

Distributed Object Model Environment

Ein objektorientiertes Softwaremodell
für verteilte Automatisierungssysteme

Matthias Riedl

ifak e.V. Magdeburg

Werner-Heisenberg-Str. 1, 39106 Magdeburg

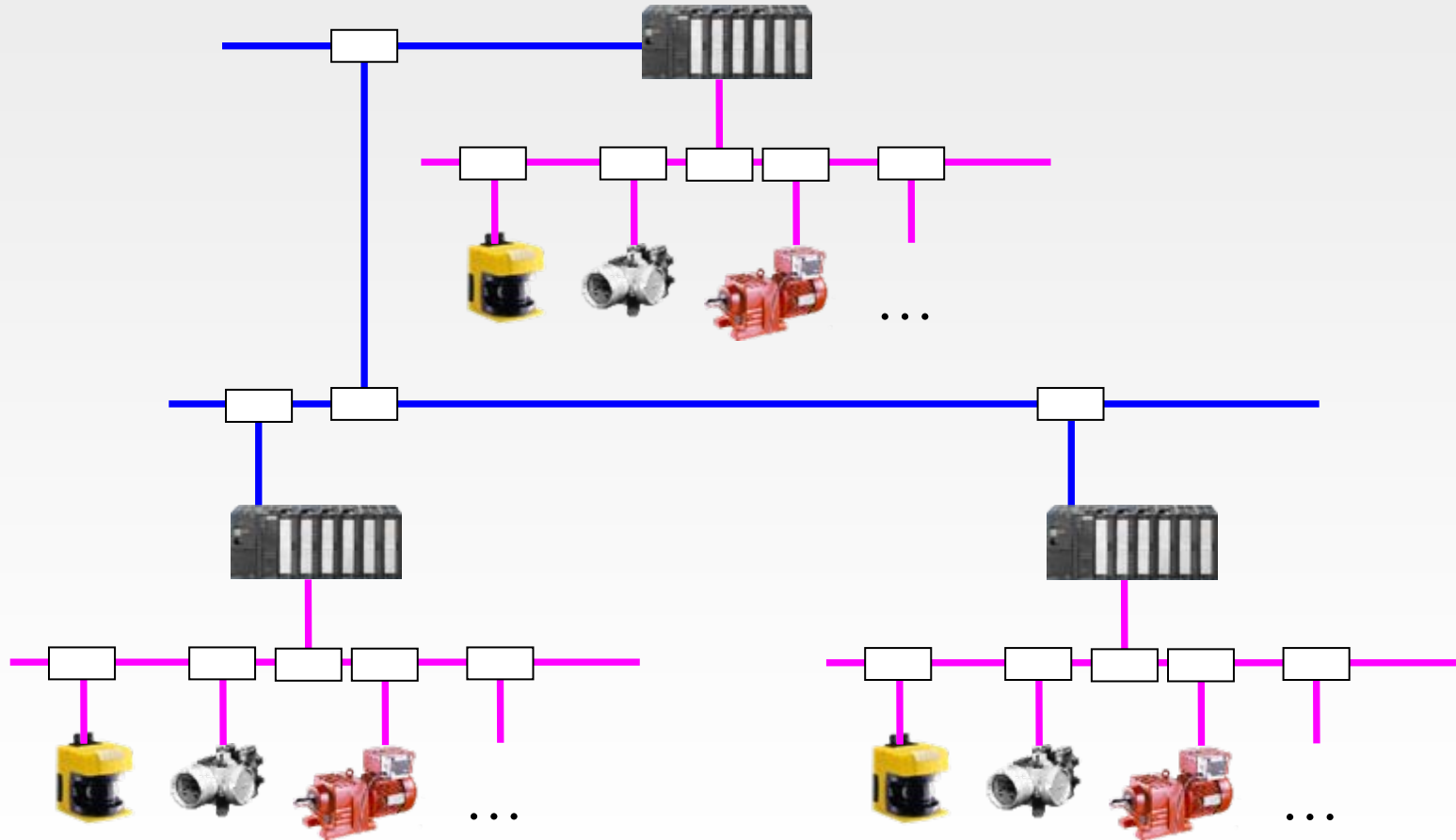
www.ifak.eu

matthias.riedl@ifak.eu

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

Motivation

