Architekturen und Frameworks für zuverlässige und adaptive Informationssysteme

Christoph R. Hartel

27. Januar 2006

Vorbemerkung: Stand der Technik

”The most certain and effectual check upon errors which arise in the process of computation, is to cause the same computations to be made by separate and independent computers; and this check is rendered still more decisive if they make their computations by different methods.“

Vorbemerkung: Stand der Technik

”The most certain and effectual check upon errors which arise in the process of computation, is to cause the same computations to be made by separate and independent computers; and this check is rendered still more decisive if they make their computations by different methods.“

Dionysius Lardner, **1834**

Vorbemerkung: Stand d. Technik (2)

- Was soll uns das sagen?
  - Zahlreiche „neue“ Konzepte in der Literatur
    - Noch zahlreicher: Namen („Buzz Words“!?)
  - Im Grunde nur Varianten schon lange bekannter Ideen

➔ **Kernprinzipien** identifizieren!

Vorgehen

1. Grundlagen und Terminologie
2. Frameworks für DAIS
3. Prinzipien von DAIS-Architekturen
4. Architekturbeispiele
5. Realisierungsprobleme
6. Zusammenfassung
7. Ausblick

1. Grundlagen und Terminologie

- Adaptivität
- Zuverlässigkeit
- Architektur
- Framework

Grundlagen und Terminologie:

Adaptive Systeme

- Adaptivität vs. Adaptierbarkeit
- Self-*?
  - ... -adaptive, -healing, -modifying, -*conscious*
  - **Dyn. Veränderung der Zusammensetzung** vs. **Selbstmodifikation**
- Voraussetzungen
  - Mehrere Wege, ein Ziel zu erreichen
  - System hat Modell von sich selbst

Grundlagen und Terminologie:

Zuverlässige Systeme

- Zuverlässigkeit umfasst:
  - **Availability**
  - **Reliability**
  - Safety, Security, Integrity
    - Siehe spezielle Vorträge
  - Maintainability
    - „Uninteressant“
- Zuverlässigkeit vs. Adaptivität

Grundlagen und Terminologie:

Def. Architektur

- Die Architektur eines Informationssystems ist seine Organisation, d.h.
  - seine **Komponenten**,
  - deren **Beziehungen** untereinander und zu ihrer Umgebung und
  - die **Prinzipien**, welche das Design und die Entwicklung des Systems bestimmen.
  - (vgl. ANSI/IEEE Std. 1471-2000)

Grundlagen und Terminologie:
Def. Framework

- Ein Framework ist
  - ein **erweiterbares und anpassbares System** kollaborierender Softwareeinheiten,
  - das für eine **allgemeine, übergeordnete Aufgabenstellung**
  - **Kernfunktionalitäten** mit entsprechenden Bausteinen bereitstellt.

Grundlagen und Terminologie:

Def. Framework

- Ein Framework ist
  - ein **erweiterbares und anpassbares System** kollaborierender Softwareeinheiten,
  - das für eine **allgemeine, übergeordnete Aufgabenstellung**
  - **Kernfunktionalitäten** mit entsprechenden Bausteinen bereitstellt.

2. Frameworks für DAIS

- **Es gibt keine!**

2. Frameworks für DAIS

- **Es gibt keine!**
- In der Literatur finden sich zahlreiche „Frameworks“ für DAIS.
- Problem:
  - Entweder keine Bereitstellung von Kernimplementierungen
    → **Architektur**
  - oder nicht auf Erweiterbarkeit ausgelegt
    → **Referenz-Implementierung**

3. Architekturprinzipien

- Redundanz
- Komponentenbasierung
- Architectural Reflection
- Separation of Concerns

Architekturprinzipien:

Redundanz

- Def.: Redundanz ist das mehrfache Vorhandensein funktionsgleicher Systembestandteile.

- 2 Arten der Redundanz (Vgl. Lardner):
  - **Identische Redundanz**
    - Eignung für Software?
  - **Diversifizierende Red.** („Design Diversity“)
    - Kosten/Nutzen?

