28 Aspektorientierte Entwurfsmethoden (Aspektorientierte Zerlegung)

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de

Version 12/13-1.1, 09.01.13

(Versions above 1.0 are corrections after lecturing)

1) Aspektorientierte Zerlegung
2) Essentielle Zerlegung

Obligatorische Literatur

- View models (Wikipedia)

Andere Literatur

- De Marco, T.: Structured Analysis and System Specification; Yourdon Inc. 1978/1979. Siehe auch Vorlesung ST-2
- McMenamin, S., Palmer, J.: Strukturierte Systemanalyse; Hanser Verlag 1988
- Steve Vinoski. An Overview of Middleware. In: A. Llamos ı and A. Strohmeier (Eds.): Ada-Europe 2004, LNCS 3063, pp. 35–51, 2004. Springer-Verlag Berlin Heidelberg 2004

Ziel

- Lerne die verschiedenen aspektorientierten Zerlegungsstrategien kennen
- Verstehe den Unterschied zwischen D&C-Zerlegung und aspektorientierter Zerlegung

28.1 Aspektorientierte Zerlegungen

Entwicklung mit Zerlegungsstrategien

- Bisher waren die Zerlegungsstrategien der Entwurfsmethoden **uniform**, d.h. Es wurde nach einem einzigen Kriterium zerlegt (divide&conquer-Zerlegungen)
- Ausgehend vom System als Ganzes (Kontextmodell) wird schrittweise top-down-verfeinert:
  - Funktionsorientierte Zerlegung
  - Aktionsorientierte Zerlegung
  - ECA-basierte Zerlegung
  - Datenorientierte Zerlegung
  - Objekt-orientierte Zerlegung
  - Komponentenorientierte Zerlegung
  - Transformative Zerlegung
- **Aspektorientierte Entwicklung** zerlegt das Kontextmodell in **mehrere** Perspektiven gleichzeitig

Perspektivenmodelle, Viewpoints and Views

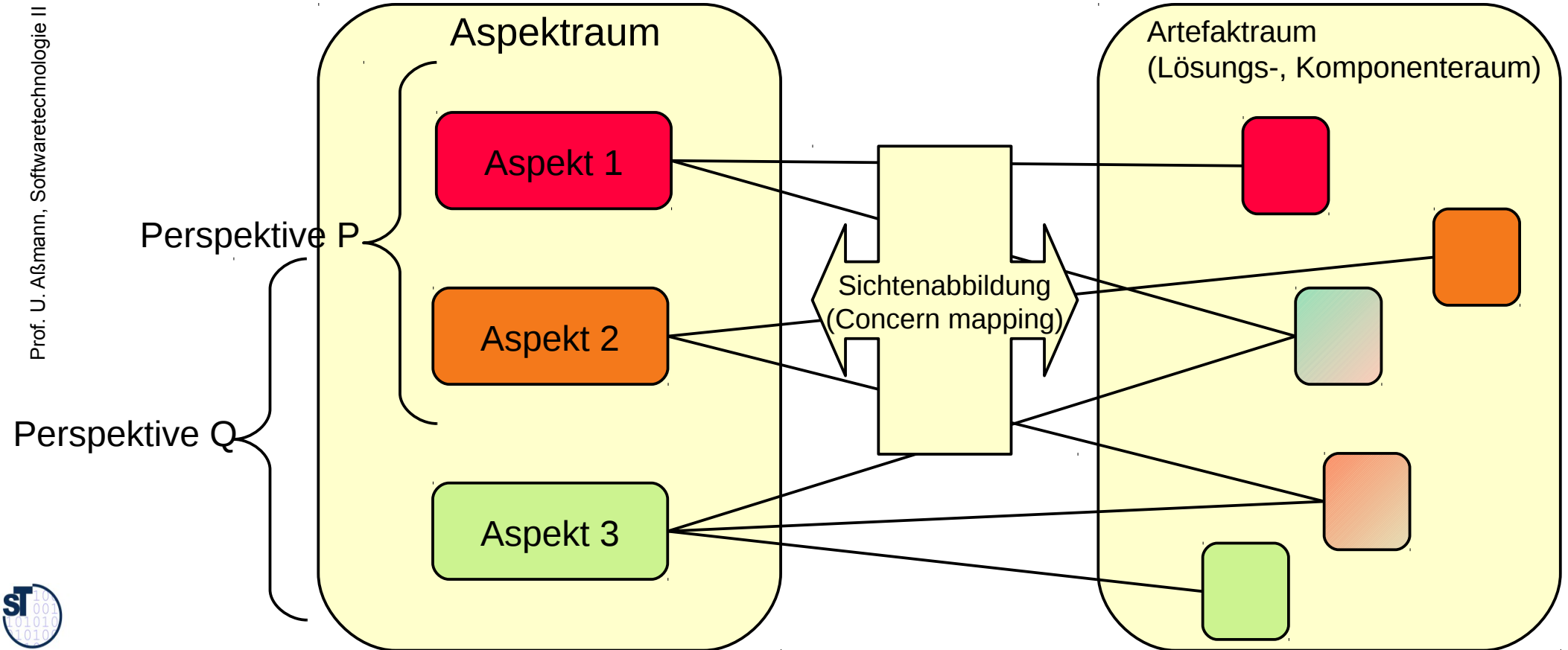
- Ein **Komponentenraum** (oft auch **System-, Artefakt-** oder **Lösungsraum** genannt) besteht aus einer Menge von Modell- und Programmfragmenten, -komponenten
  - Ein Artefaktraum kann ein einziges oder auch mehrere Softwaresysteme enthalten
- Ein **Aspekt (Belang)** bildet einen einfachen Blickwinkel (viewpoint) auf ein System oder einen Artefaktraum
- Ein **Belangraum (Aspektraum)** besteht aus einer Menge von Belangen (Aspekten, aspects, concerns) und einer algebraischen Struktur, die sie gliedert
- Ein **Sichtenbildungsraum** besteht aus einem gekoppelten Belang- und Artefaktraum, mit dem Sichten gebildet werden können
- Eine **Perspektive (viewpoint)** besteht aus einer Menge von Aspekten (Belangen, concerns)
- **Perspektivenmodelle (Viewpoint models, view models)** sind systematische Modelle für aspektorientierte Zerlegungen
  - RM-ODP
  - Zachmann framework
  - MDA