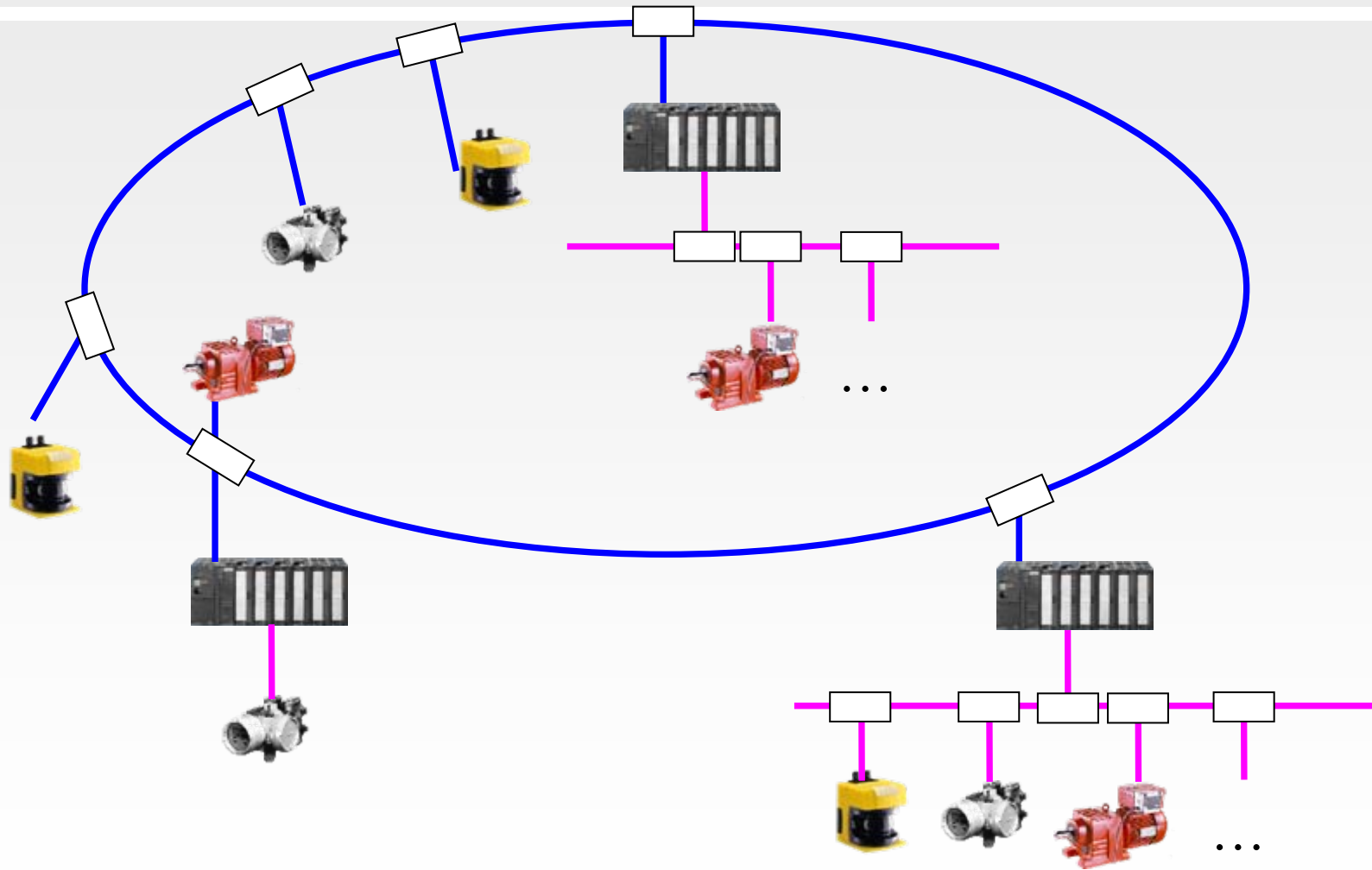
Ausgangszustand



- ❑ 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

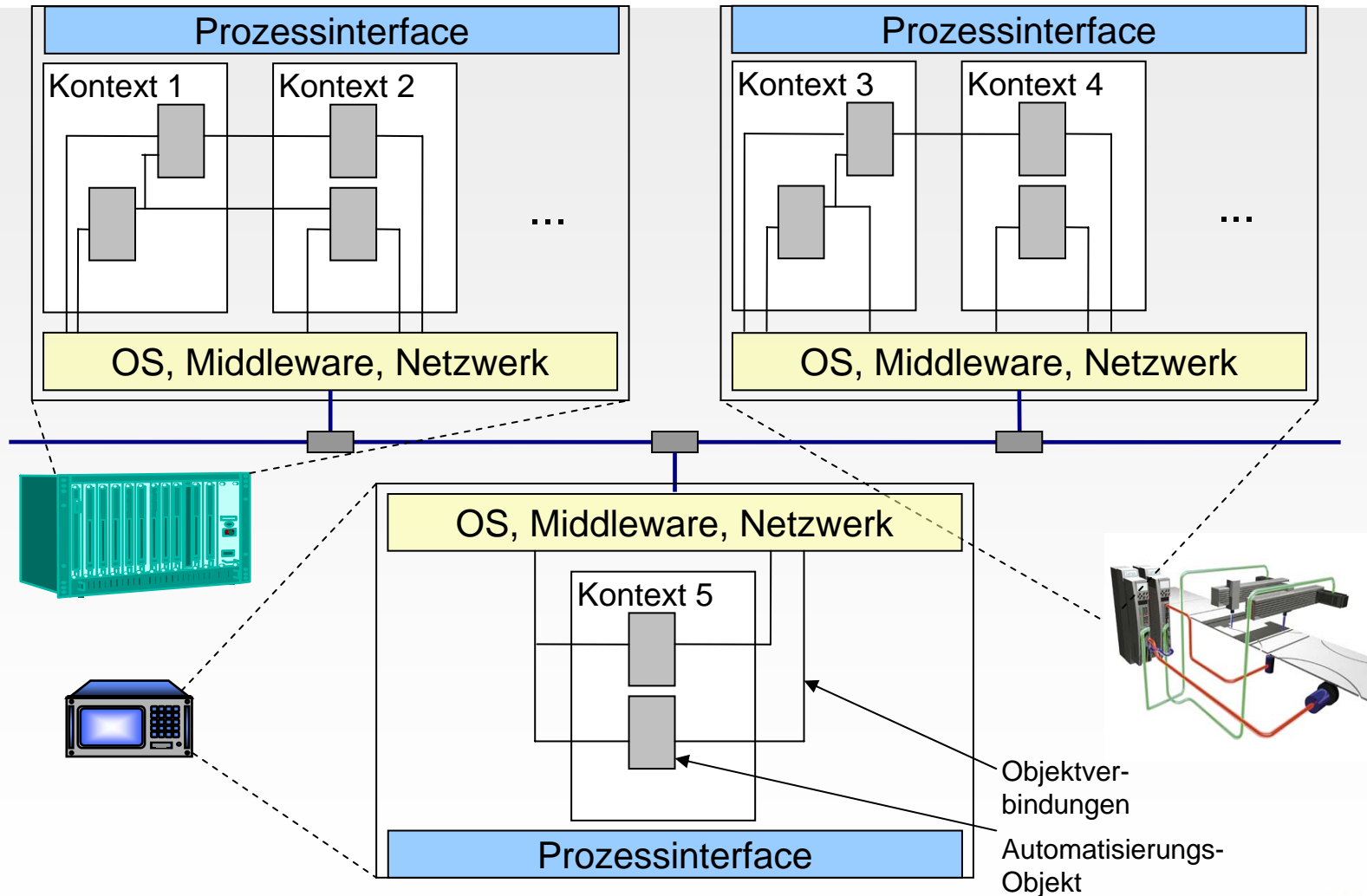
Motivation

Ziel: Verteiltes System



Motivation

Verteiltes Automatisierungssystem

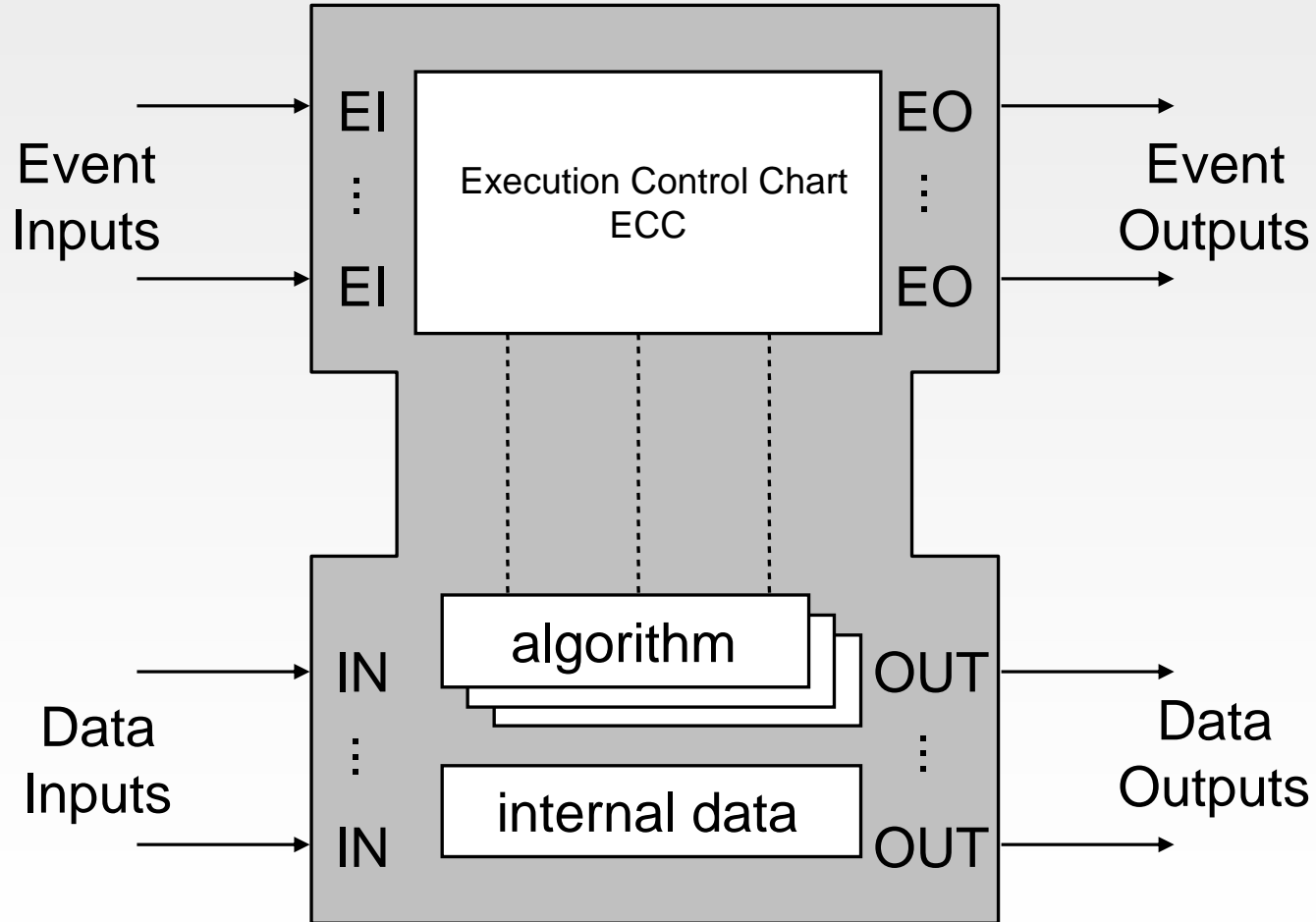


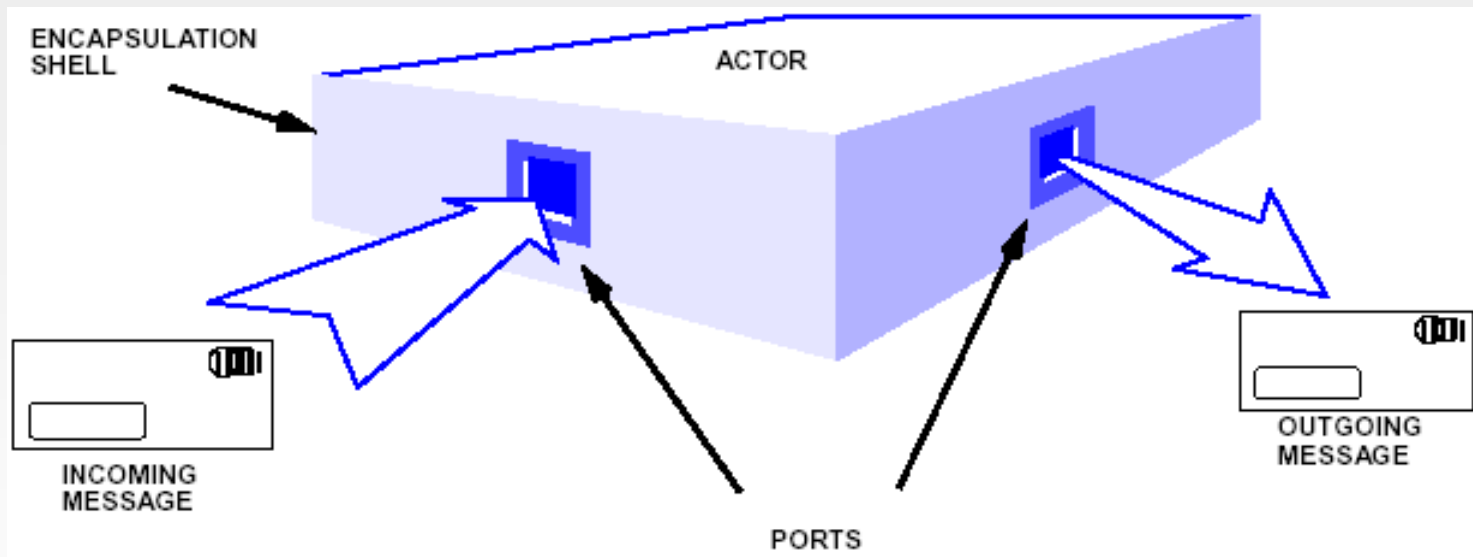
DOME - Distributed Object Model Environment