Architekturprinzipien:

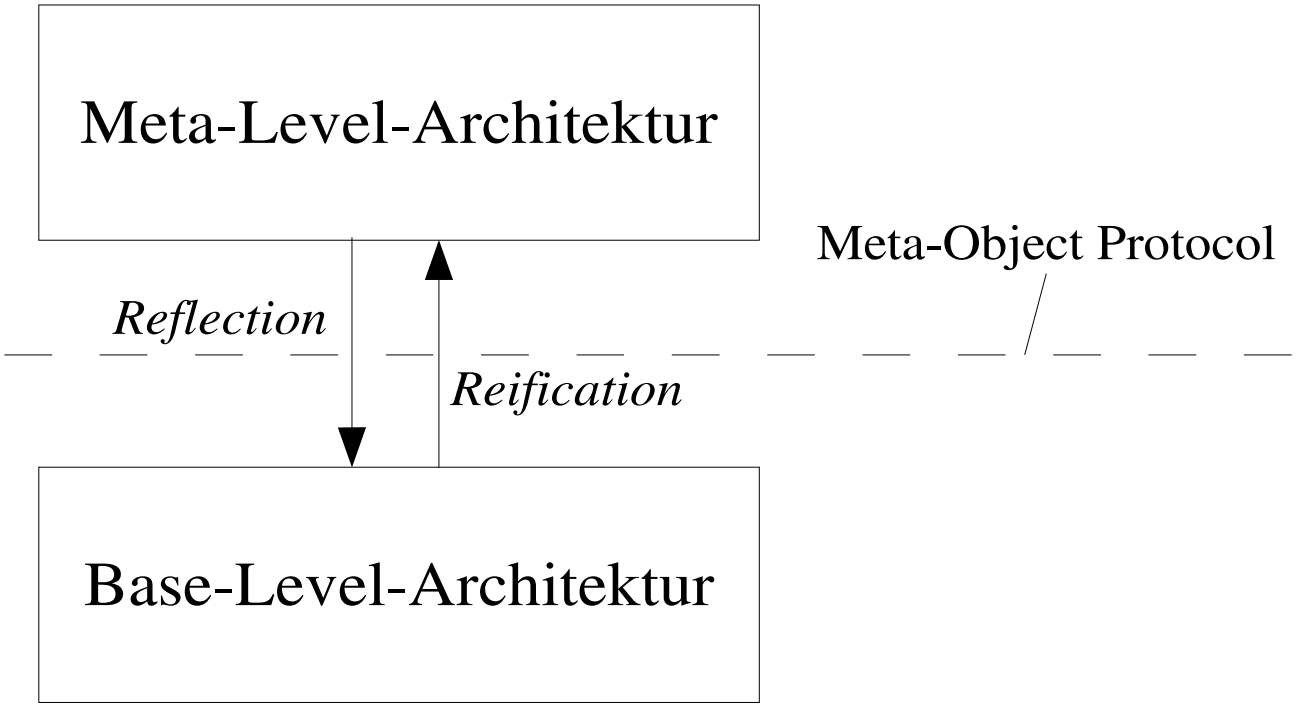
Komponentenbasierung

- Unabhängigkeit
  - Lose Kopplung, ...
- Austauschbarkeit
  - Wichtig:
    Verwendbarkeit von Common-off-the-Shelf-Komponenten
- Komplexitätsreduktion
- Kostenreduktion
  - Vgl. Design Diversity

Architekturprinzipien:

Architectural Reflection

➔„System hat Modell von sich selbst“



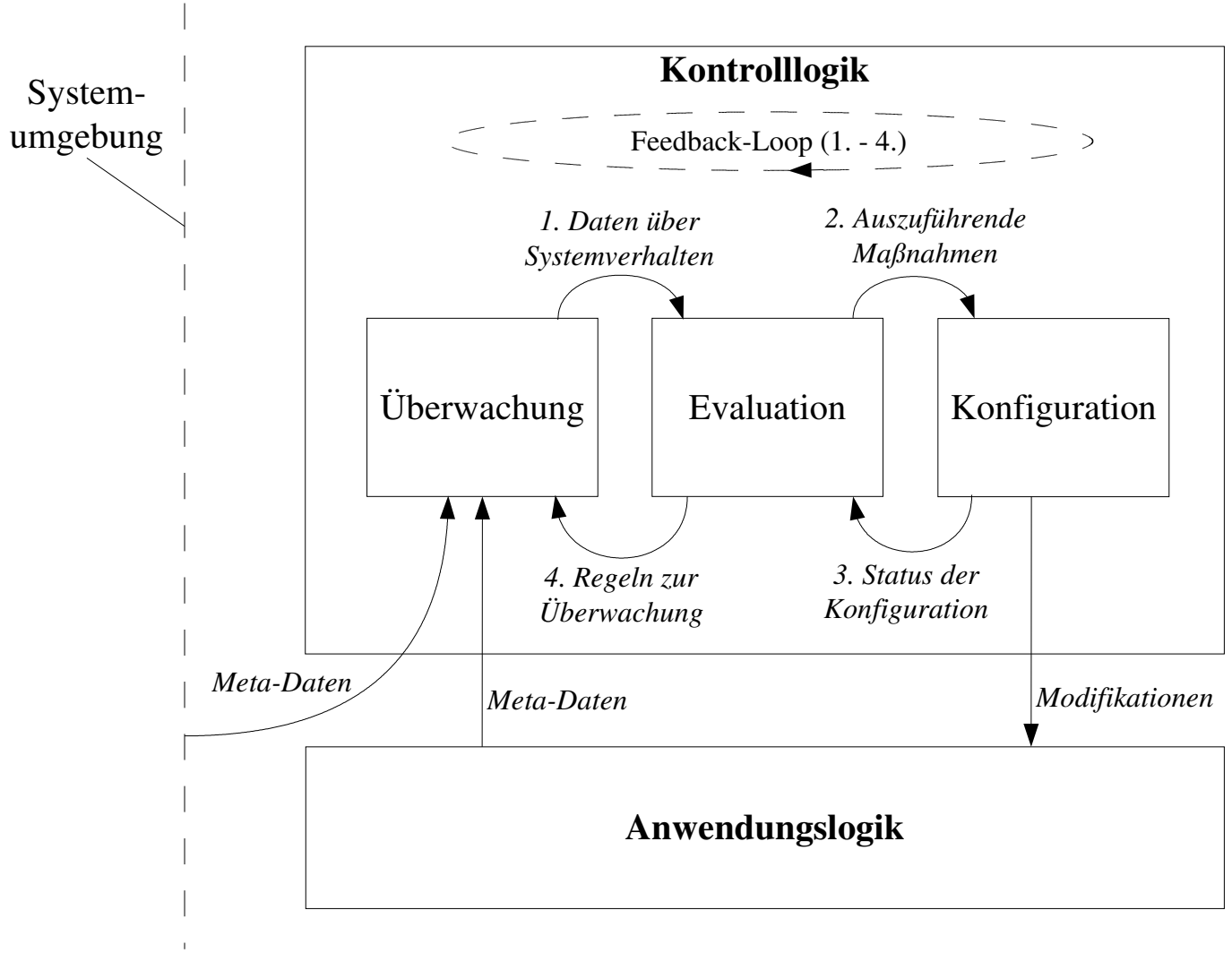
Architekturprinzipien:

Separation of Concerns

- Klare Trennung *logischer* Systembestandteile
  - != Aufteilung der Implementierung
- Ziel:
  - Reduktion der Komplexität
    - Beherrschbarkeit!
  - Ausreichende Berücksichtigung aller wesentlichen Aspekte

Architekturprinzipien:

Separation of Concerns (2)

