand der Technik

Informatik – Verteilte Systeme

- Verteiltes Betriebssystem
  - Einziges Systemabbild
  - Kontrolle über alle Knoten
- Netzwerkbetriebssystem
  - Zugriff auf entfernte Ressourcen
- Middleware basiertes Verteiltes System



Stand der Technik

Informatik – Software-Engineering

- Entwicklungsmodelle / Standards
  - Softwarelebenszyklus: IEC 12207
  - Qualitätsmerkmale: IEC 9126
  - Sicherheit: IEC 61508
  - RUP / oo-Paradigma
- Anforderungen an Design & Codierung
  - Quellcode ist entscheidender Faktor
    - „Quelle des Produkts“
  - Codegeneratoren
  - Programmierrichtlinien

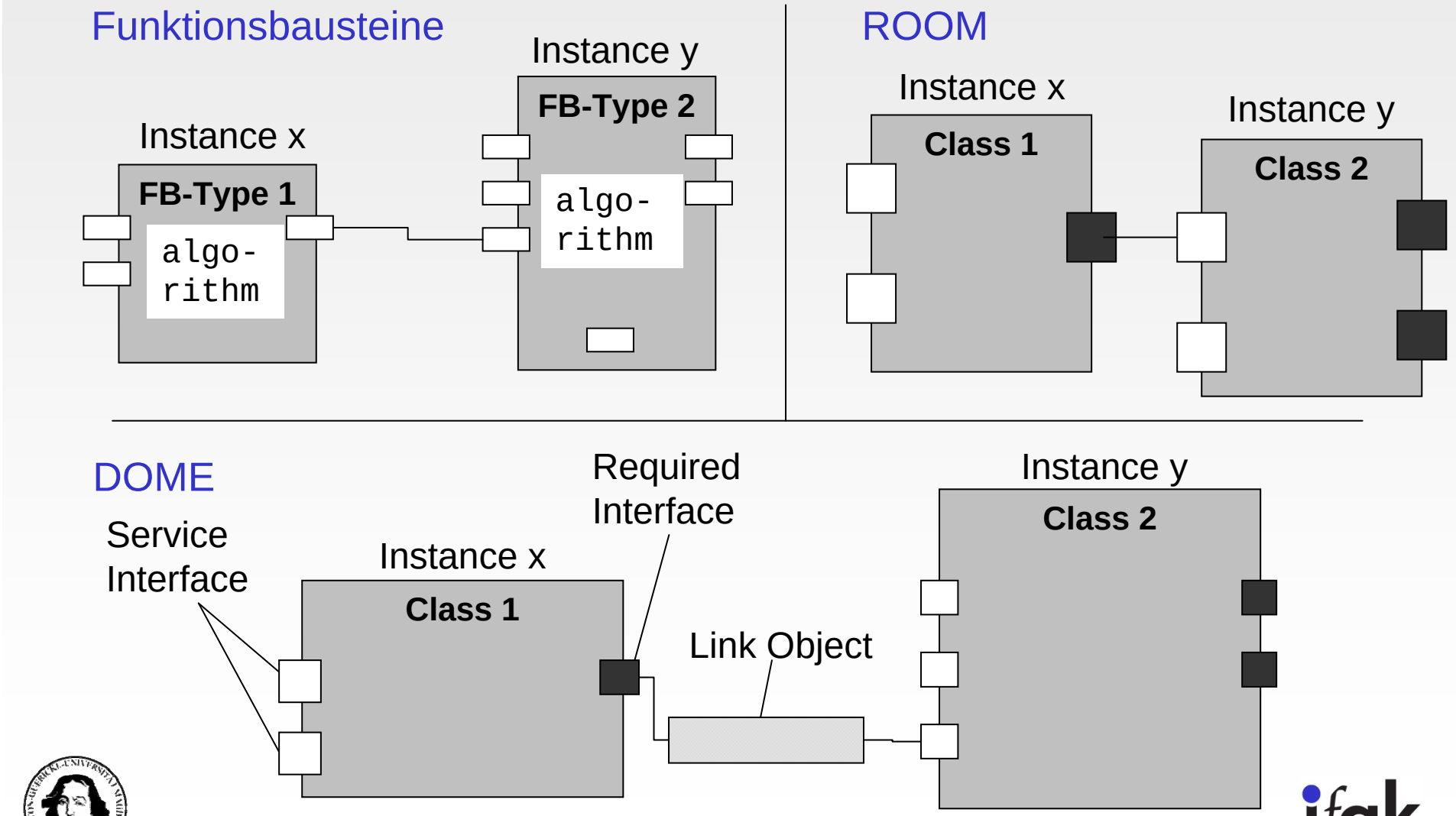
→ metrikbasierte Qualitätssicherung für ausgewählte Paradigmen und Systemarten