- ❑ 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 herstellerepezifischen Features als Wettbewerbsfaktor wird erheblich erschwert (Vergleichbarkeit der Produkte wird wesentlich erleichtert)

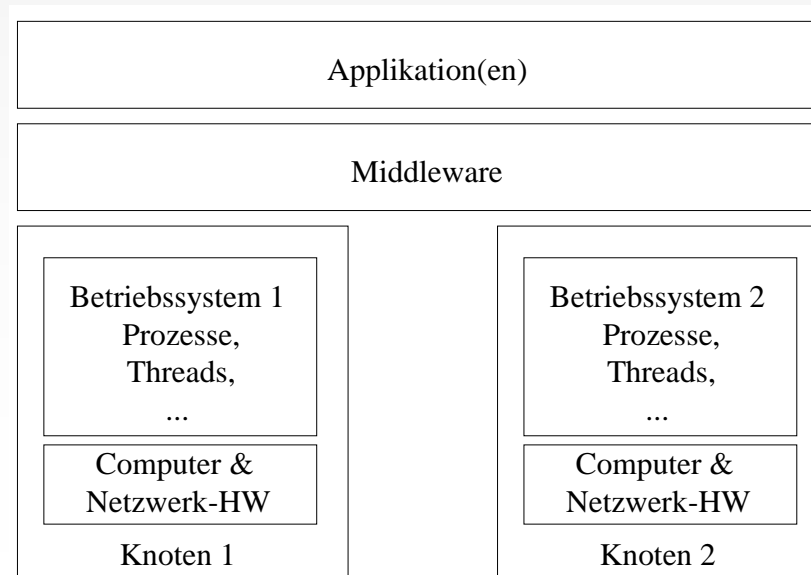
1. Motivation
2. Stand der Technik
3. Lösungsansatz / Validation
4. Zusammenfassung / Ausblick