Bereits bekannte Zerlegungsstrategien sind aspektorientiert

- In der Literatur sind verschiedene **aspektorientierte** oder **mehrdimensionale Zerlegungsstrategien** bekannt:
- **Dokument-Zerlegung in Aspekte**
  - Struktur (html, XML)
  - Layout (CSS, FO)
  - Reaktion (javascript)
- **Essentielle Zerlegung** (EAI-Zerlegung) der Strukturierten Analyse unterscheidet folgenden Aspekte bzw Gesichtspunkte (viewpoints):
  - Essentielle Aktivitäten, Datenstrukturen und ihre Speicher
  - Administrative Aktivitäten (zur Prüfung von Daten)
  - Infrastruktur-Aktivitäten (zur Kommunikation und Anpassung an Plattform)
- **Architektonische Zerlegung** zerlegt in die Aspekte und Gesichtspunkte (siehe ST-1):
  - Architektur
  - Anwendung
- **Blutgruppen-Zerlegung** (Siedersleben)
  - Anwendung
  - Technik
- **Platformorientierte Zerlegung** zerlegt in die Aspekte (siehe Kapitel “MDA”:
  - Essentielle Aktivitäten
  - Platform 1 .... Plattform n

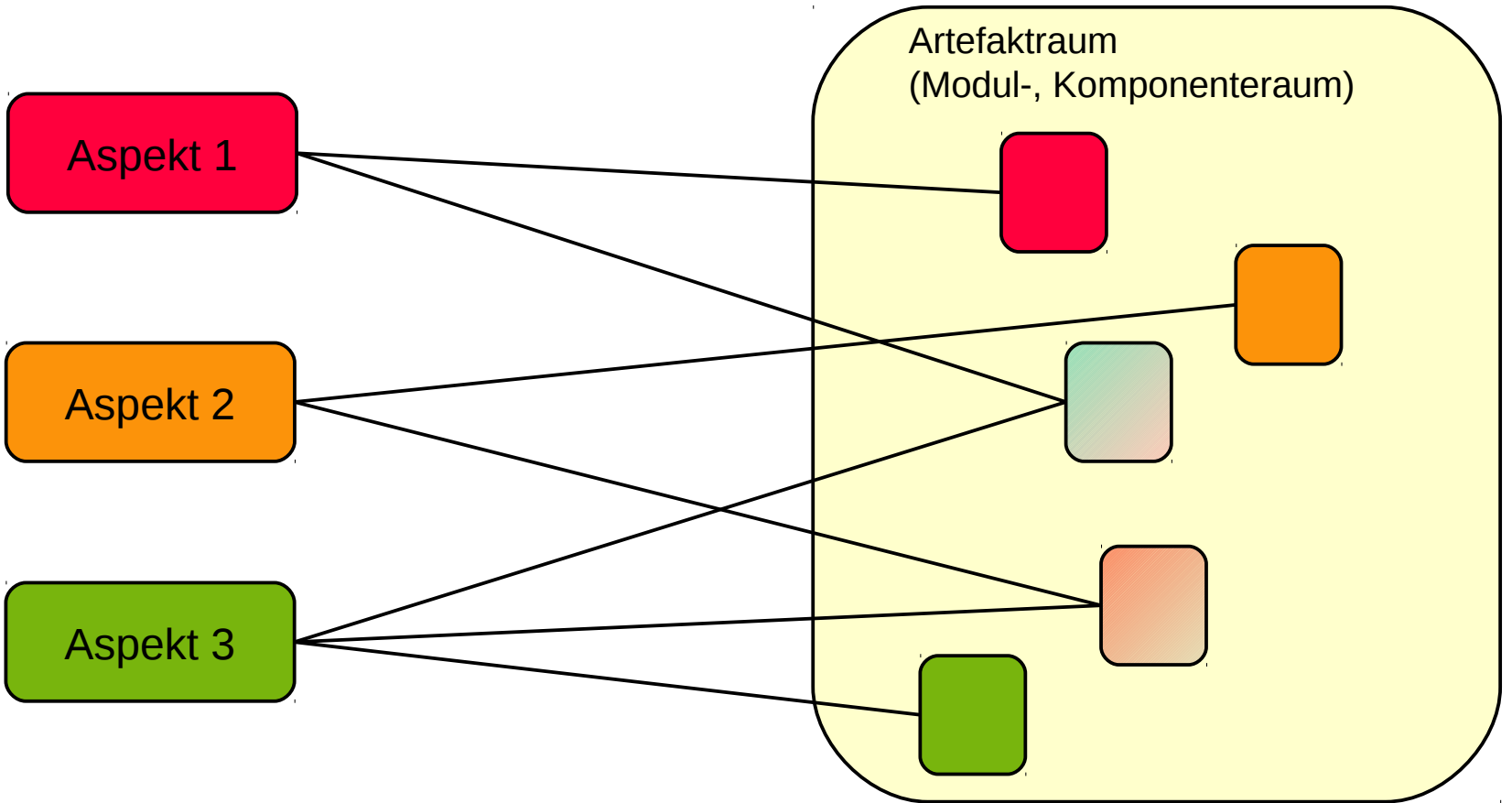
Aspekte und Blutgruppen für Software und Dokumente