Gliederung

1. Motivation
2. Stand der Technik
3. Lösungsansatz / Validation
4. Zusammenfassung / Ausblick

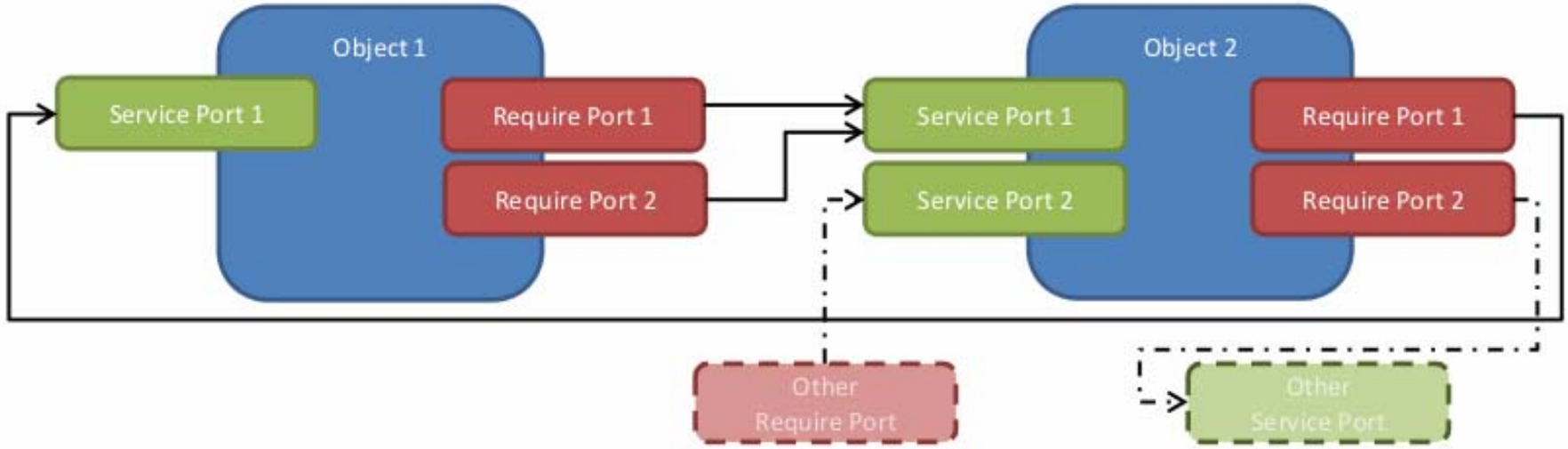
Lösungsansatz / Validation

Modelle im Überblick



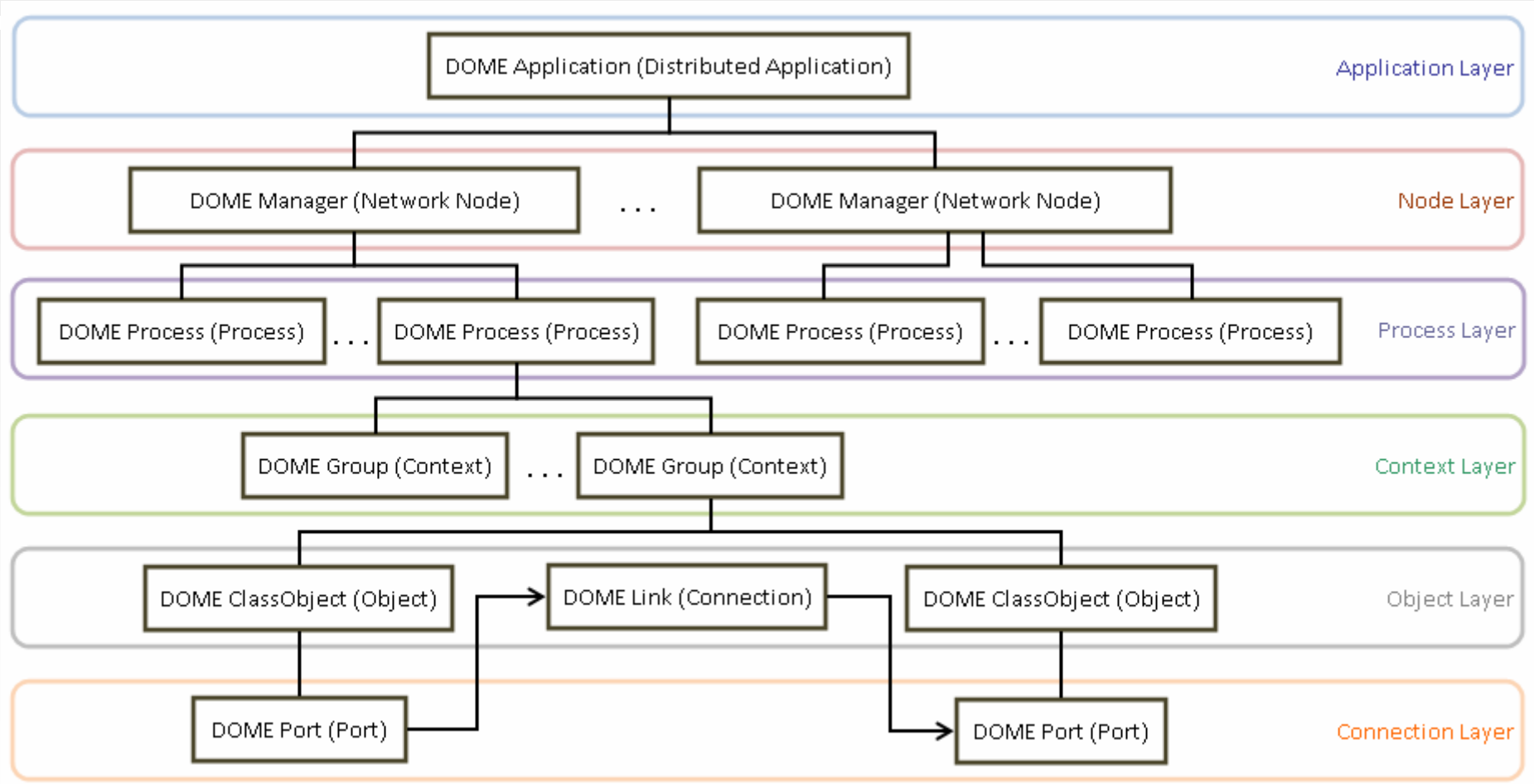
Lösungsansatz / Validation

Objekte und deren Verbindungen in DOME



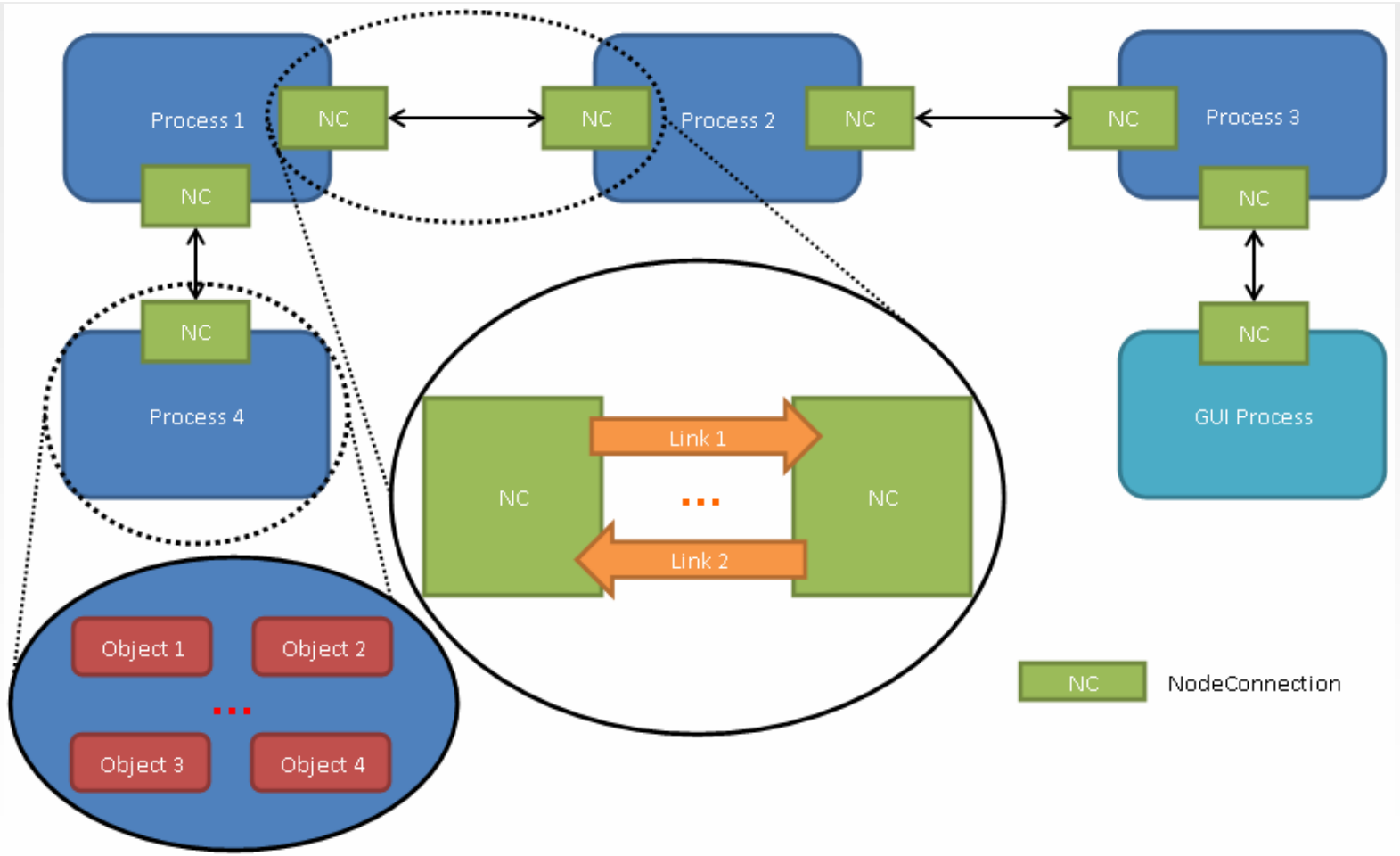
Lösungsansatz / Validation

Verteilte Applikation in DOME



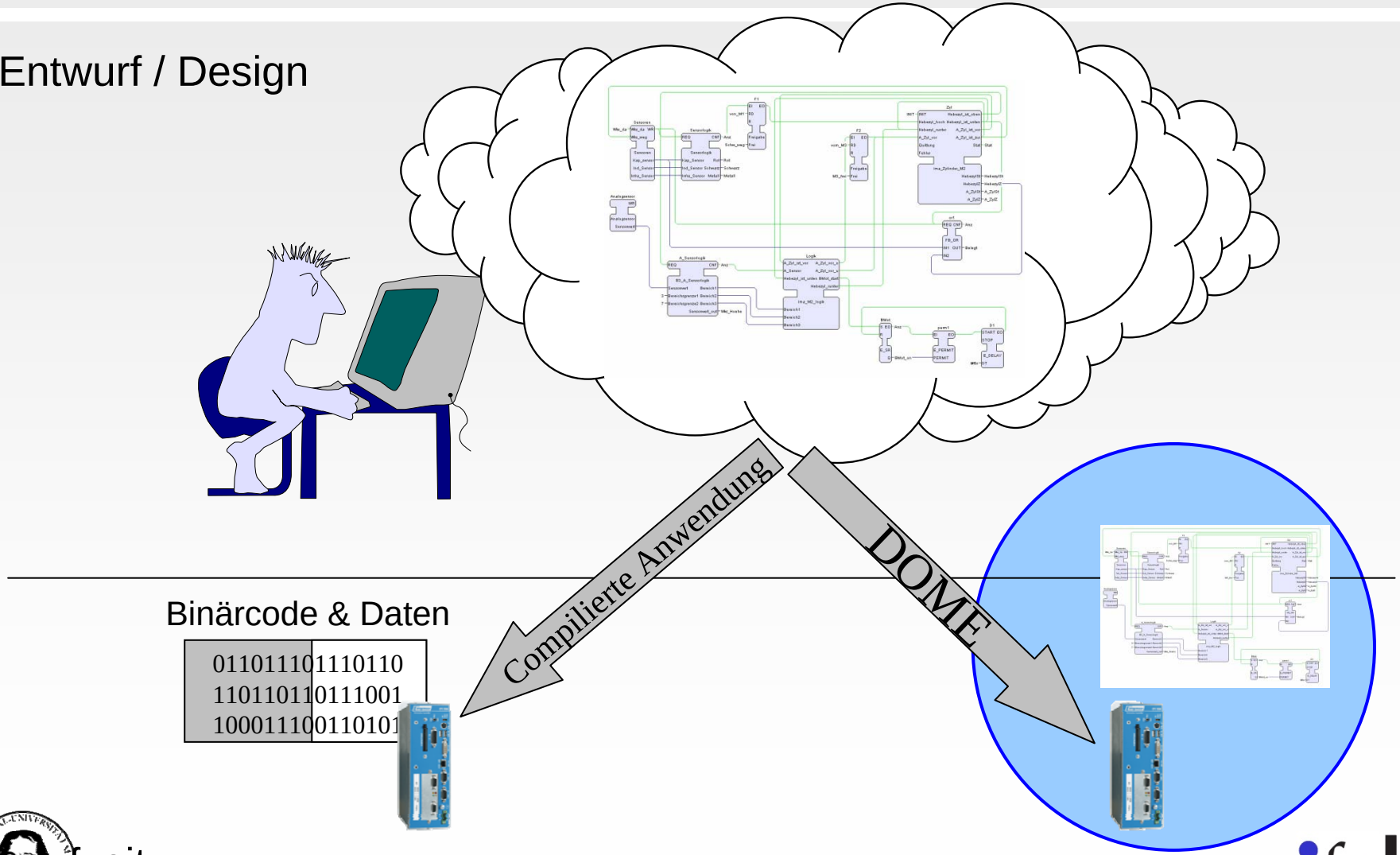
Lösungsansatz / Validation

Verteilte Applikation in DOME



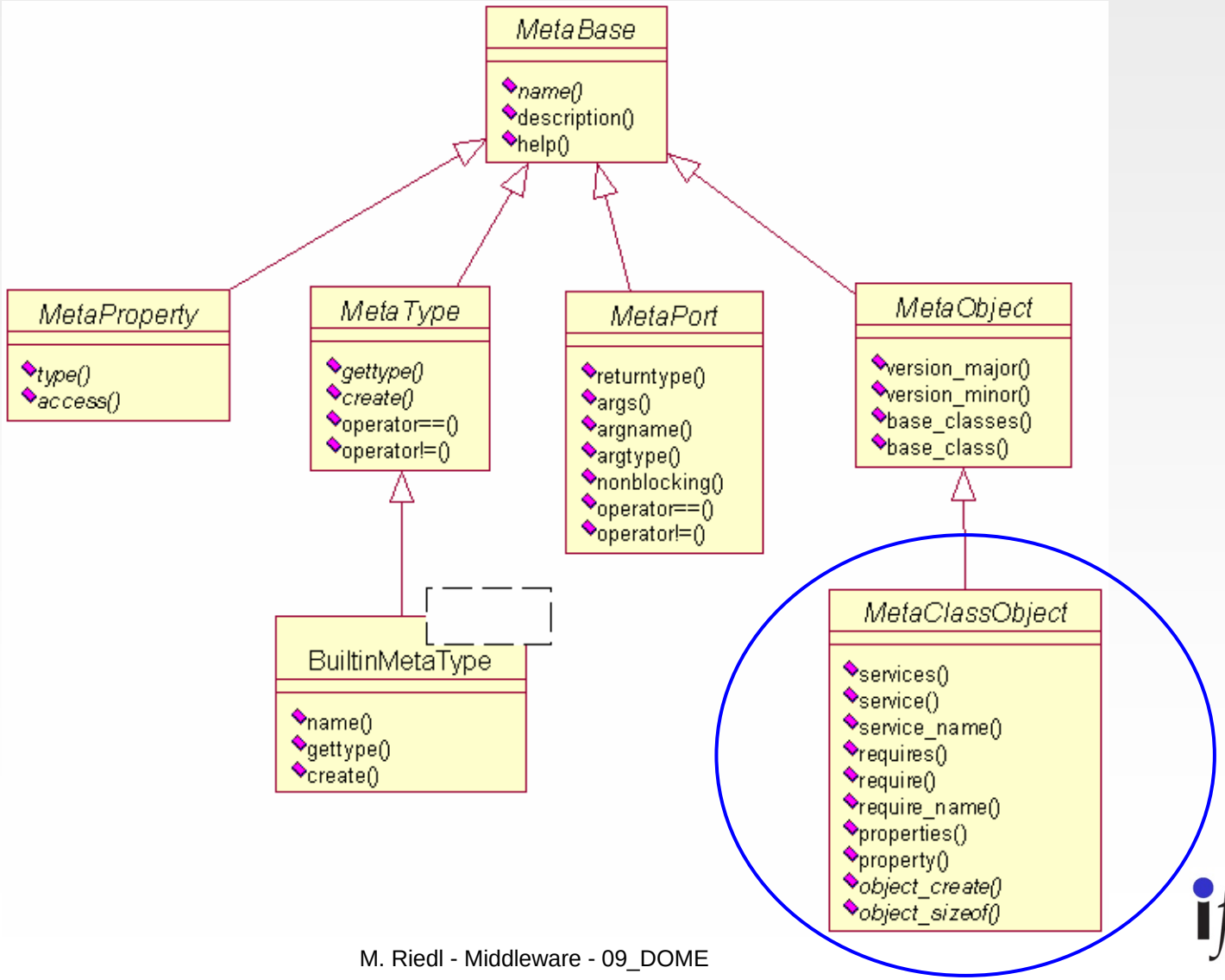
Lösungsansatz / Validation

Objektnetzwerk auch zur Laufzeit verfügbar / Browsen der Applikation zu jedem Zeitpunkt möglich