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

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



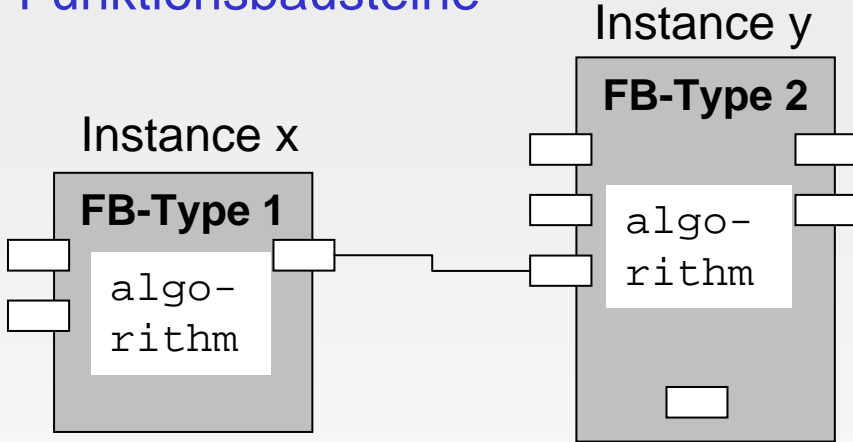
- ❑ **Entwicklungsmodelle / Standards**
 - Softwarelebenszyklus: IEC 12207
 - Qualitätsmerkmale: IEC 9126
 - Sicherheit: IEC 61508
 - RUP / oo-Paradigma
- ❑ **Anforderungen an Design & Codierung**
 - Quellcode ist entscheidender Faktor
 - „Quelle des Produkts“
 - Codegeneratoren
 - Programmierrichtlinien