- Man kann die Aspekte als Farben oder Blutgruppen von Lösungseinheiten (Modulen, Modellen, Modellelementen, -fragmenten) sehen
- Dann ist die Software gegliedert in einen **Sichtenbildungsraum (SoC space)**
  - Aspektraum (Belangraum, concern space) mit Belangen (Farben), die in Perspektiven (viewpoints) gegliedert sein können
  - Komponentenraum (Lösungsraum) enthält die Komponenten des Softwaresystems
  - Abbildung dazwischen (Concern mapping) definiert views (Sichten, slices, Scheiben, Schicht), die korreliert sind zu einem Aspekt oder einer Perspektive (viewpoint)



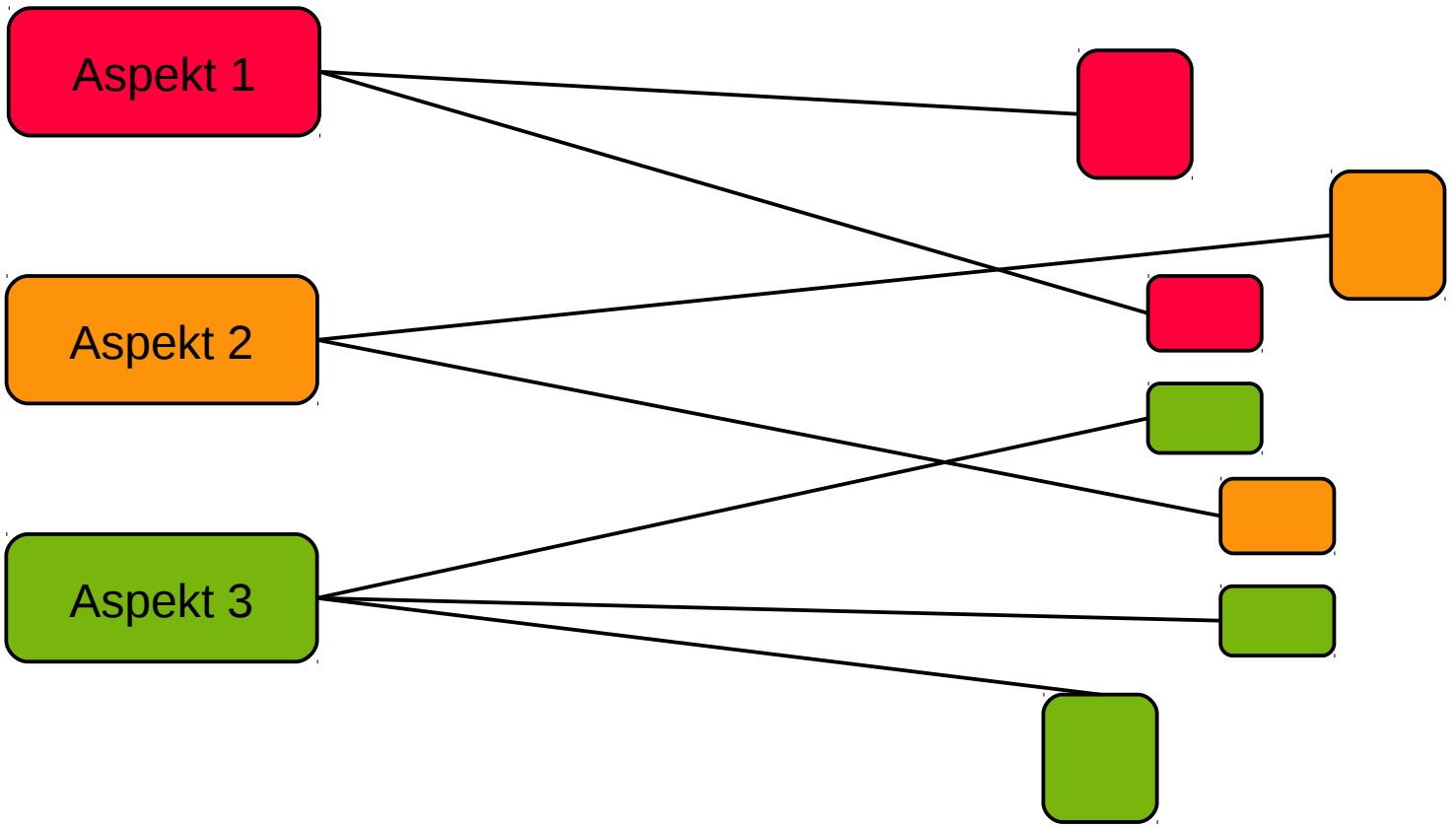
Aspektmarkierung ist unabhängig von Abstraktionsebene

- Aspekte für Module oder Komponenten:
  - Für alle Module: ordne Aspekte zu
  - Für alle Komponenten: ordne Aspekte zu
- Aspekte für Aktivitäten, Speicher, Stellen:
  - Für alle Modellelemente: ordne Aspekte zu