Lösungsansatz / Validation

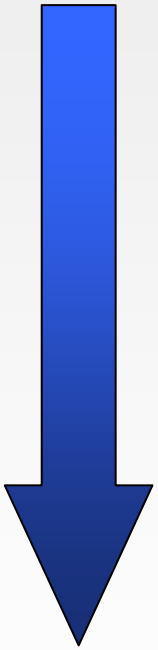
DOME-Reflexionskonzept → Browsen



Lösungsansatz / Validation

Verbindungsart ⇔ Kopplungsgrad

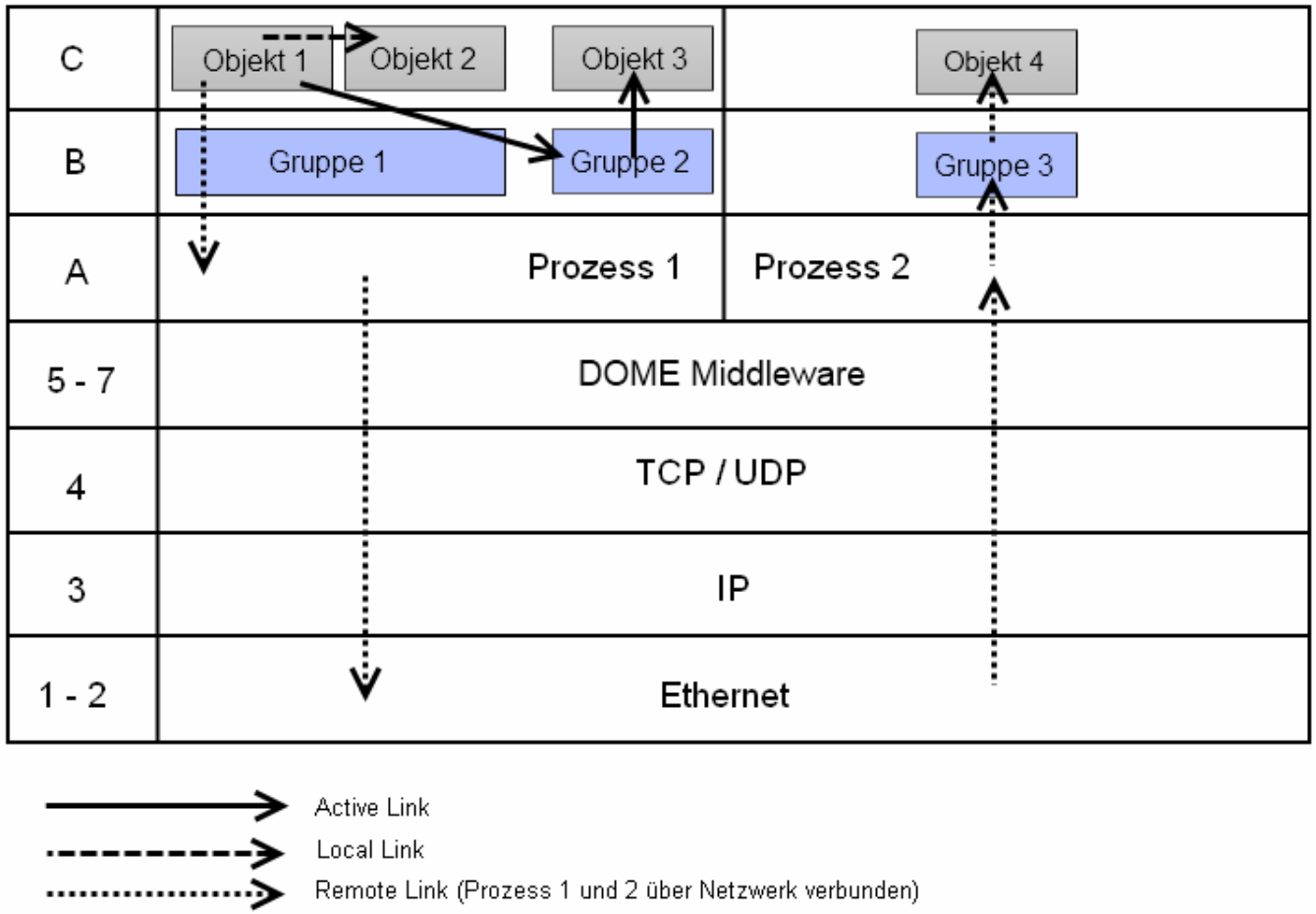
- Objektverbindungsarten
  - Lokal →unterbrechungsfrei
  - Prozess-lokal
    - Synchron
    - Asynchron
  - Interprozess
    - Synchron
    - Asynchron



eng

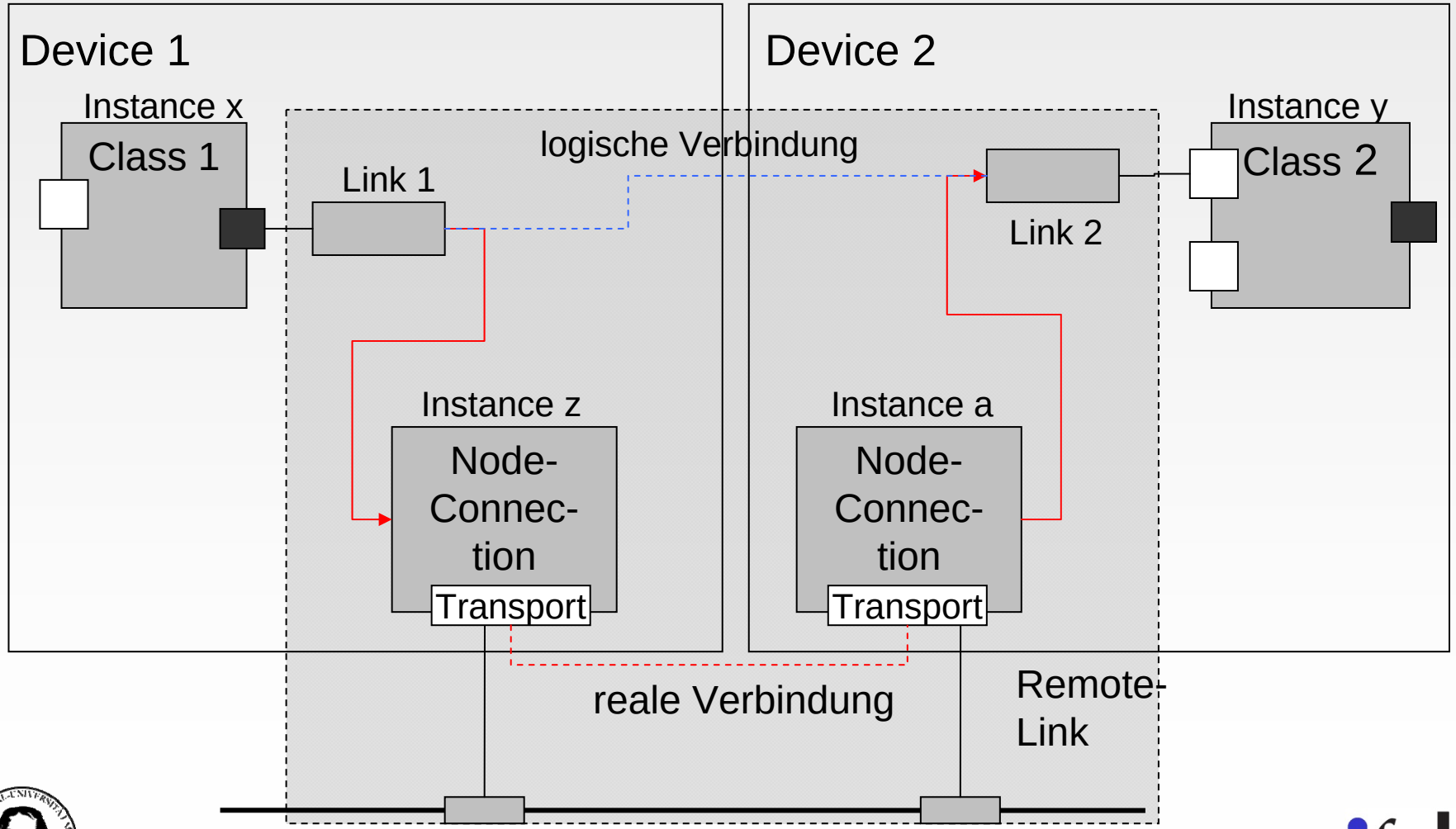
dynamisch,
lose

Lösungsansatz / Validation
Verbindungsart ⇔ Kopplungsgrad



Lösungsansatz

Entfernter Nachrichtenaustausch



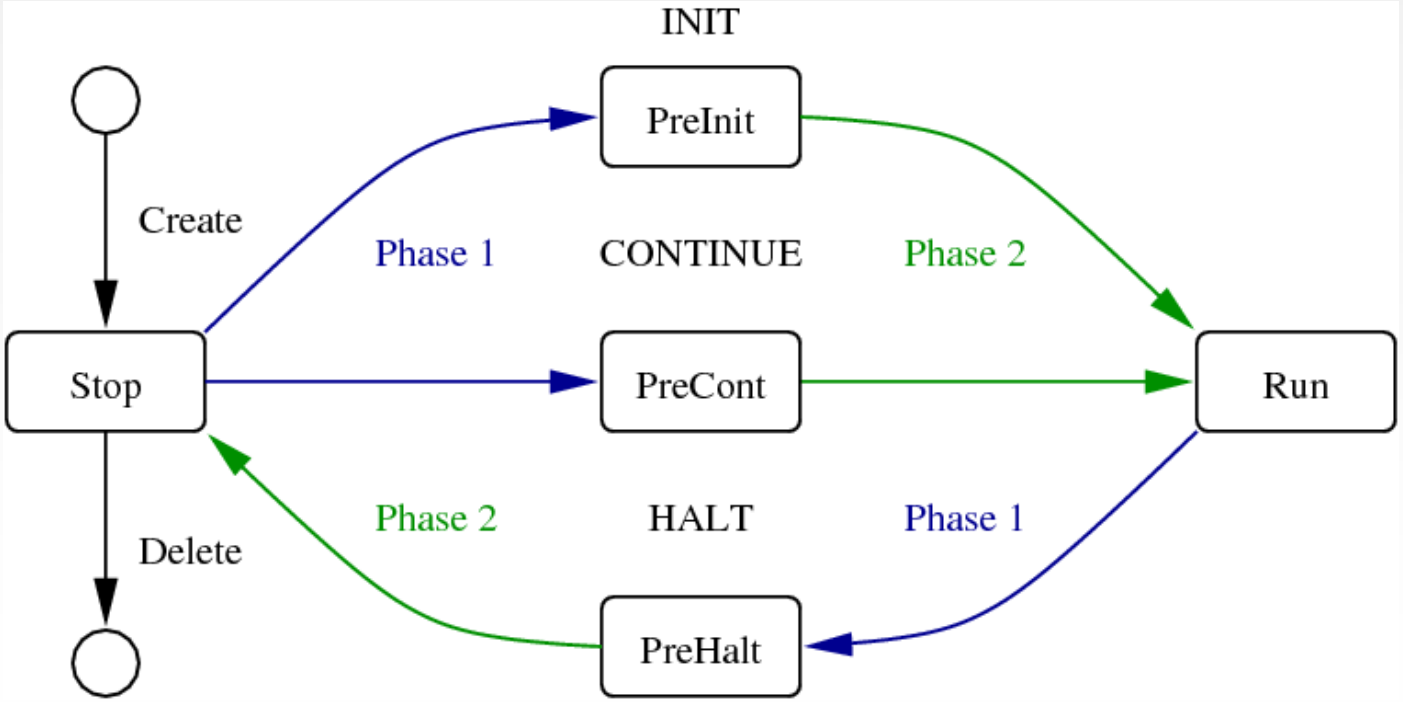
Lösungsansatz / Validation

Flexible Links

- Basis für rekonfigurierbare Systeme
  - Skalierbarkeit
  - Ausfallerkennung
- Erforderliche Funktionalitäten
  - Automatische Wiederverbindung
  - Verteilter Anlauf