metrikbasierte Qualitätssicherung für ausgewählte Paradigmen und Systemarten

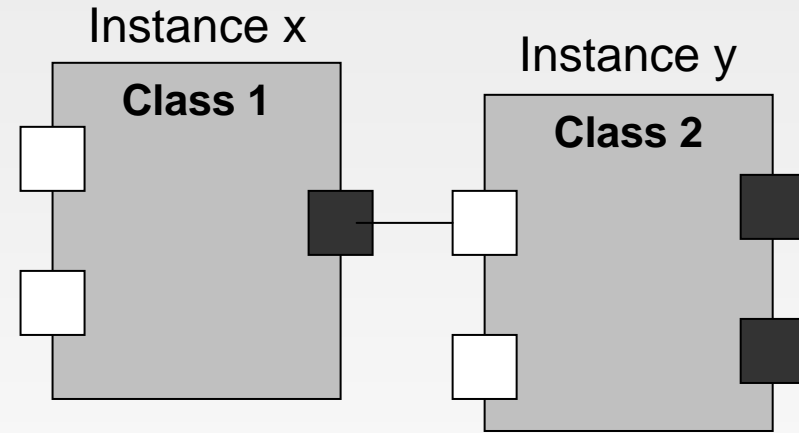


1. Motivation
2. Stand der Technik
3. Lösungsansatz / Validation
4. Zusammenfassung / Ausblick

Funktionsbausteine

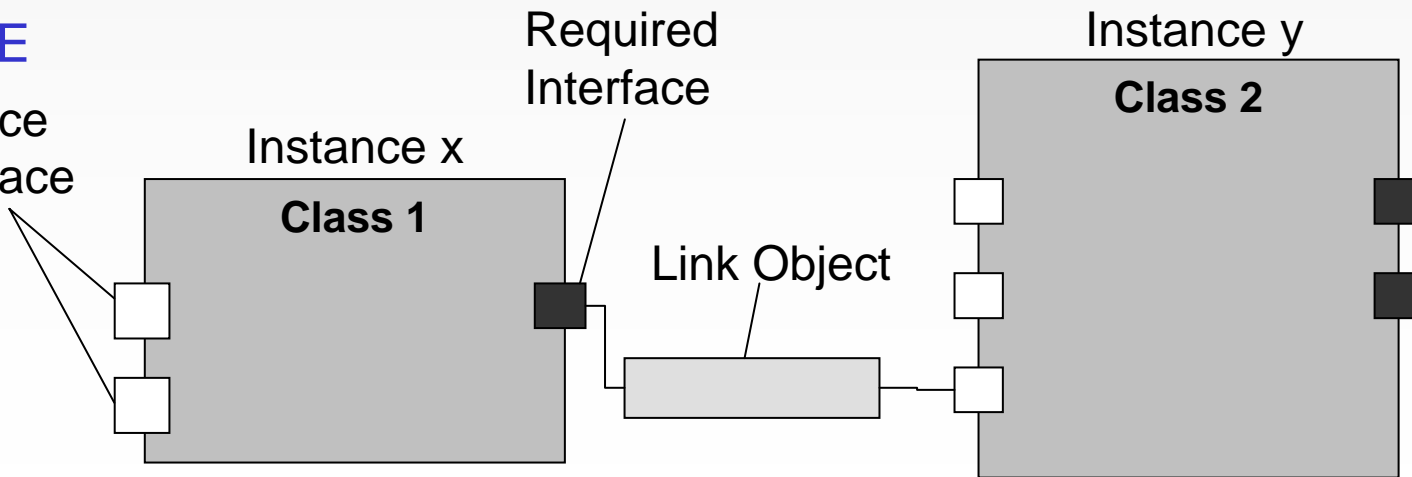


ROOM



DOMÉ

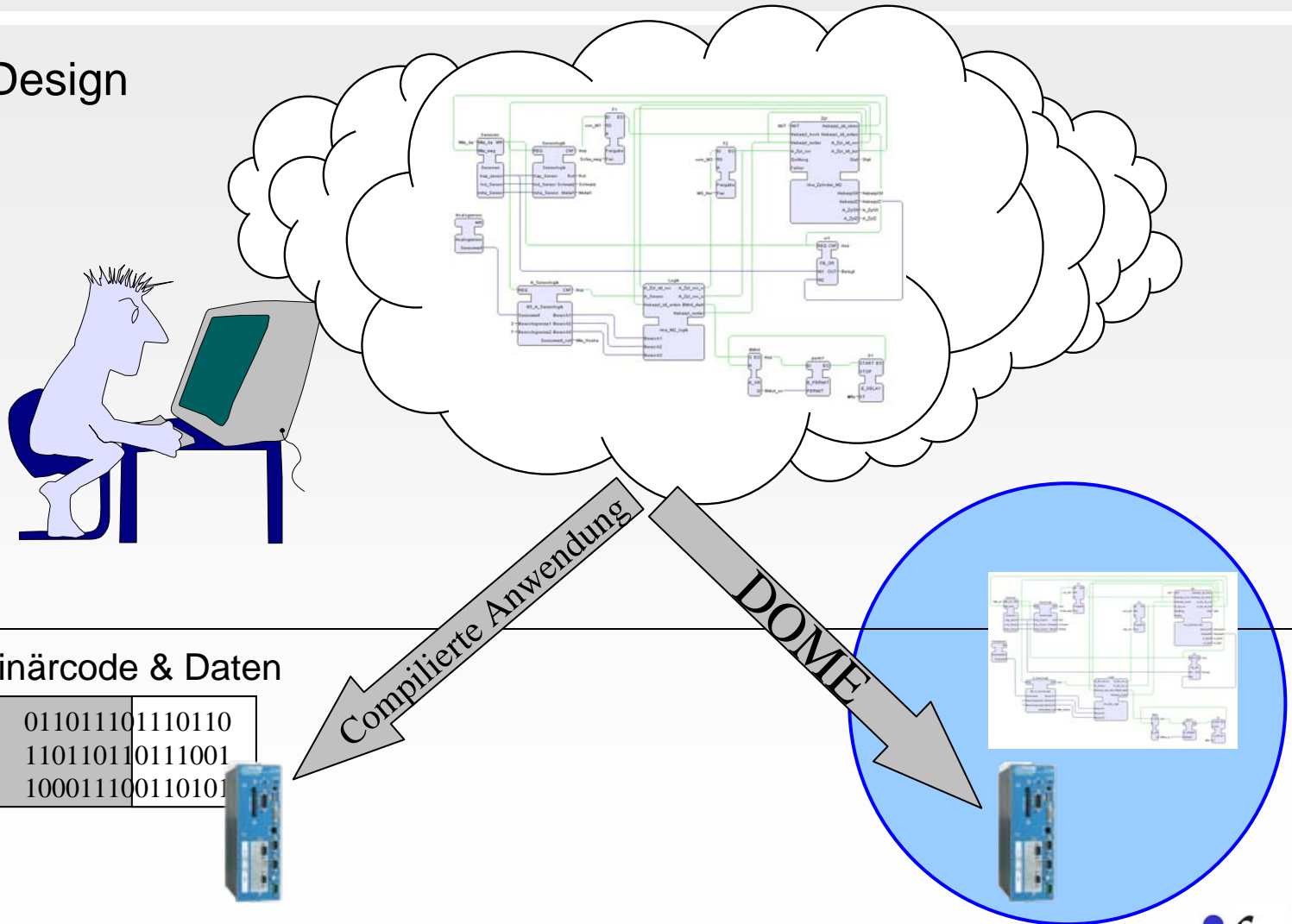
Service Interface



Lösungsansatz / Validation

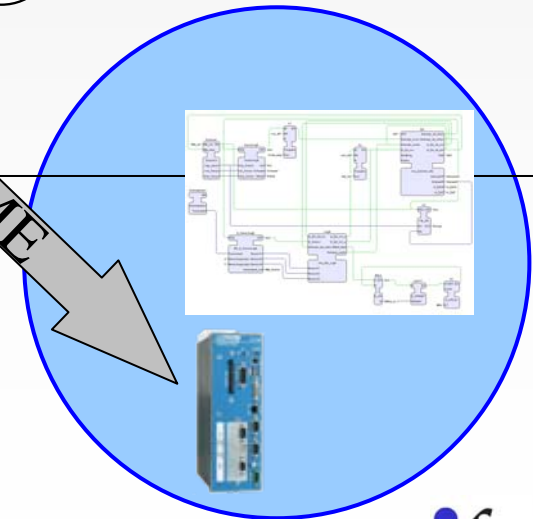
Objektnetzwerk auch zur Laufzeit verfügbar / Browsen der Applikation zu jedem Zeitpunkt möglich