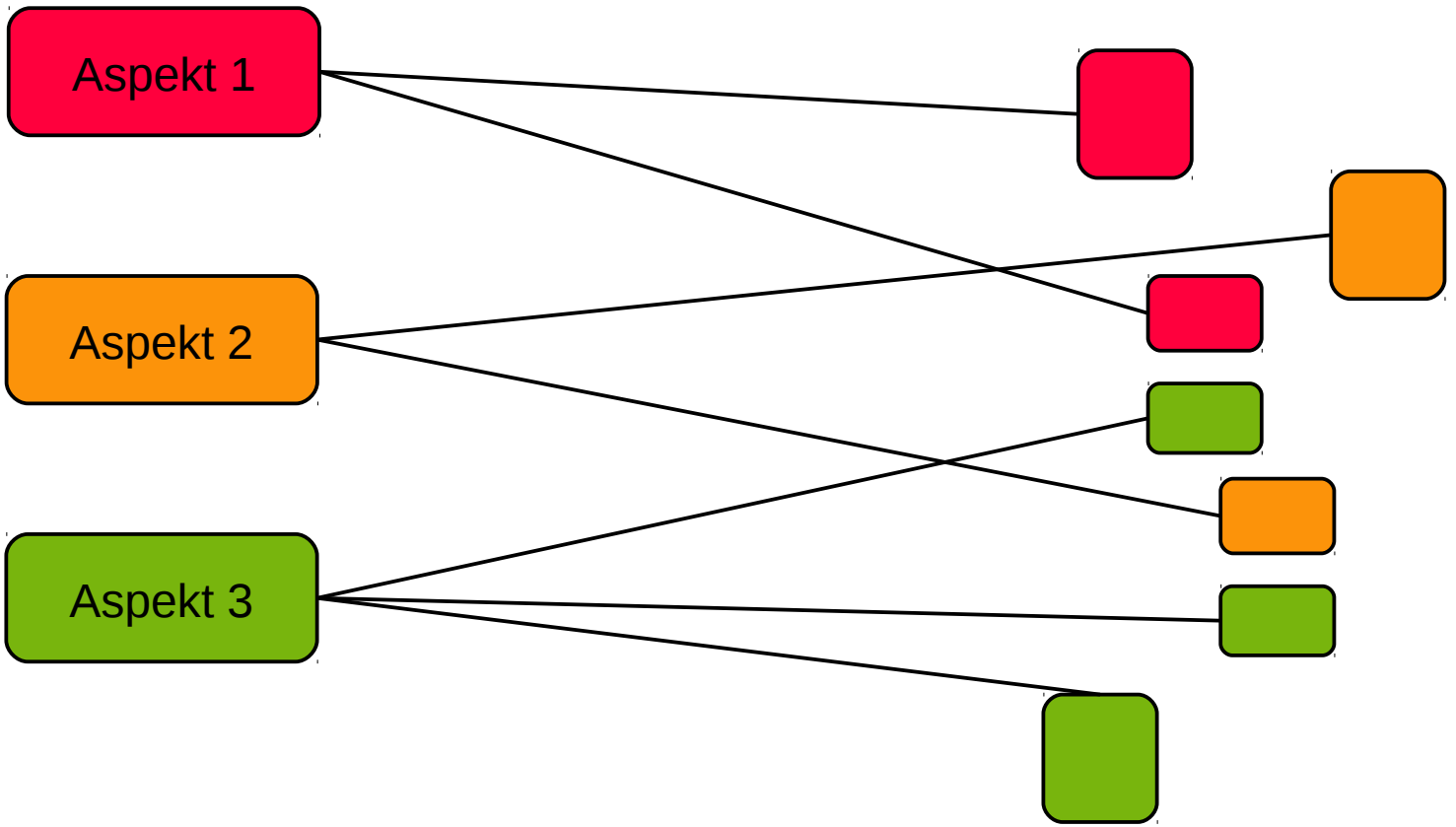
Transformation der Aspekttrennung

- Aspekttrennung ist eine wichtige Aufgabe im Entwurf: Transformiere die Einheiten des Lösungsraumes derart, dass jede Einheit nur zu einem Aspekt (Farbe) korreliert ist
  - Spalte Einheiten ggf. Auf
- In einem **aspektseparierten Komponentenraum** trägt jede Komponente nur eine Farbe



Aspektseparierte Komponentenräume sind modular

- In einem spektseparierten Komponentenraum bilden alle Komponenten der gleichen Farbe eine **Scheibe (slice, view)**, eine Art Modul