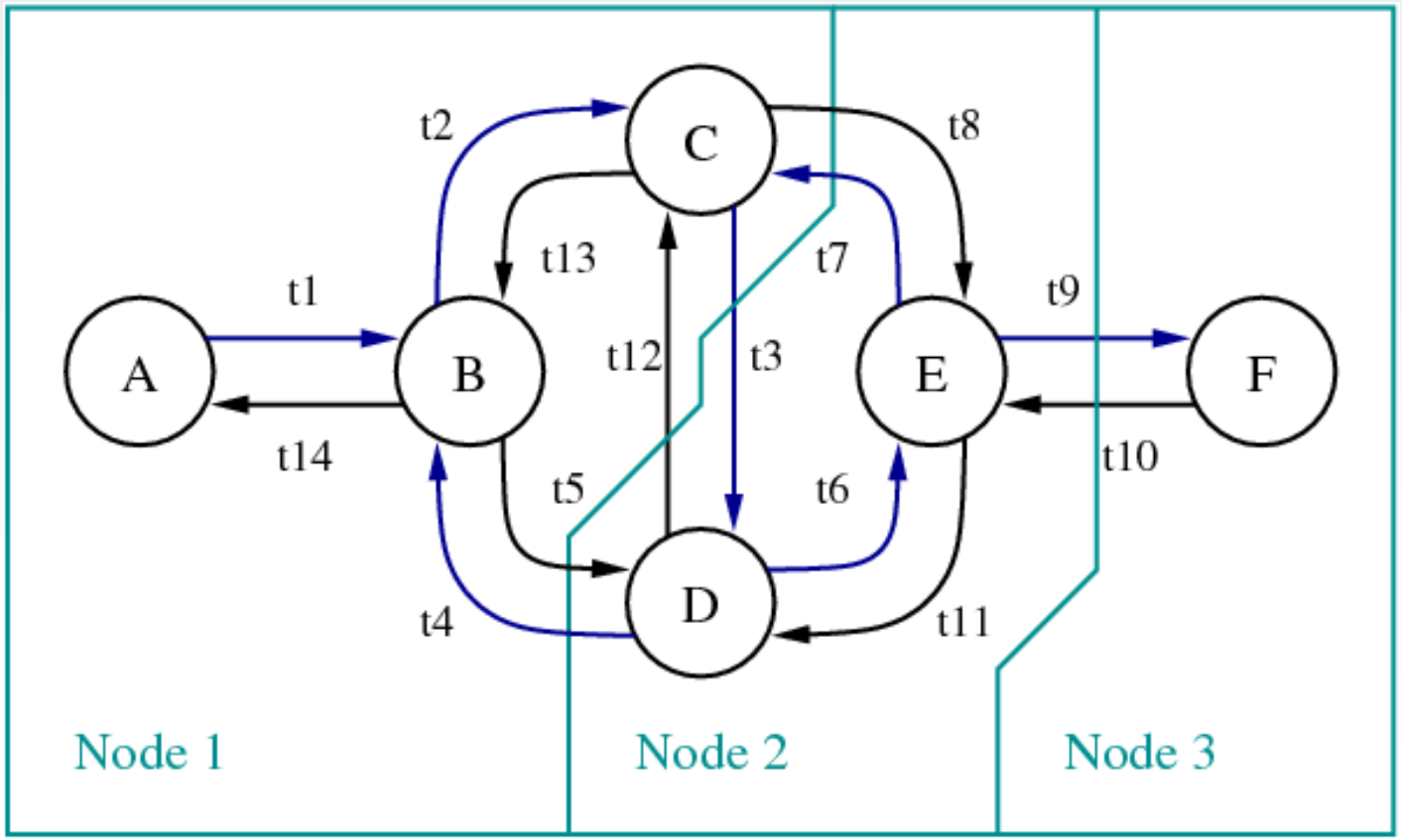
Lösungsansatz / Validation
Verteilter Start / Stop

- Zwei-Phasen-Commit-Protokoll
  - Vorbereitung des neuen Zustands (z.B. Sperren der Ressourcen)
  - Aktion



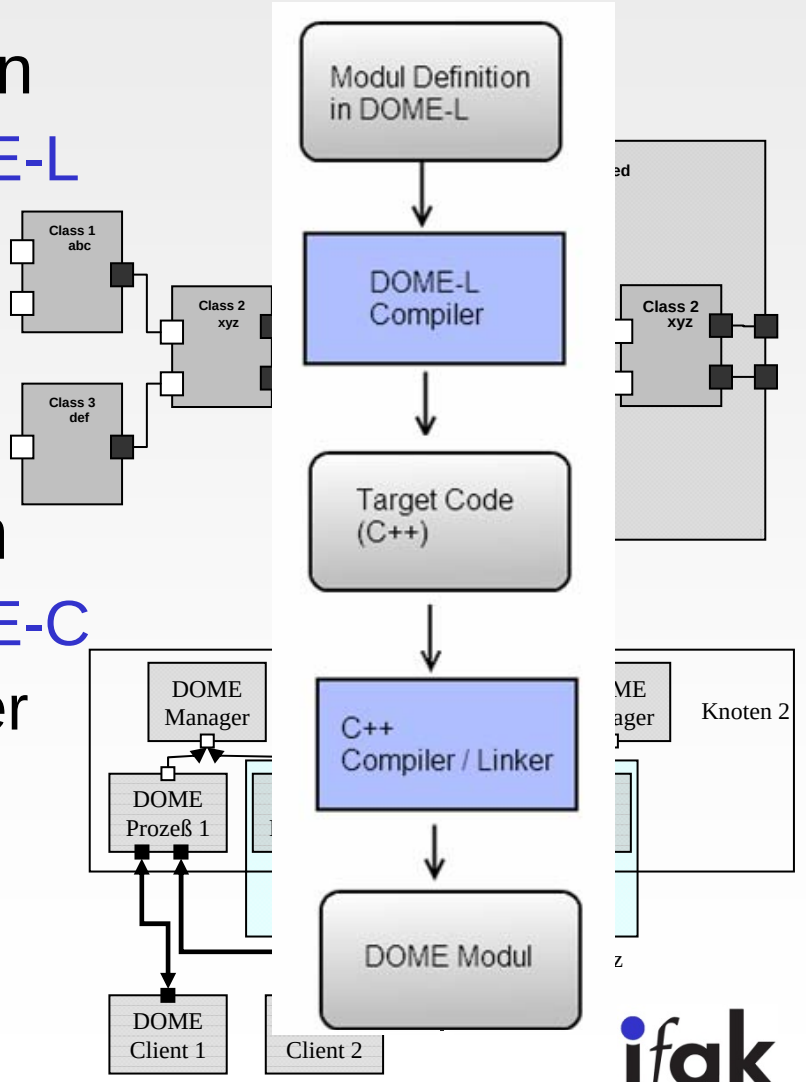
Lösungsansatz / Validation
Verteilter Start / Stop

- Start im Prozeß ‚A‘ (logischer Ur-Master)

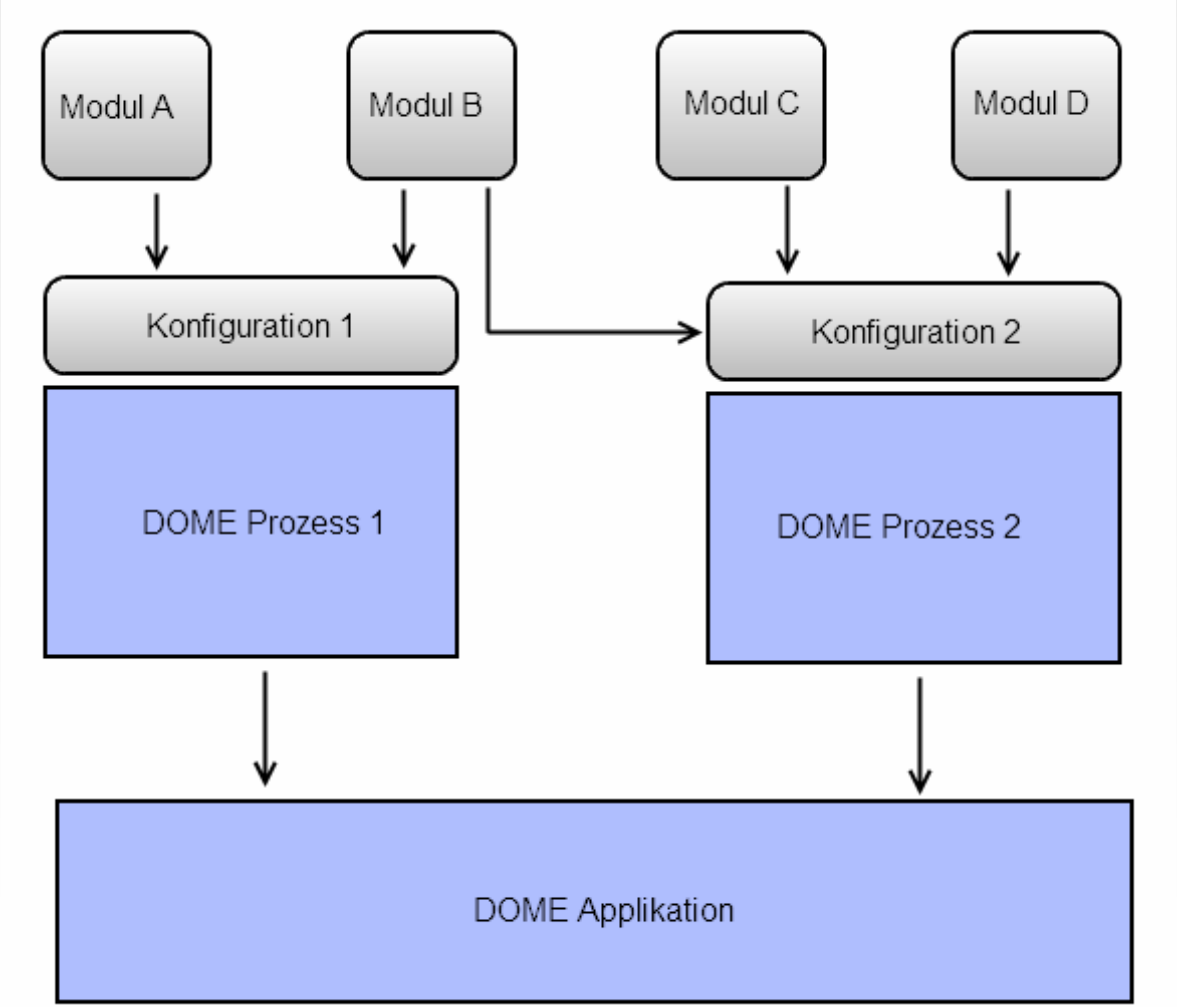


Lösungsansatz / Validation
Engineering

- Erstellung von DOME-Objekten
  - Spezifische Sprache → DOME-L
  - Werkzeugkette
- Komposition von Objekten
- Bibliothekskonzept
- Erstellung von Applikationen
  - Spezifische Sprache → DOME-C
  - Auswertung im ObjectManager
- DOME-Manager