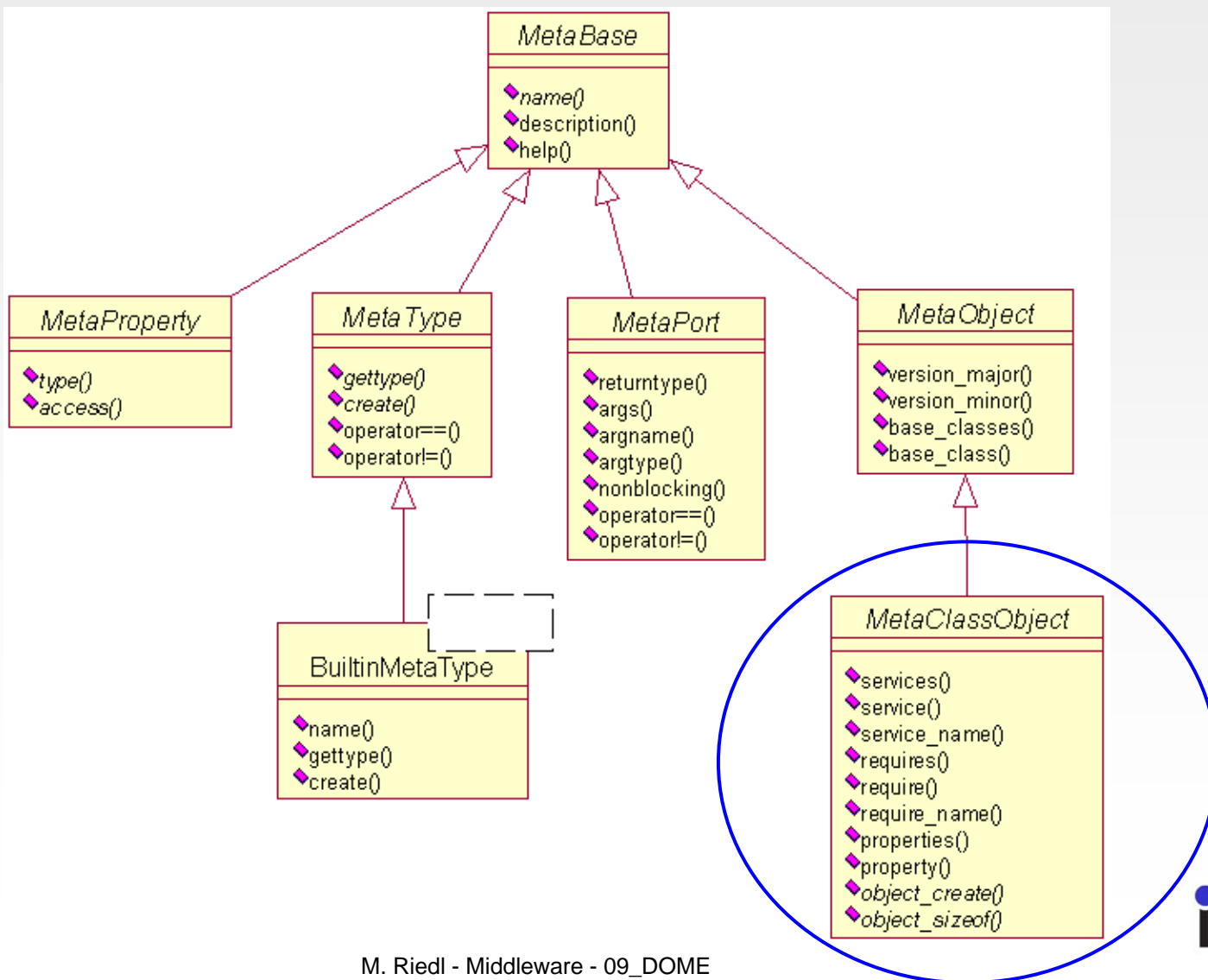
Entwurf / Design



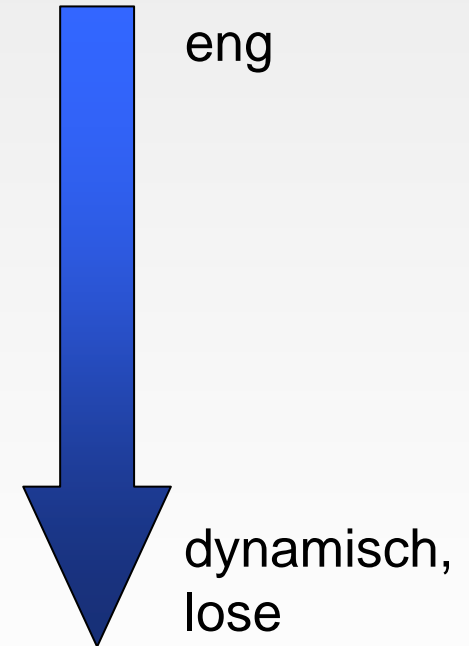
Binärcode & Daten

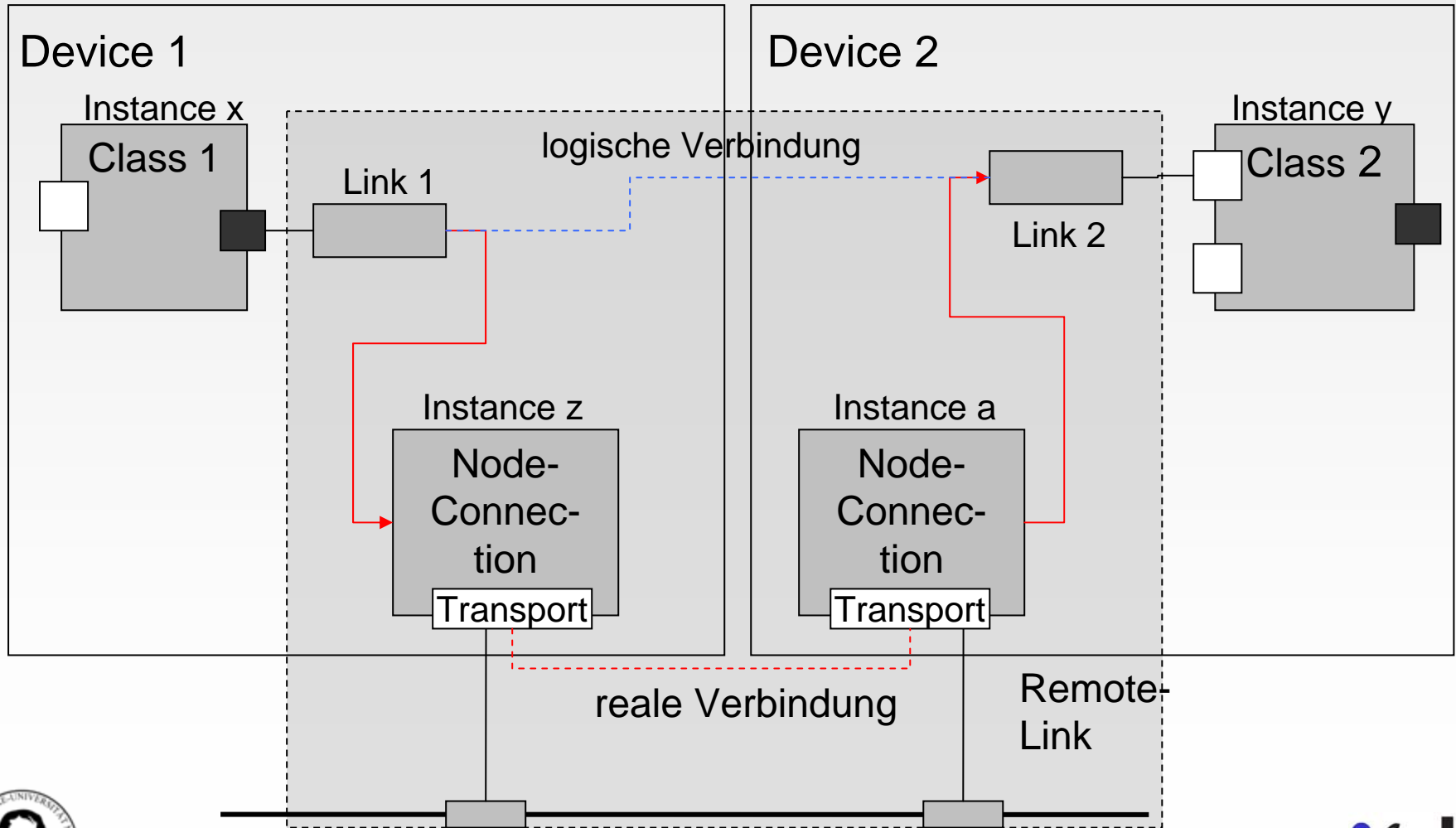
```
011011101110110
110110110111001
100011100110101
```





- Objektverbindungsarten
 - Lokal \rightarrow unterbrechungsfrei
 - Prozess-lokal
 - Synchron
 - Asynchron
 - Interprozess
 - Synchron
 - Asynchron