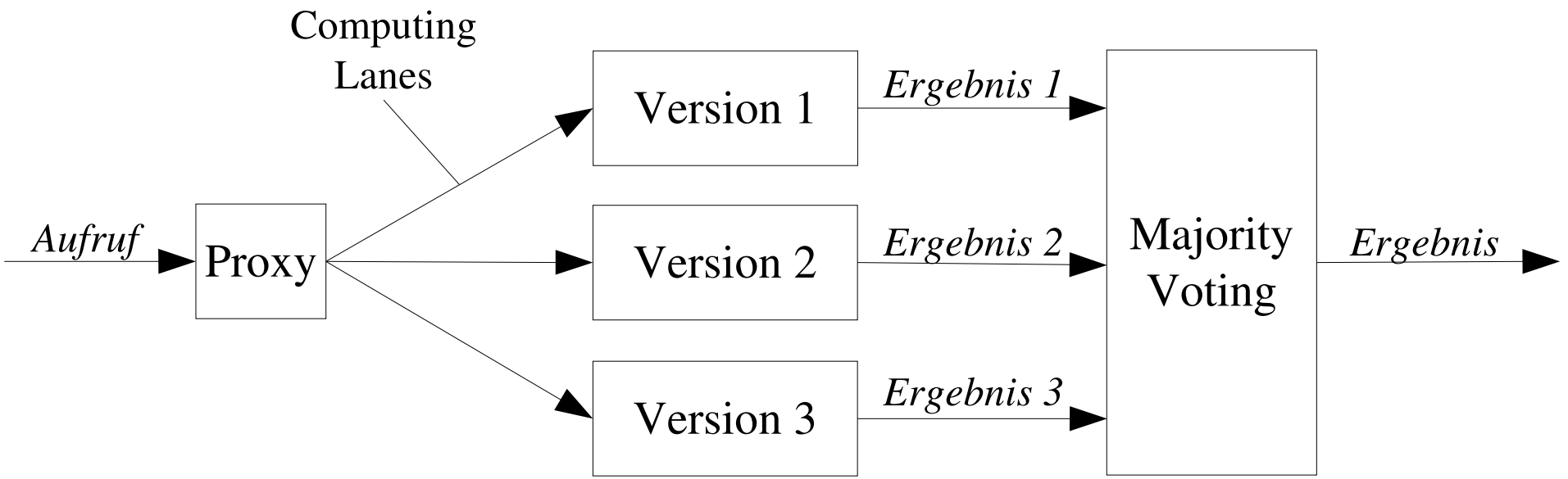


4. Architekturbeispiele

- **Zuverlässige Architekturen**
  - N-Version-Software
  - Component Redundancy
- **Adaptive Architekturen** (Ansätze!)
  - Dynamic Dispatch
  - Distributed Configuration Routing

Architekturbeispiele (Zuverlässige IS):