Lösungsansatz / Validation
Topologie

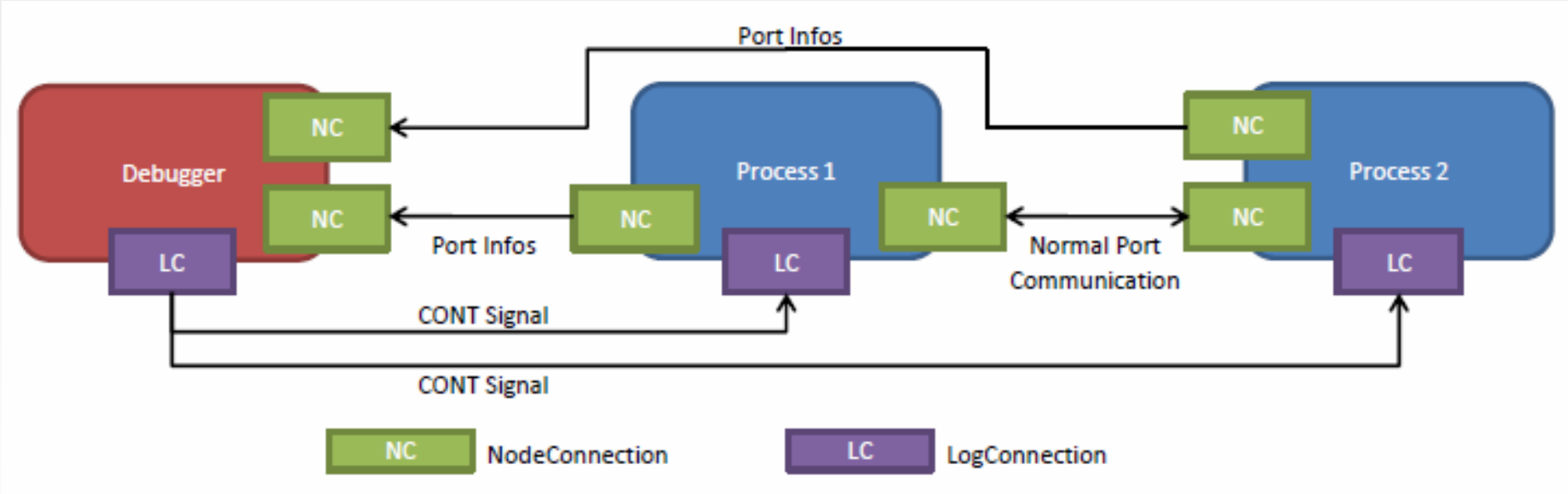


Lösungsansatz / Validation
Engineering

- Graphische Editoren
  - Applikationsdesign
    - IEC 61499 – FBDK, XML-Converter ist ‚fast' vollständig
  - Instrumentierung
- Inbetriebnahme
  - Browsen der Applikation
  - Tracing
- Bedienen & Beobachten
  - OPC - Anschluss