Variabilitätsgesetz der aspektseparierten Systeme

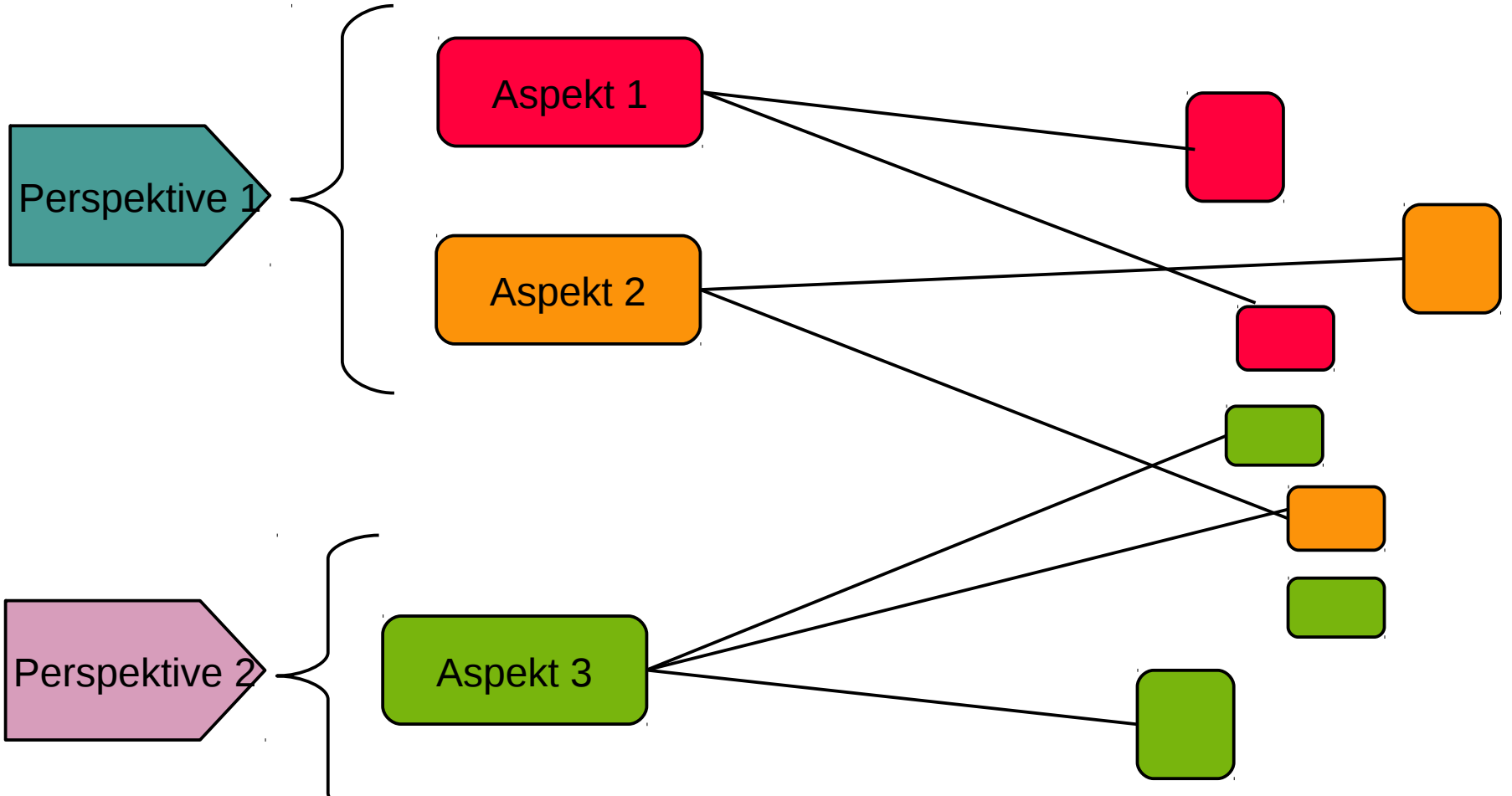
Ist ein System aspektsepariert, können die zu einem Aspekt korrelierten Komponenten des Lösungsraumes leicht ausgetauscht werden

- Austausch von Implementierungen
- Austausch von Fragmenten
- Austausch von Modellelementen

Ist ein System nicht aspektsepariert, können die zu einem Aspekt korrelierten Komponenten des Lösungsraumes nicht unabhängig von anderen Aspekten behandelt werden

Perspektiven (Viewpoints)

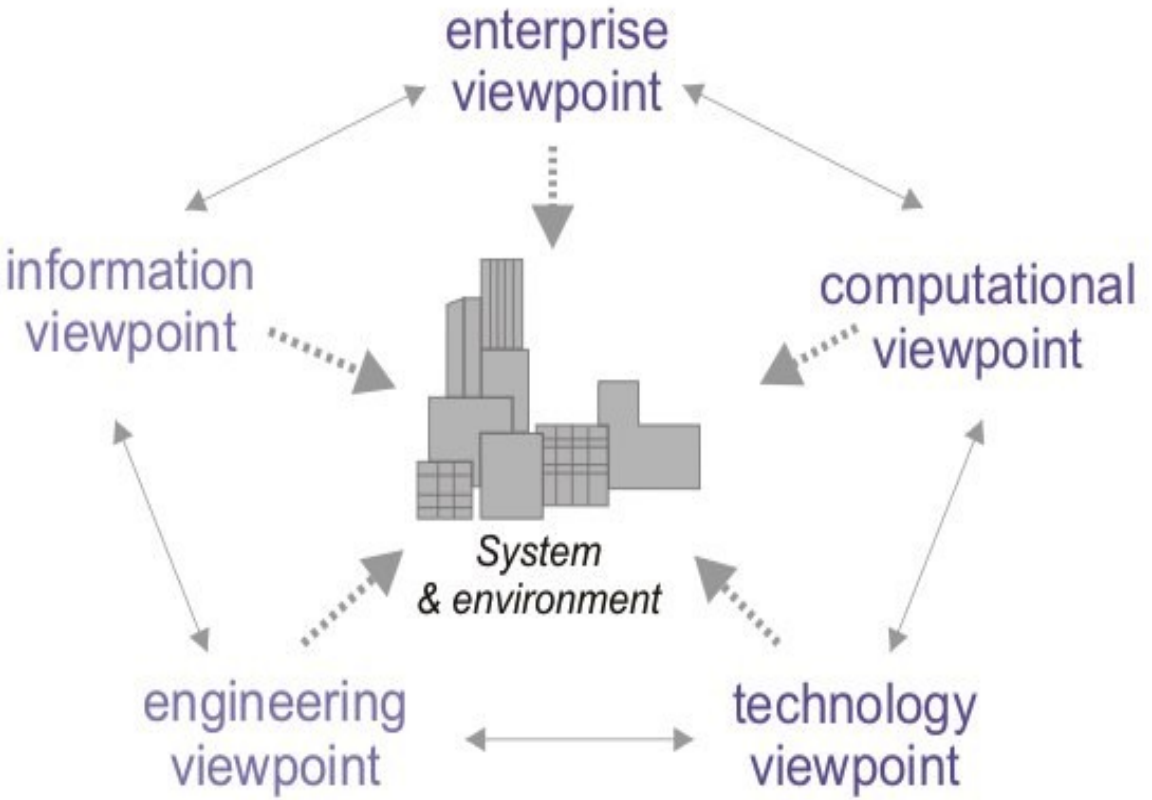
- Eine **Perspektive** besteht aus einer Menge von Aspekten (Belangen)
- Eine **Sicht** aus einer Perspektive besteht aus allen Sichten des Systems, die den Aspekten der Perspektive zugeordnet sind
- Eine **einfache Perspektive** besteht aus einem Aspekt