N-Version-Software

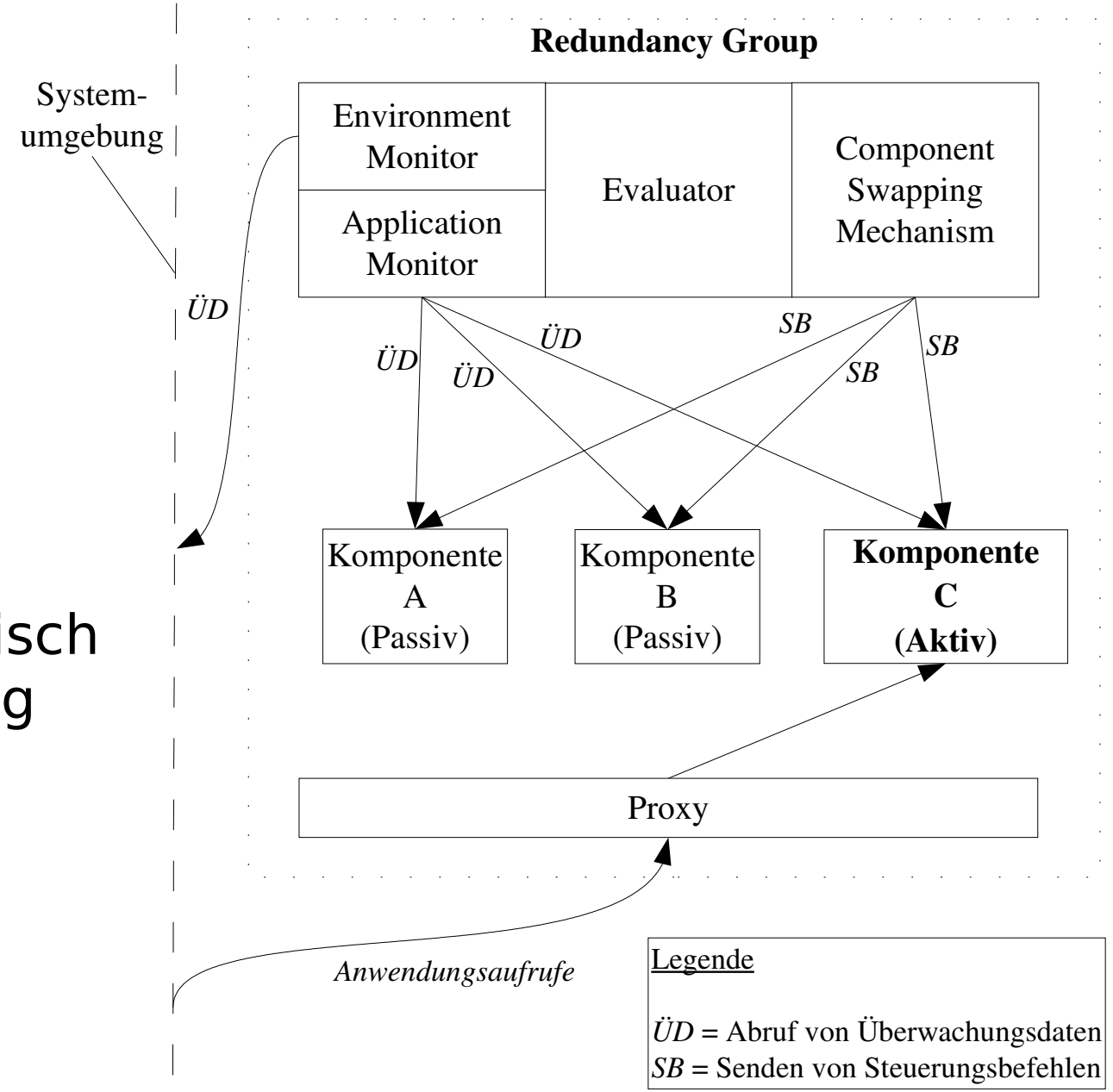


- Circa 1977
- Anlehnung and Hardwaresysteme
- Architektur?

Architekturbeispiele:

Component Redundancy

- Circa 2003
- Klare SoC
- Verändert dynamisch Zusammensetzung
- Historie von Meta-Daten



Architekturbeispiele (Adaptive IS):