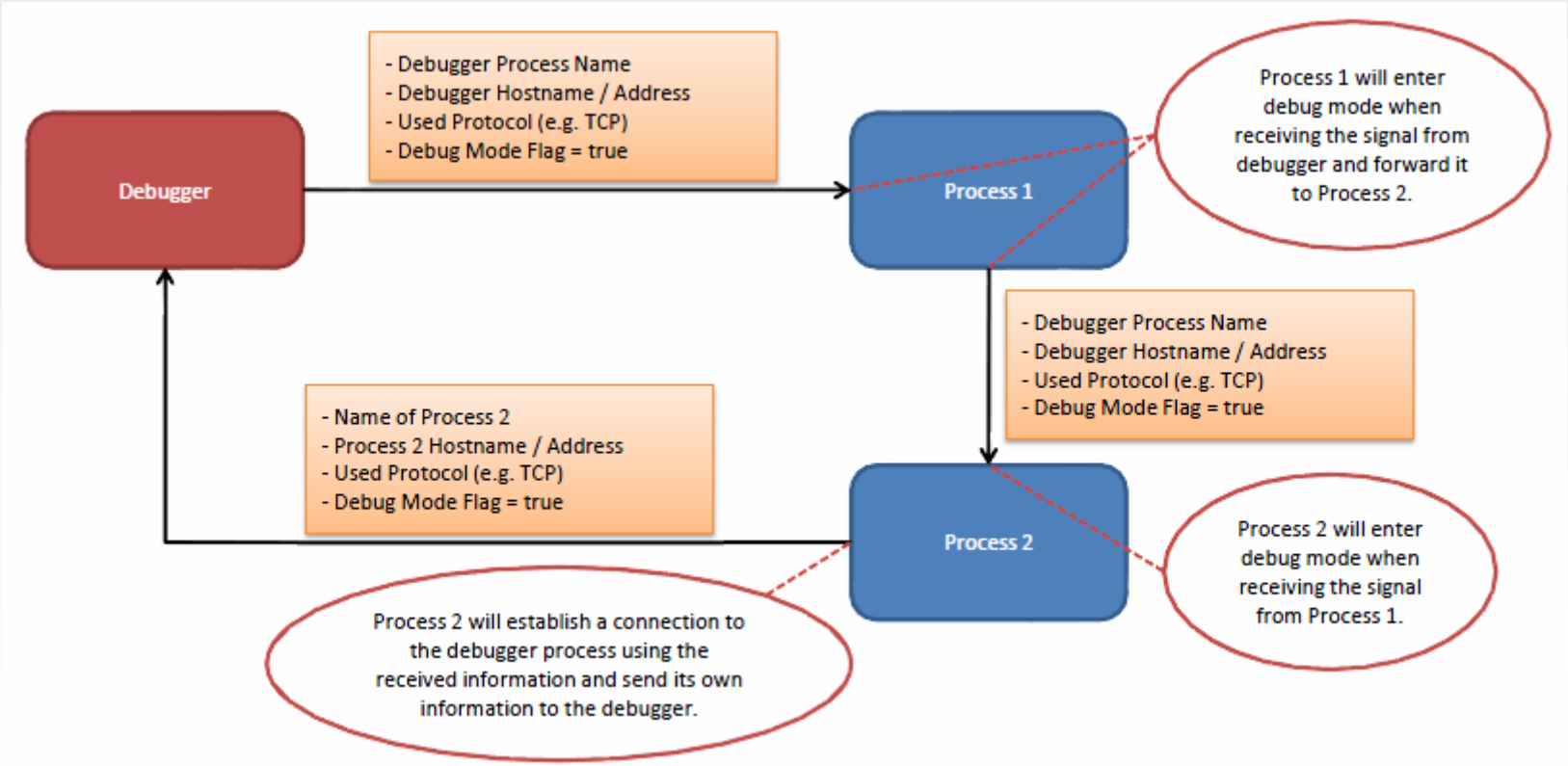
Lösungsansatz / Validation

Inbetriebnahme - Debugger



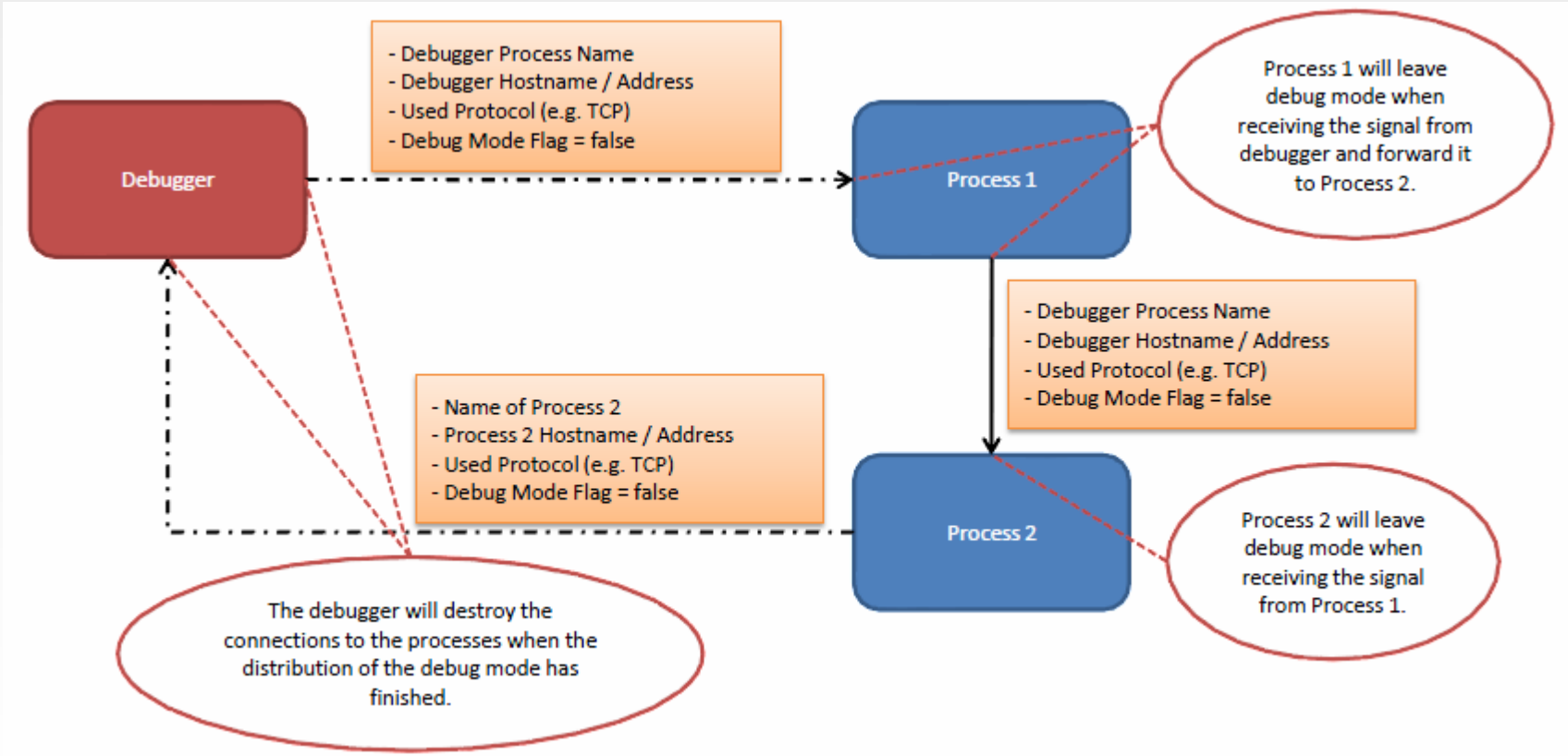
Lösungsansatz / Validation

Inbetriebnahme – Debugger - Start



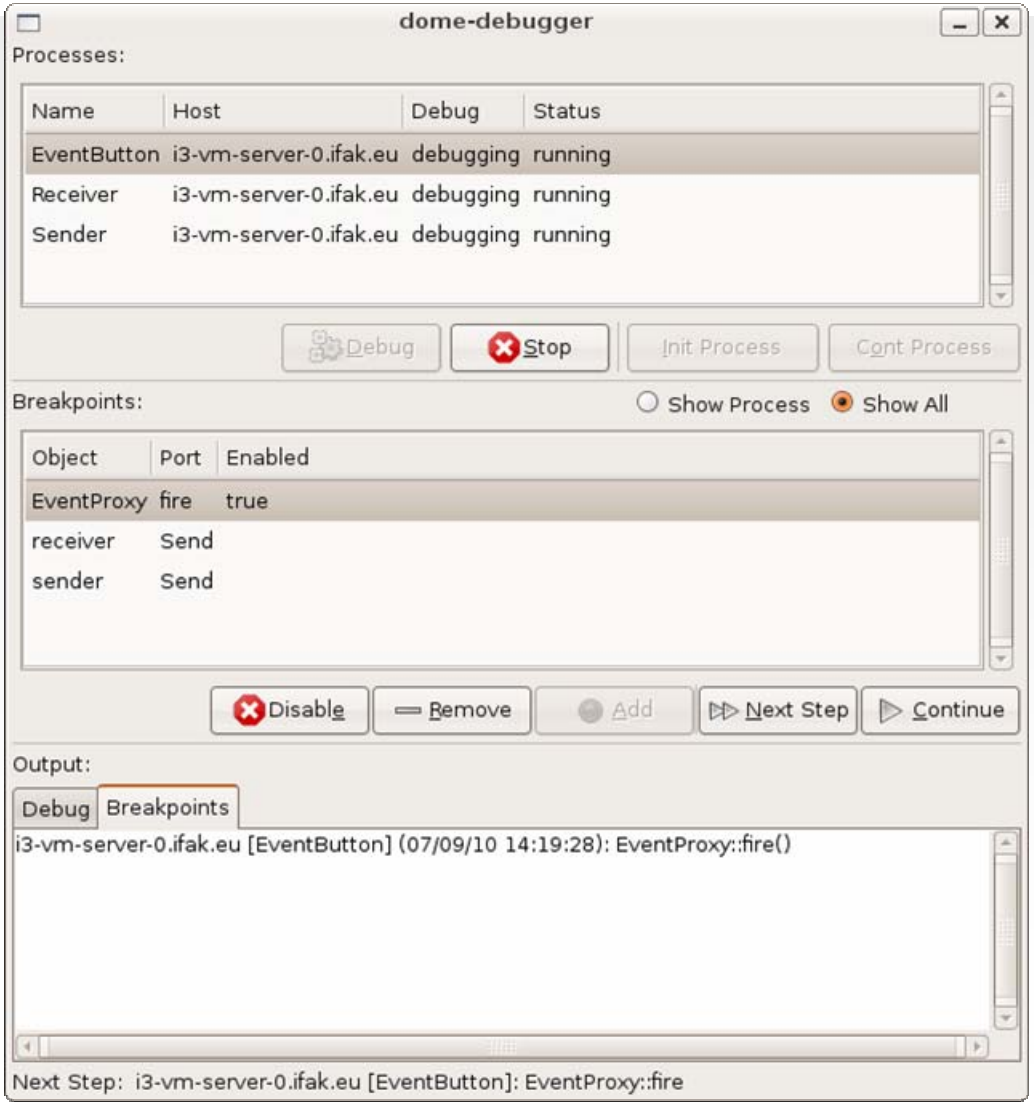
Lösungsansatz / Validation

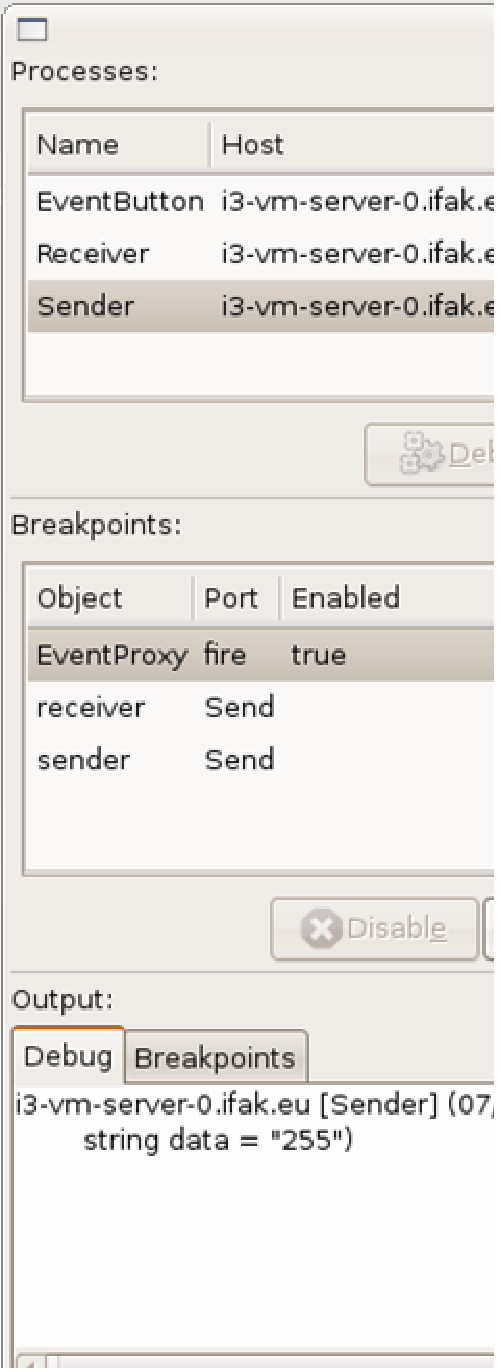
Inbetriebnahme – Debugger - Stop



Lösungsansatz / Validation

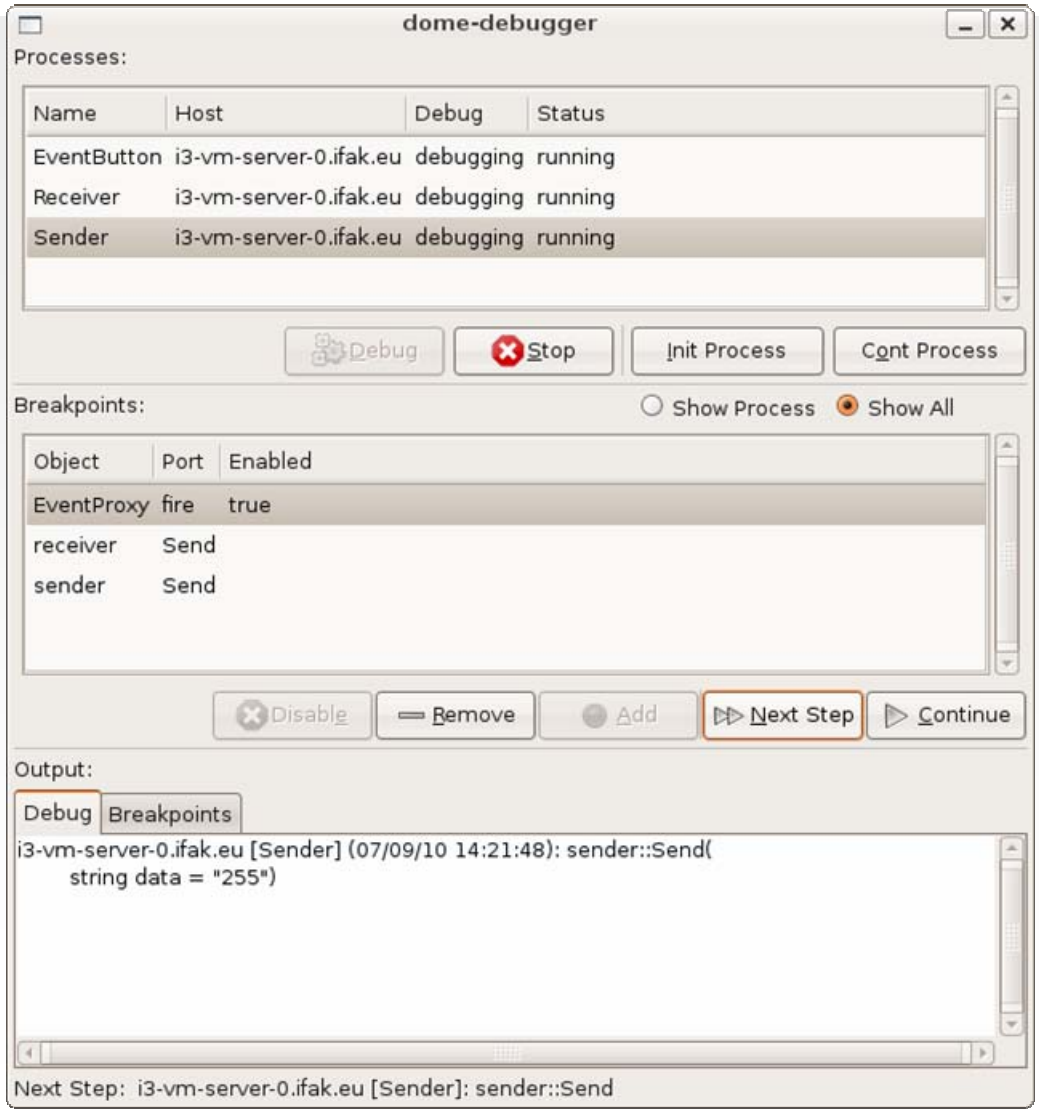
Inbetriebnahme – Debugger - Breakpoints





Lösungsansatz / Validation

Inbetriebnahme – Debugger – Stopp der verteilten Applikation



Lösungsansatz / Validation

Bewertungsmöglichkeiten

- Qualitative Bewertung
  - Entwicklungsressource → DOME
  - Produkte / Systeme → DOME-L
  - Werkzeug
    - Telelogic Tau® Logiscope™
    - Schwellwerte: Bewertung über ‚alle' Systeme
    - Bewertungen ordinal (EXCELLENT, GOOD, ...)
- Performance

Lösungsansatz / Validation

Qualitative Bewertung der DOME-Laufzeit

- Applikationsebene
  - (MOOD, Summe / Durchschnitt der CC, Durchschnittliche Kopplung)

Kriterium	Bewertung
Analysierbarkeit	FAIR
Anpaßbarkeit	GOOD
Stabilität	FAIR
Testbarkeit	GOOD

Lösungsansatz / Validation

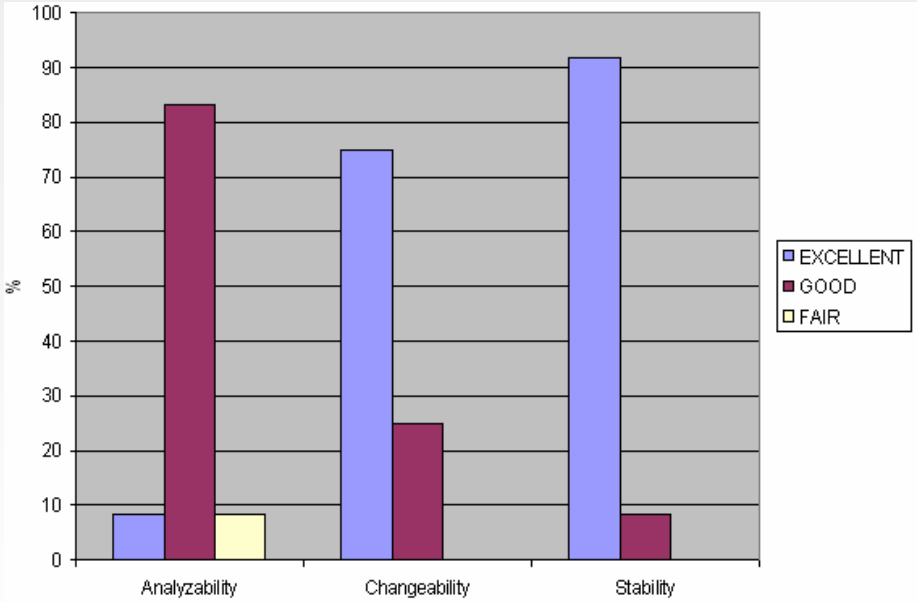
Qualitative Bewertung unterschiedlicher Systeme

- Beispiel: Bereichsüberwachung
  - 3 Objekte
  - Unterschiedliche Modelle
    - IEC 61131
    - IEC 61499
    - DOME pur (Client-Server-Relation)
  - Enge Kopplung der Objekte
  - Abgesetzte Visualisierung