Perspektivenmodelle (Viewpoint Models)

- Ein **Perspektivenmodell** legt 5 Perspektiven mit ihren Aspekten und Sichten auf das System fest.
- Beispiel von RM-ODP:

http://en.wikipedia.org/wiki/View_model

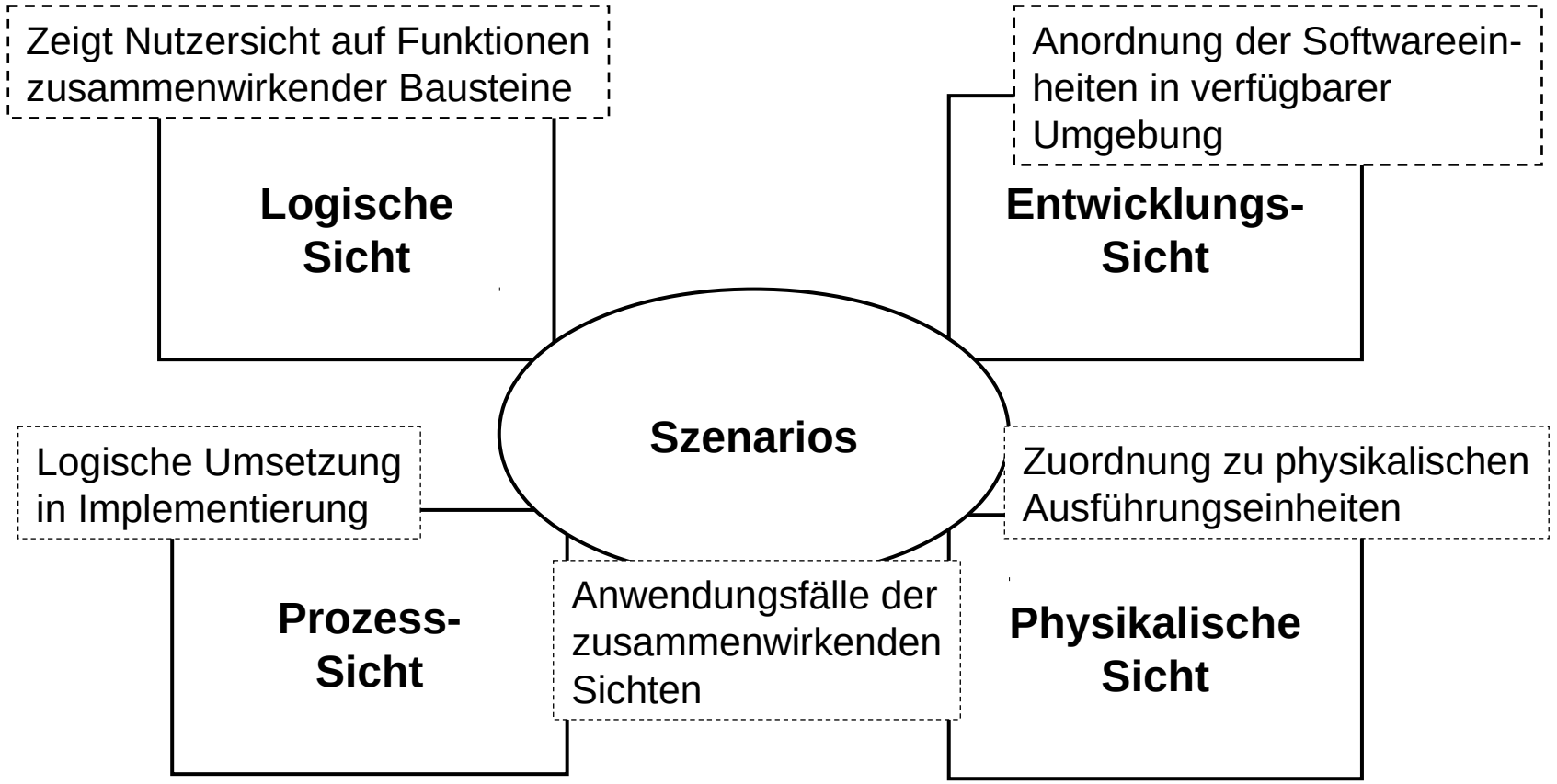


http://en.wikipedia.org/wiki/File:RM-ODP_viewpoints.jpg

Architektursichten

nach dem Perspektivenmodell „4+1 View Model of Architecture”

- Das 4+1 Modell definiert 5 Aspekte für alle Komponenten eines Systems, sowohl des Entwurfs, als auch Programms, als auch des Laufzeitsystems



Quelle: Kruchten, P., Vancouver, B., C.: The 4+1 View Model of Architecture; IEEE Software, 12 (6), November 1995, IEEE, S. 42-50

Perspektivenmodell Zachmann Framework

- Zachmann proposed a matrix aspect space out of 5 aspects and 6 questions to group all activities of the software life cycle