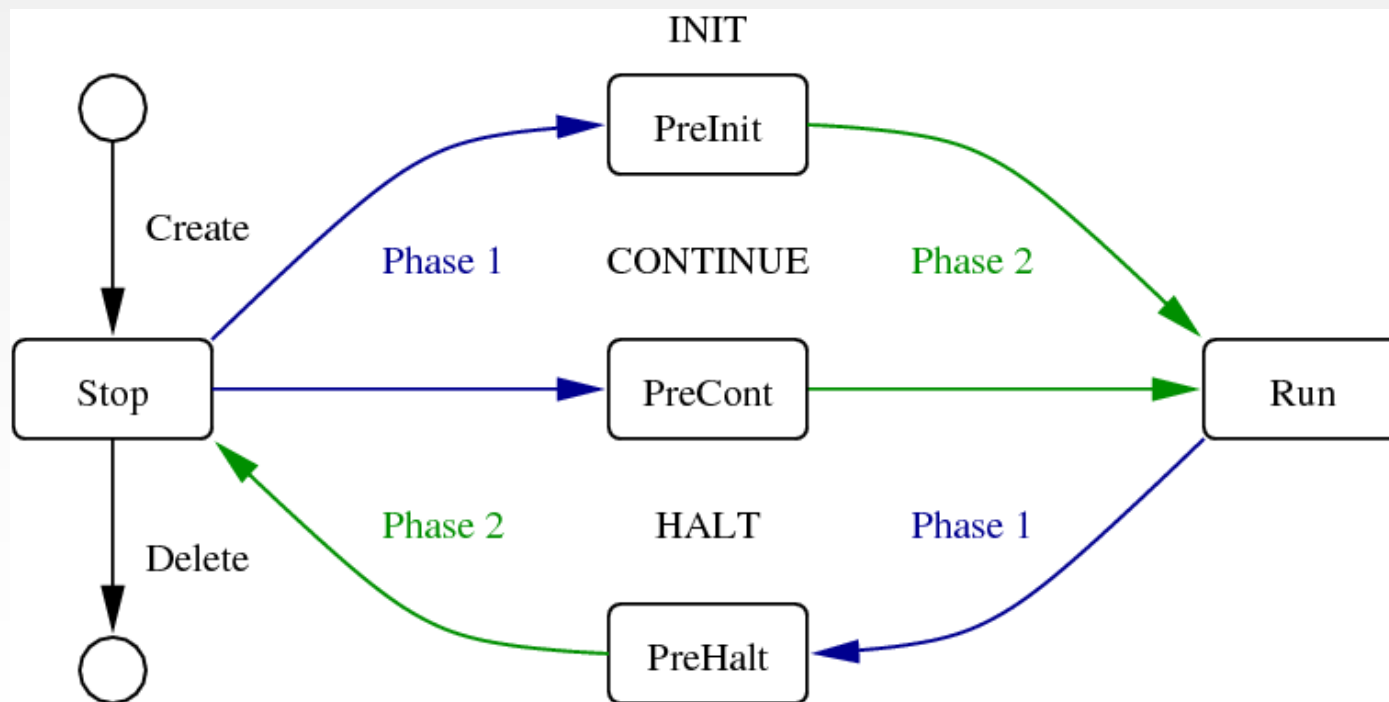


- ❑ Basis für rekonfigurierbare Systeme
 - Skalierbarkeit
 - Ausfallerkennung

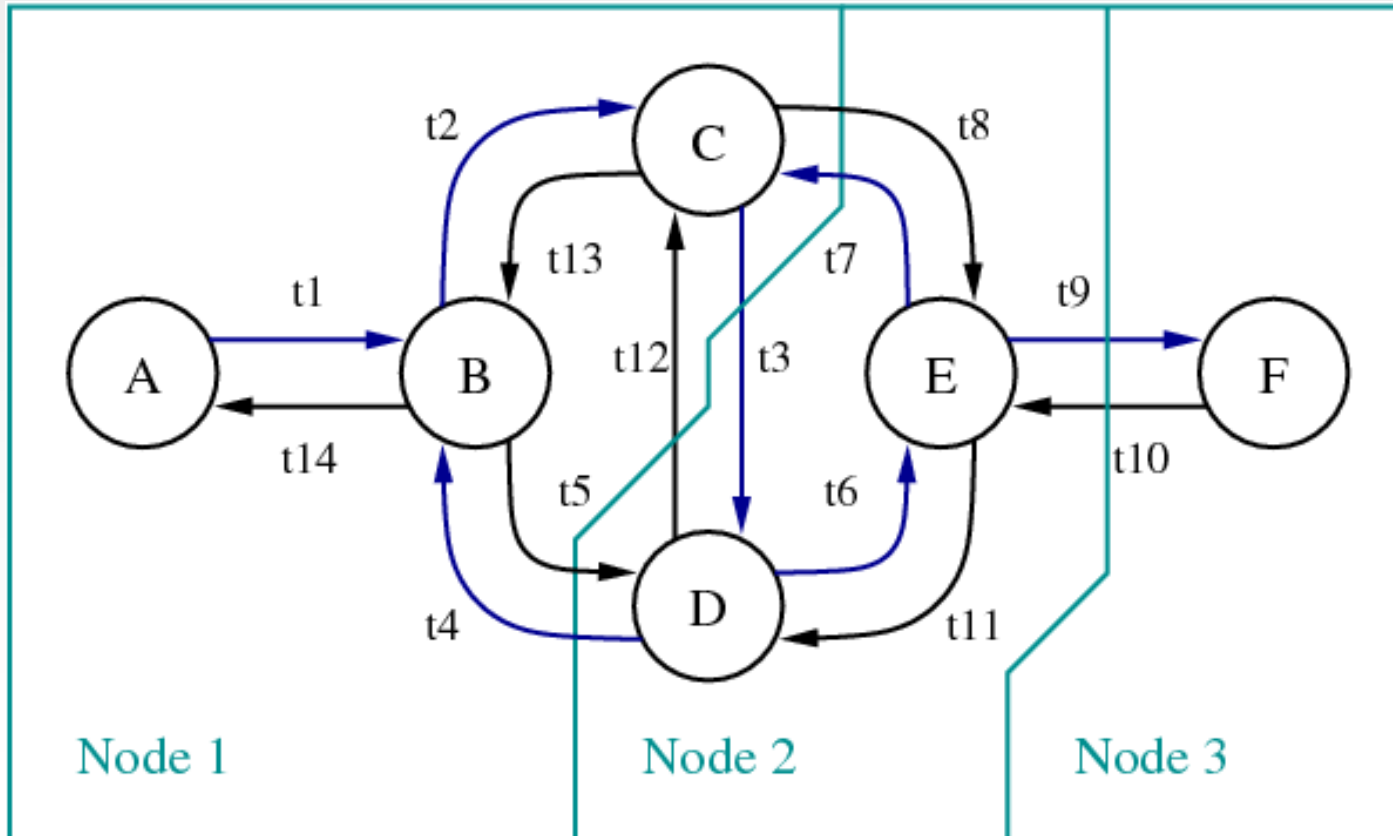
- ❑ Erforderliche Funktionalitäten
 - Automatische Wiederverbindung
 - Verteilter Anlauf