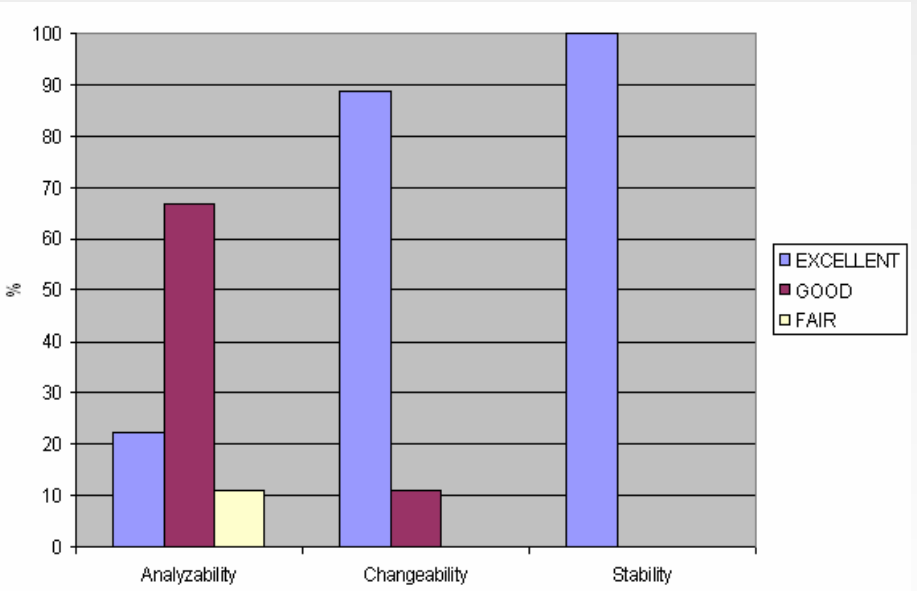
Lösungsansatz / Validation

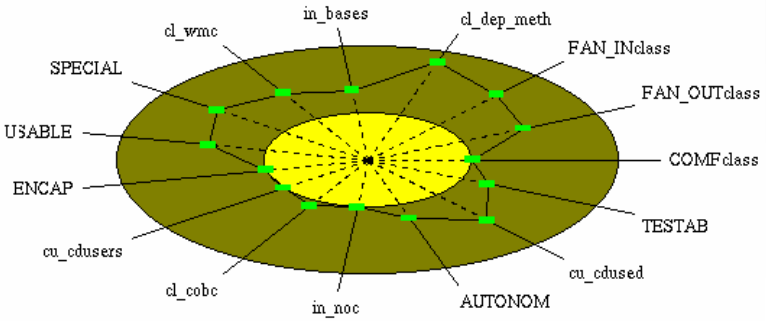
Vergleich der Produkte hinsichtlich Qualität

IEC 61499



DOME pur





Lösungsansatz / Validation

Performance - Laufzeitmessungen

- Vergleich
  - Eingebettete C++ Objekte
  - Virtuelle Interfaces
  - DOME-Ports
- 10.000.000 Dienstaufrufe
- Betriebssysteme:
  - NetBSD (32- und 64Bit)
  - Linux (32- und 64Bit)
  - MS-Windows XP, SP2 (32- und 64Bit)
  - µLinux
- Verschiedene Board-Ausstattungen

$$\rightarrow f_{Ports} = \frac{t_{DOME}}{t_{C++}} = 9{,}7 - 16$$

Gliederung

1. Motivation
2. Stand der Technik
3. Lösungsansatz / Validation
4. Zusammenfassung / Ausblick

Zusammenfassung
Bewertung von DOME

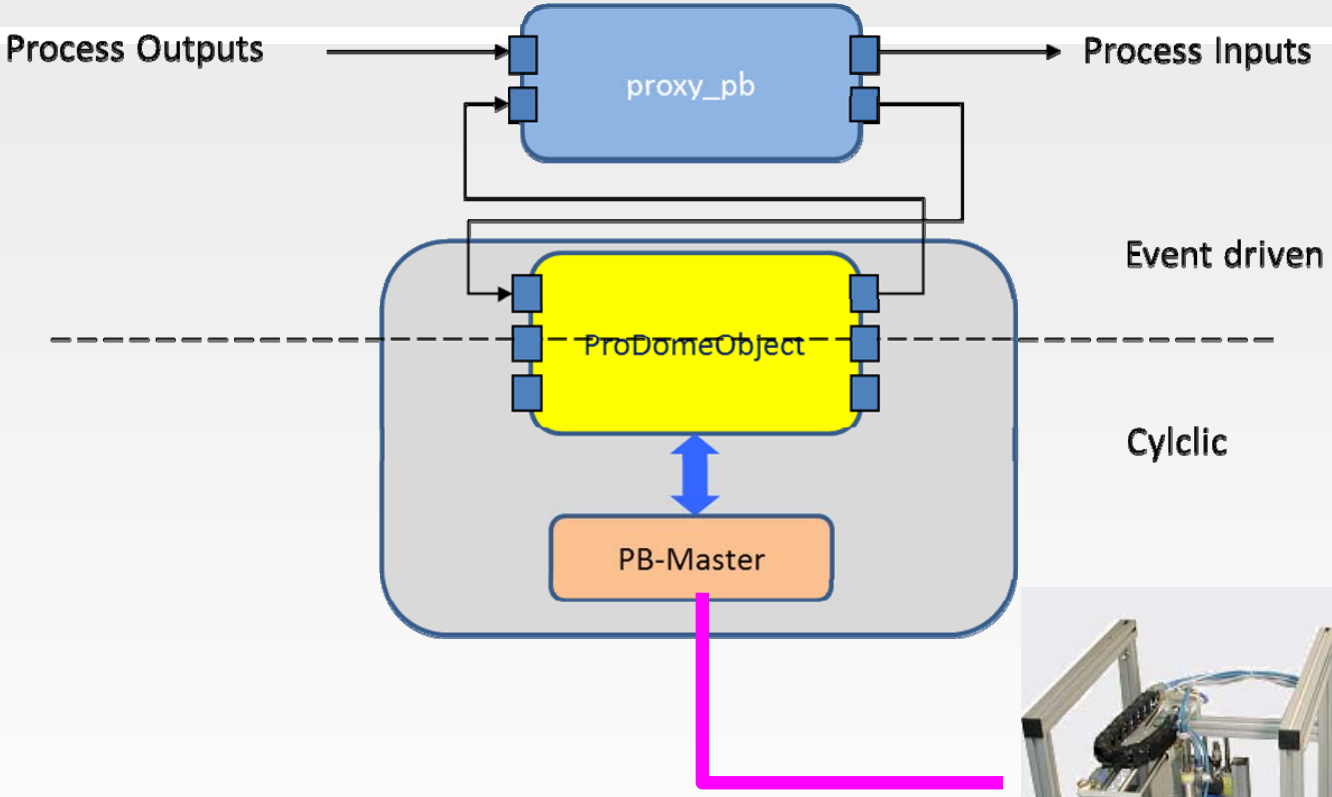
- Vorteile von DOME
  - Einfachheit
  - Flexibilität durch späte Bindung → Voraussetzung für Plug & Play
  - Typsicherer Daten- und Informationsfluss
  - Portabilität
  - Verschiedene Kopplungsgrade
  - Introspektion → Browsen
- Nachteil von DOME
  - Laufzeit bei entfernter Objektkommunikation

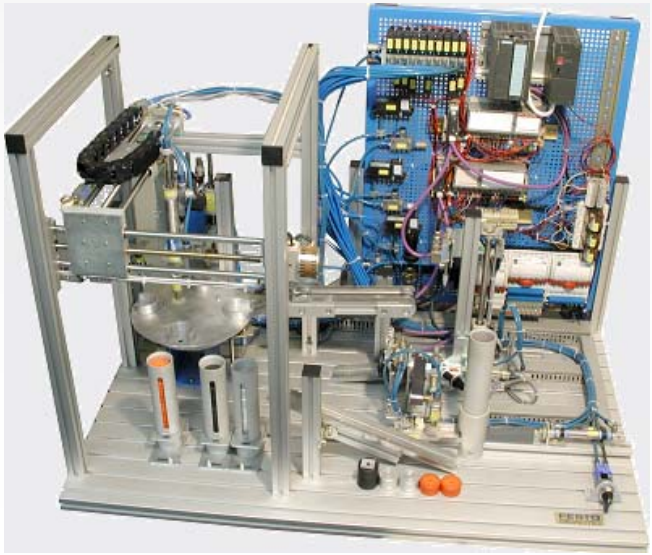
ifak

Zusammenfassung

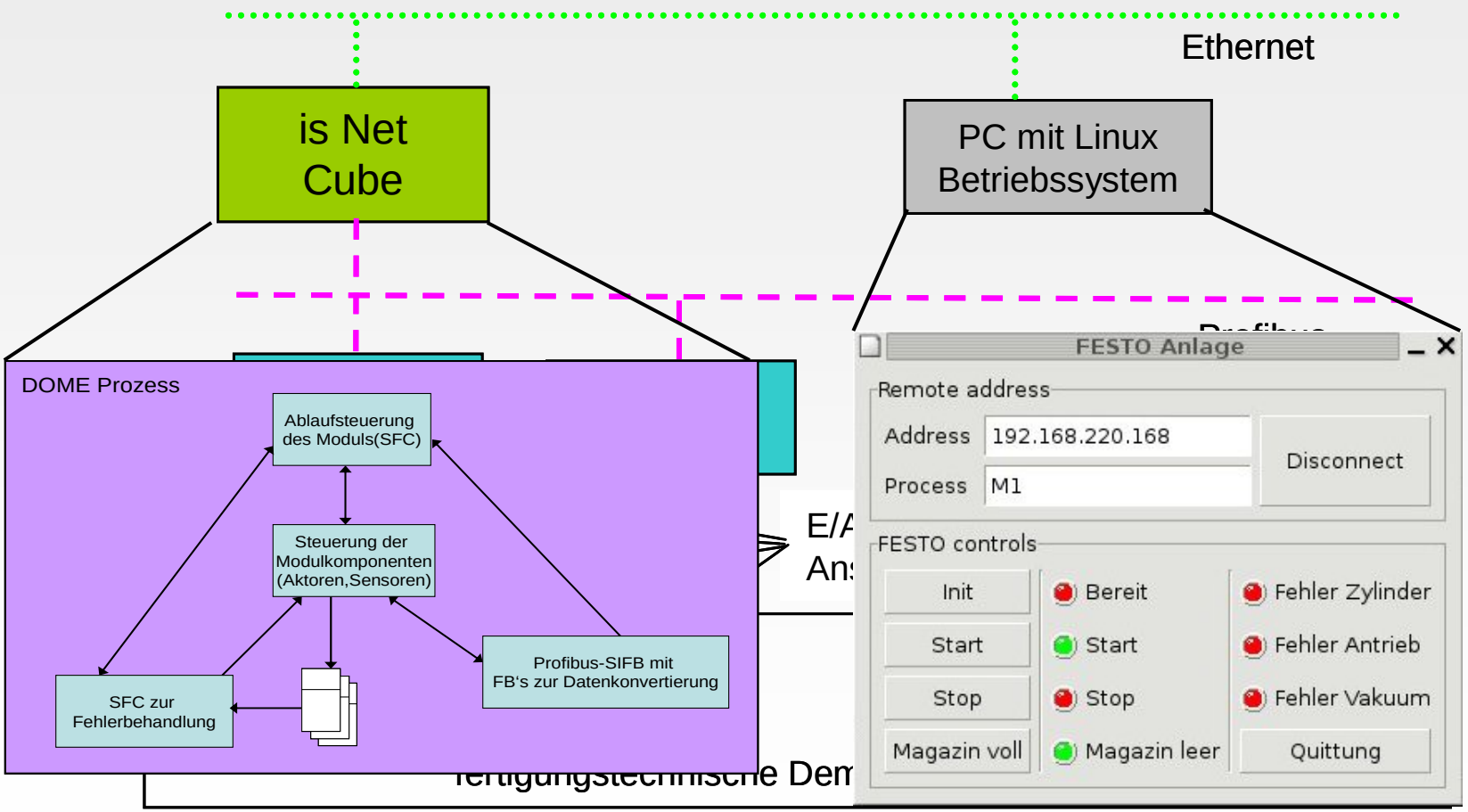
- Vorteile von DOME überwiegen
- Qualitative Bewertungsmöglichkeit der Objekte (offline)
- Anwendbarkeit des Funktionsblock-Ansatzes, d.h. von Modularisierung
- Typsicherer Daten- und Informationsfluss
- Spezifische Sprachen DOME-L, DOME-C
- Skalierbarkeit des Gesamtsystems
- Einsatz in ifak Demonstrationsanlage
  - Positive Einschätzung der Automatisierungstechniker