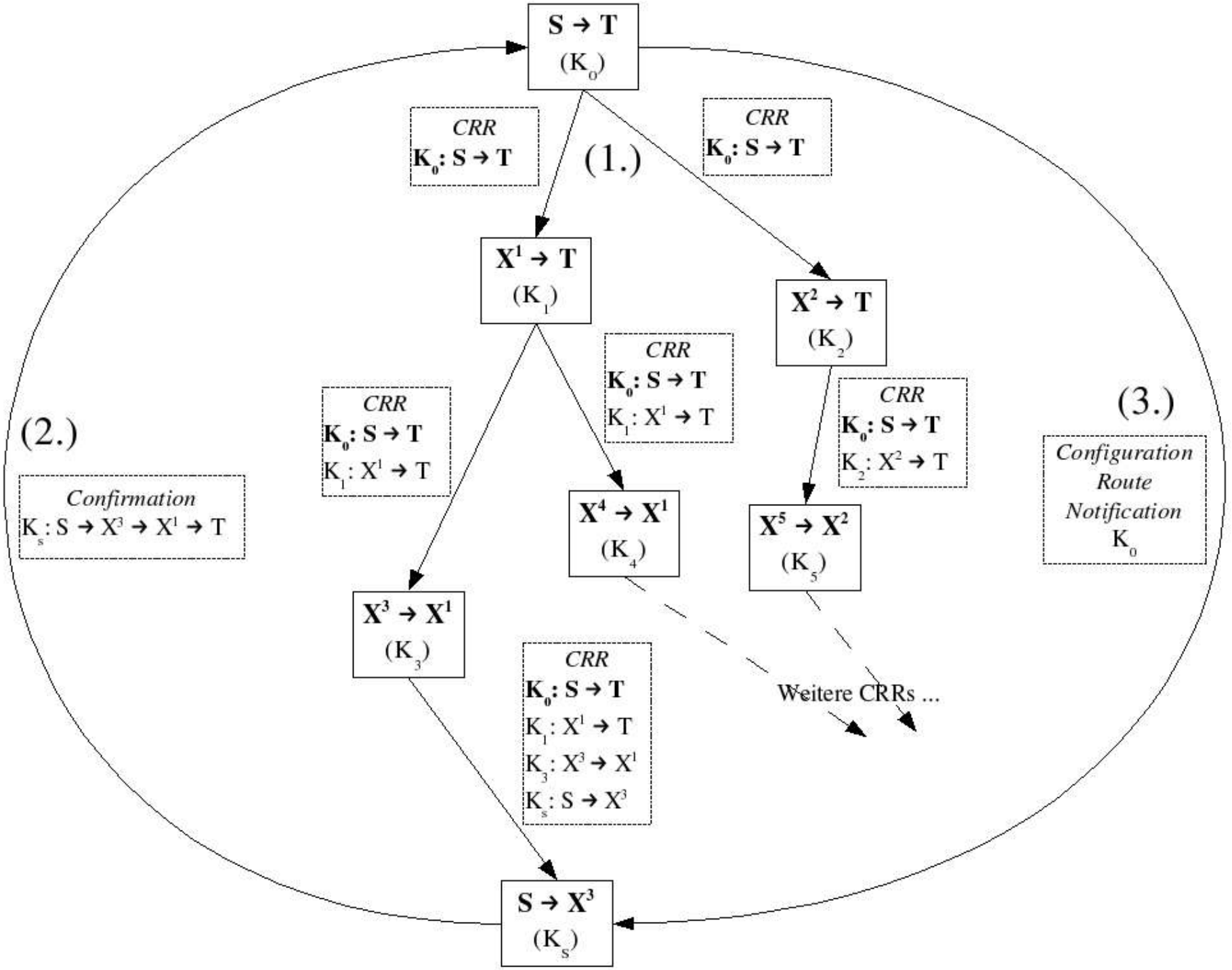
Dynamic Dispatch

- Redundanz auf Methoden-Ebene
- Dispatcher wählt zur Laufzeit eine aus N Methoden-Implementierungen
- **Probabilistic Dispatch**
  - Kriterium: (Durchschnittliche) Wahrscheinlichkeit, dass Methode erfolgreiche terminiert
- **Expected Utility Dispatch**
  - Kriterium: Erwarteter Nutzen, abhängig z.B. von Ergebnisgenauigkeit und Laufzeit

Architekturbeispiele (Adaptive IS):

Distributed Configuration Routing

- **Ziel:**
  - Konfiguration flexibler gestalten:
    - Keine hartkodierten Konfigurations-Strategien
    - Mögliche Aufspaltung in Teil-Strategien
- **Idee:**
  - Jeder Dienst = Zustandsänderung
  - Daher ist auch Anpassung einer Komponente Zustandsänderung
  - Kontrolllogik enthält „Adaptor-Komponenten“


S → T
(K0)
CRR
K0: S → T
(1.)
CRR
K0: S → T
X1 → T
(K1)
X2 → T
(K2)
CRR
K0: S → T
K1: X1 → T
CRR
K0: S → T
K2: X2 → T
(2.)
CRR
K0: S → T
K1: X1 → T
(3.)
Configuration
Route
Notification
K0
Confirmation
Ks: S → X3 → X1 → T
X4 → X1
(K4)
X5 → X2
(K5)
X3 → X1
(K3)
CRR
K0: S → T
K1: X1 → T
K3: X3 → X1
Ks: S → X3
Weitere CRRs ...
S → X3
(Ks)

5. Realisierungsprobleme

- (Einige) Probleme:
  - Vergleichbarkeit von Ergebniswerten
  - Zustandstransfer
  - Ähnlichkeit von Schnittstellen
  - Seiteneffekte ➔ Isolation
  - Unabhängigkeit von Faults?
  - ...
- Lösungen?

6. Zusammenfassung

- Zuverlässigkeit und Adaptivität eng verwoben
- Adaptivität hat viele Facetten
  - Realistisch: Dynamische Veränderung der Systemzusammensetzung
- Echte DAIS-Frameworks existieren nicht
- Wesentliche Konzepte schon sehr alt, nur wenige tatsächliche Neuerungen
- Umsetzung der Konzepte problematisch

7. Ausblick

- Umfassende Taxonomien adaptiver Systeme
- Konsolidierung der Konzepte u. Konzentration auf Lösung vorhandener Probleme!
- Schaffung von DAIS-Frameworks basierend auf den existierenden Architekturen
- Transfer von Erfahrungen aus transaktionalen Systemen?

Fragen...

?