□ Zwei-Phasen-Commit-Protokoll

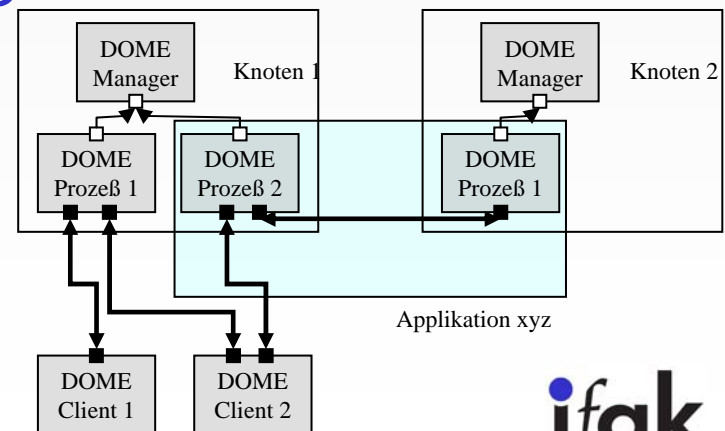
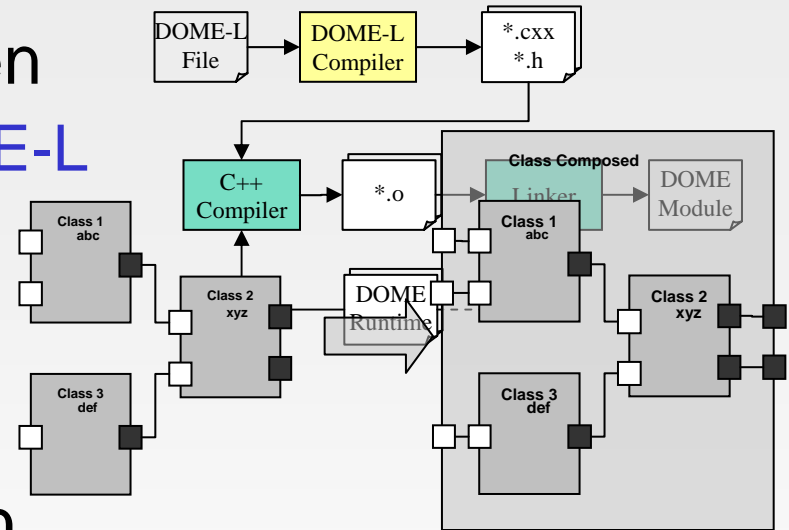
- Vorbereitung des neuen Zustands (z.B. Sperren der Ressourcen)
- Aktion



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



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



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

□ Qualitative Bewertung

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

□ Performance

□ Applikationsebene

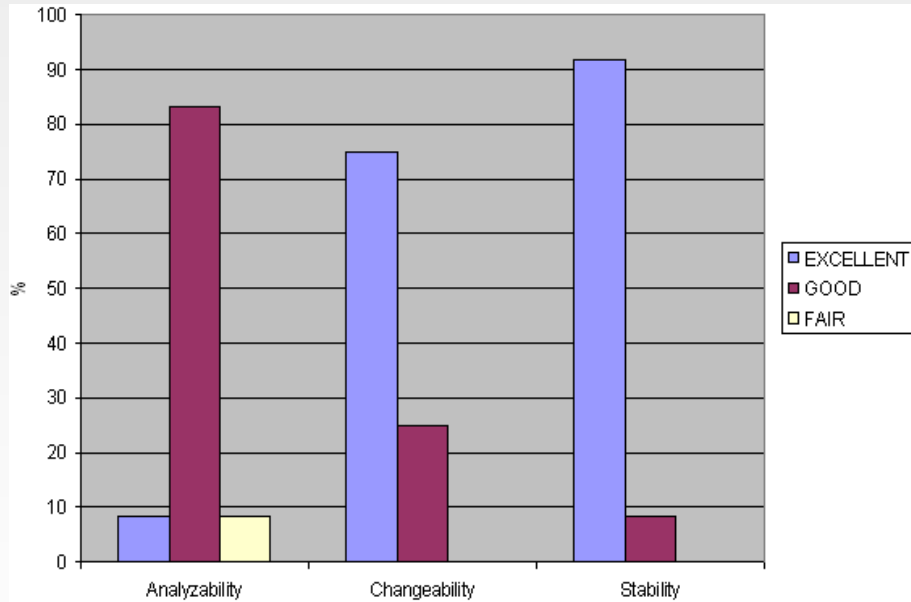
- (MOOD, Summe / Durchschnitt der CC, Durchschnittliche Kopplung)

| Kriterium | Bewertung |
|------------------|-----------|
| Analysierbarkeit | FAIR |
| Anpaßbarkeit | GOOD |
| Stabilität | FAIR |
| Testbarkeit | GOOD |

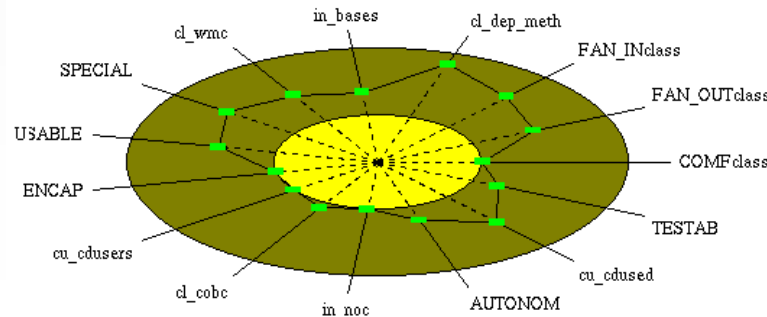
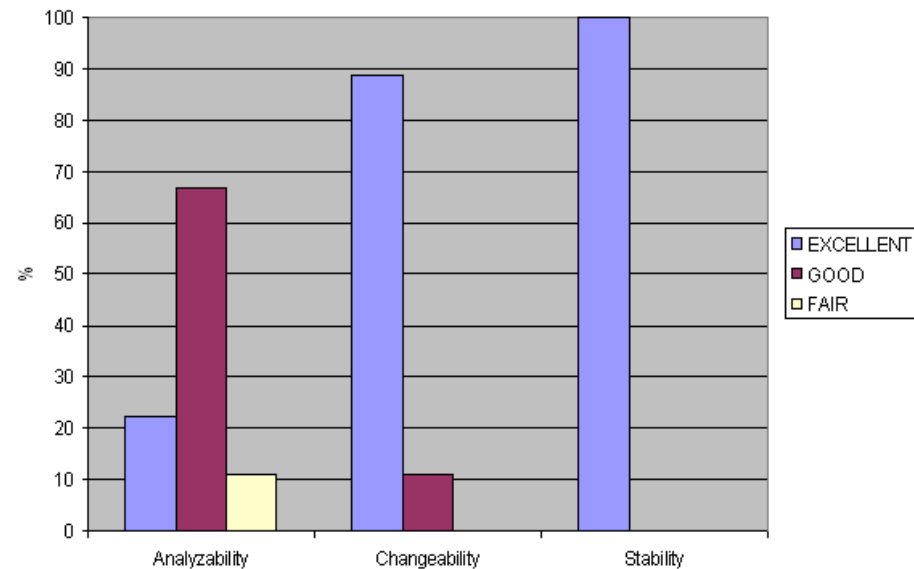
□ Beispiel: Bereichsüberwachung

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

IEC 61499



DOMÉ pur



- ❑ 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

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

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

□ 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

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

Nutzung von PROFIBUS