Nutzung von PROFIBUS

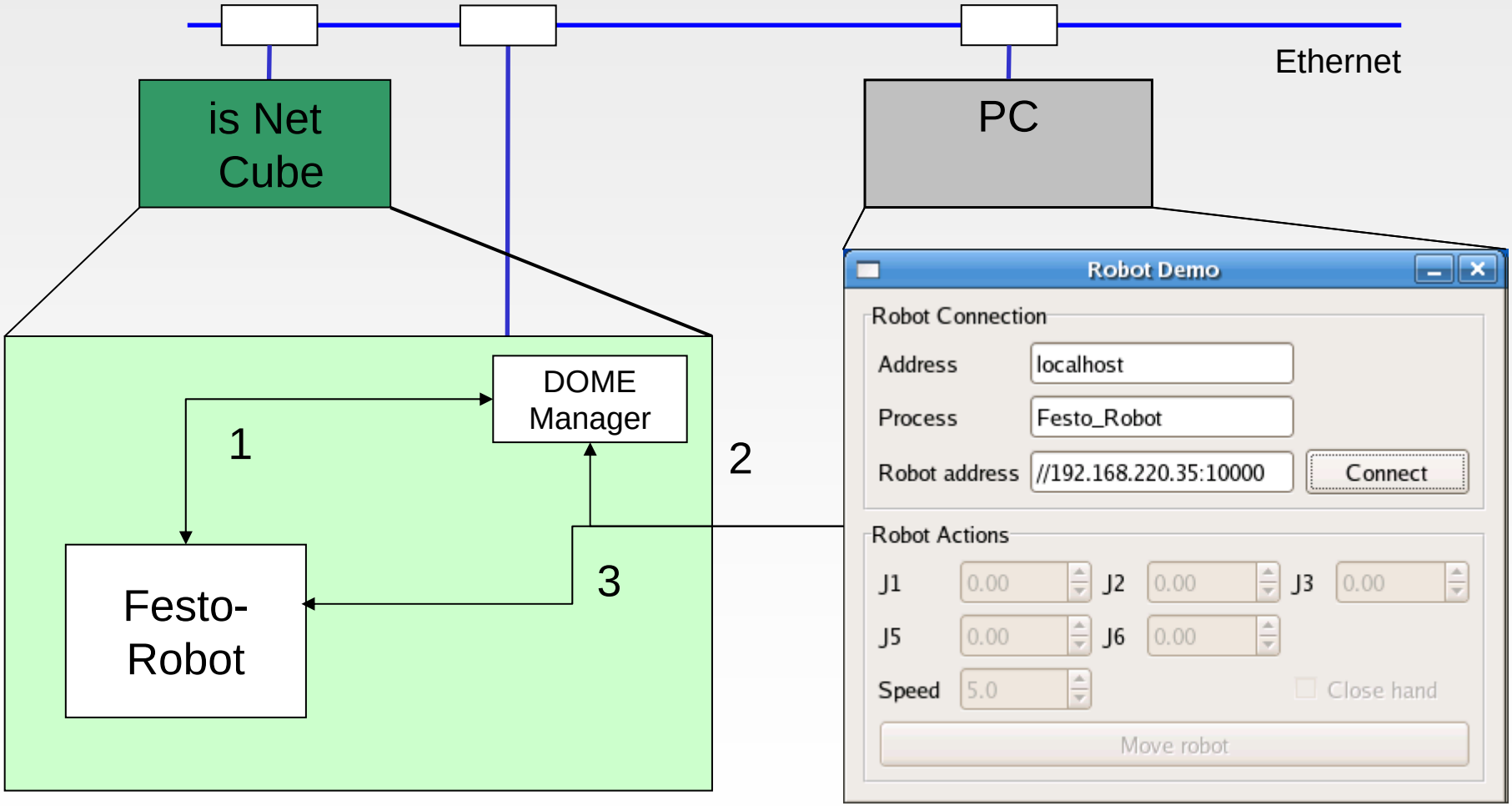




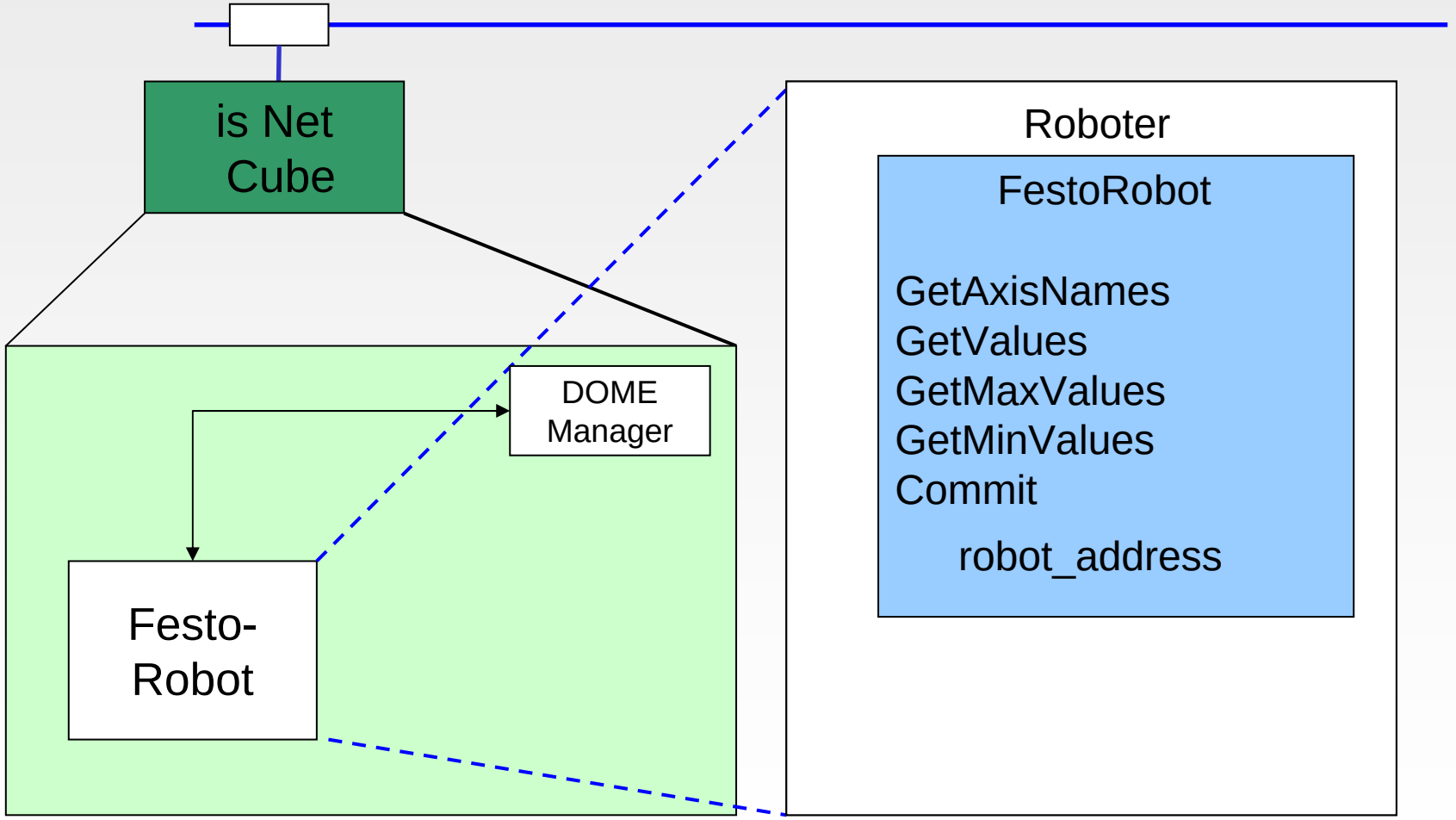
Nutzung in Festo-Anlage



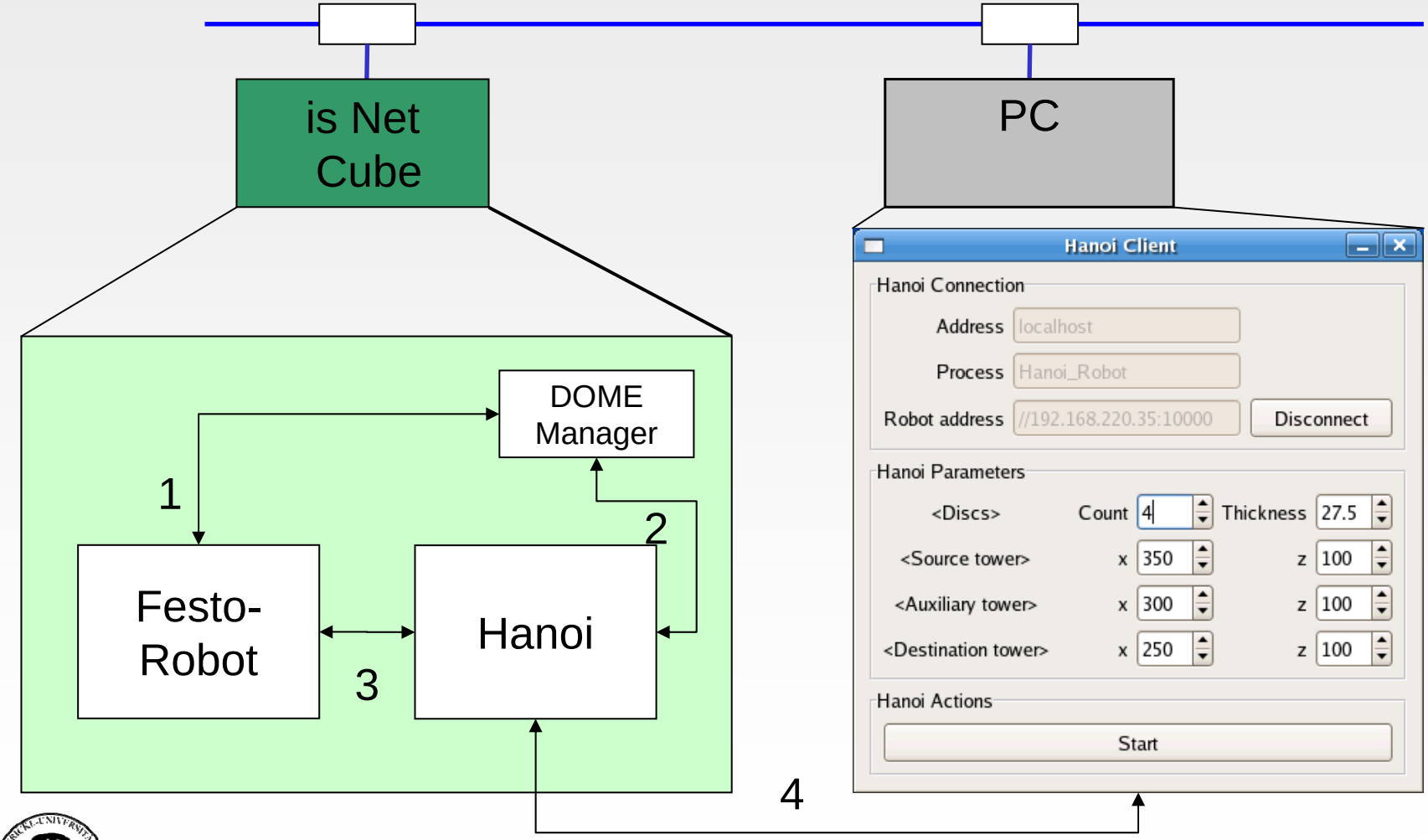
Zugriff Roboter



„Festo-Robot“ Prozeß-Interna



Applikation mit „Festo-Robot“



Vertikale Integration