	Why	How	What	Who	Where	When
Contextual	Goal List	Process List	Material List	Organisational Unit & Role List	Geographical Locations List	Event List
Conceptual	Goal Relationship	Process Model	Entity Relationship Model	Organisational Unit & Role Relationship Model	Locations Model	Event Model
Logical	Rules Diagram	Process Diagram	Data Model Diagram	Role Relationship Diagram	Locations Diagram	Event Diagram
Physical	Rules Specification	Process Function Specification	Data Entity Specification	Role Specification	Location Specification	Event Specification
Detailed	Rules Details	Process Details	Data Details	Role Details	Location Details	Event Details

http://en.wikipedia.org/wiki/Zachman_Framework

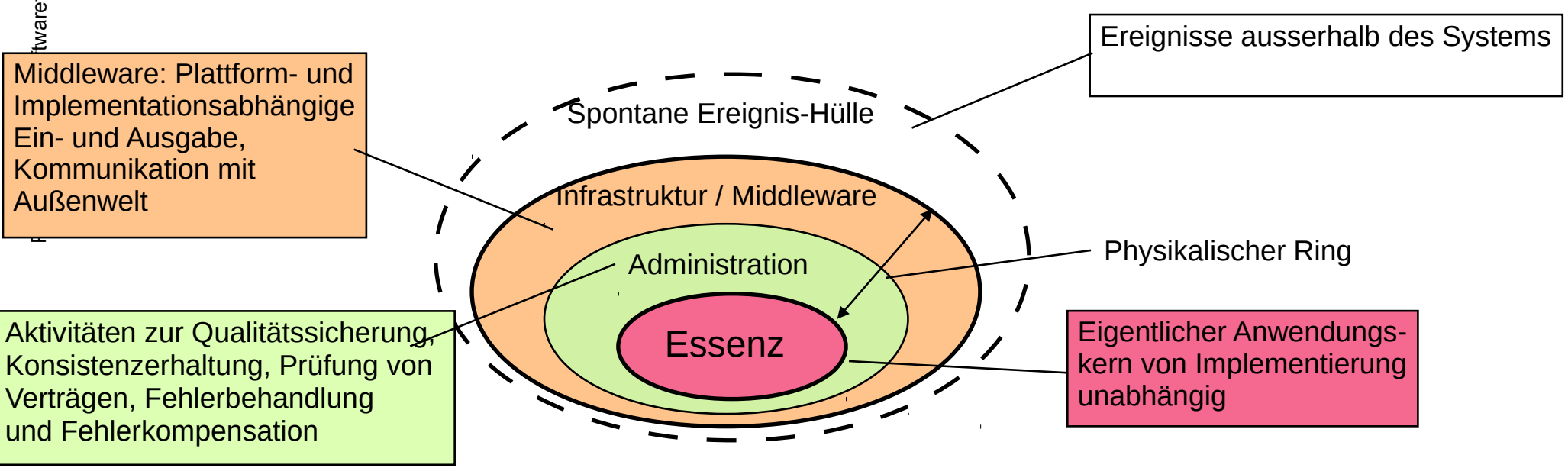
28.2 Essentielle Zerlegung in der Strukturierten Analyse

Perspektivenmodell mit den 3 Aspekten Essenz, Administration, Infrastruktur (EAI)

“The general role of middleware is to abstract away the details of underlying facilities and platforms, thereby isolating applications from them.” [Vinoski]

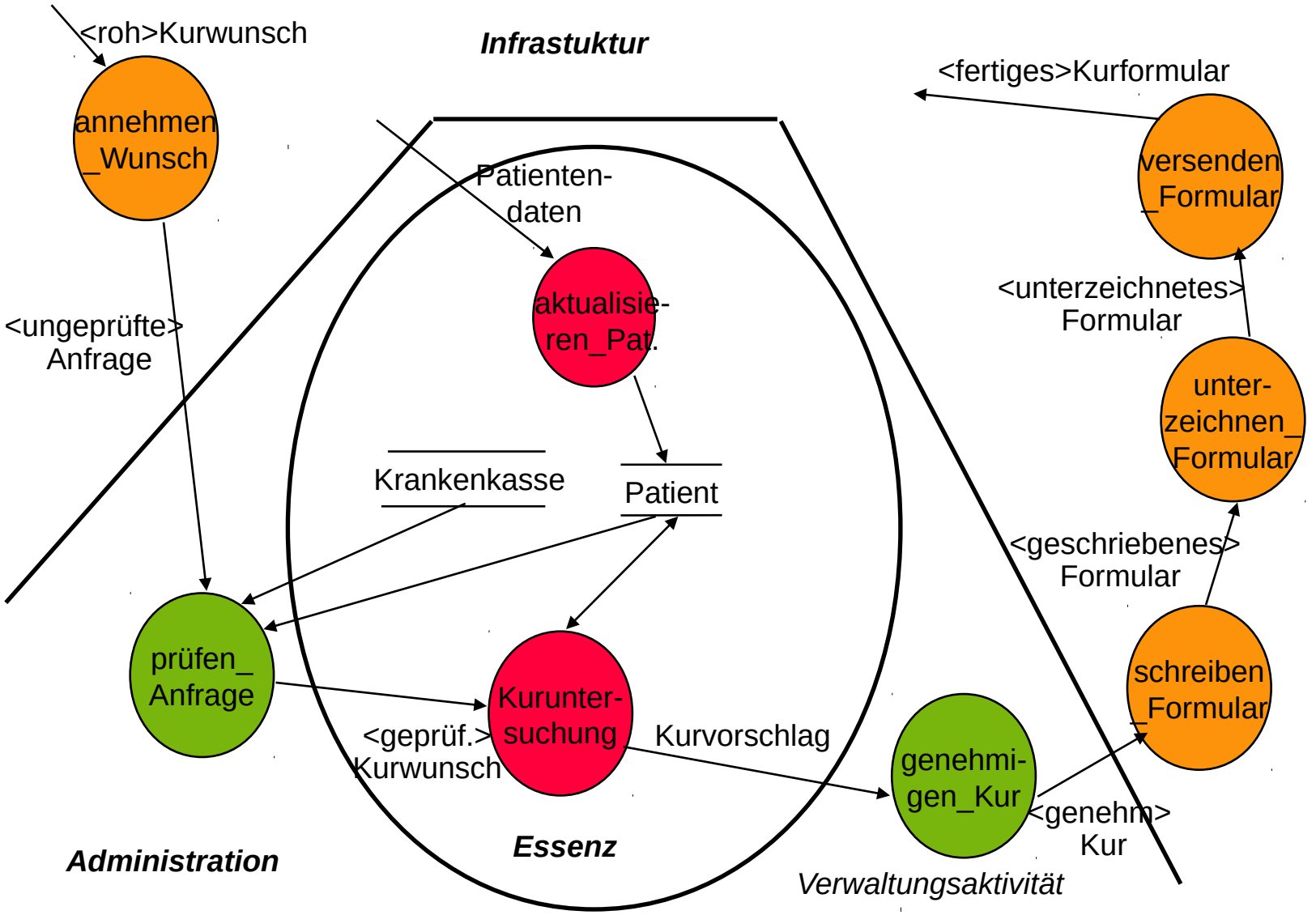
Perspektivenmodell der essentiellen Zerlegung

- **Essentielle Zerlegung** zerlegt ein System in 3 Aspekte:
  - die **Essenz** eines Systems, d.h. derjenigen Systembestandteile, die unabhängig von der Implementierung sind
    - Essenz nimmt perfekte Technologie an [McMenamen/Palmer] (Prozesse ohne Bearbeitungszeit, Speicher stets aktuell mit unendl.Kapazität)
  - die Qualitätssicherung (**Administration**).
  - die plattform-abhängigen Details (**Infrastruktur**)
- Essentielle Zerlegung bildet ein Sichtenraum mit 3 Aspekten (Essenz, Administration, Infrastruktur, EAI)



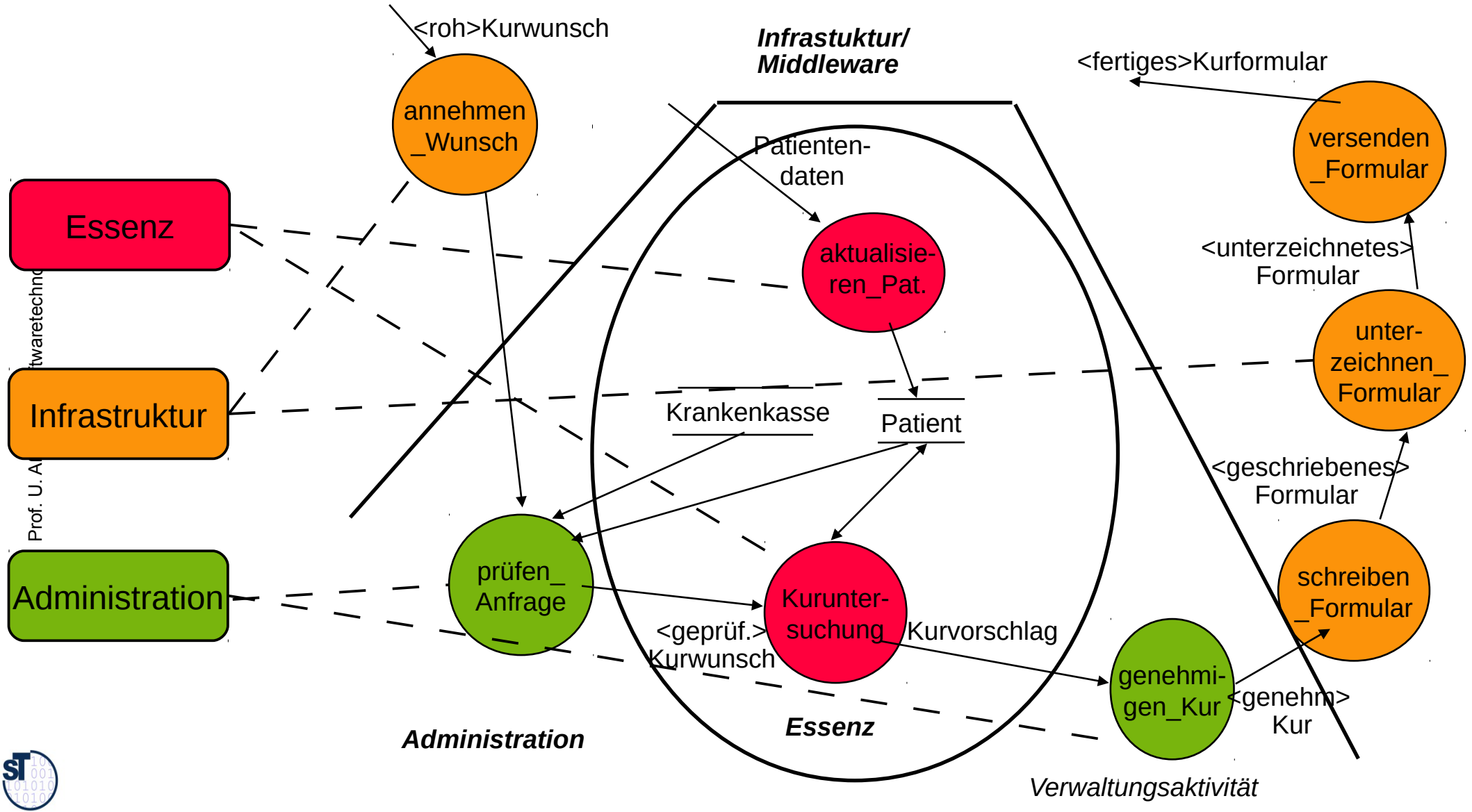
Beispiel EAI-Zerlegung eines Geschäftsprozesses “Kurantrag”

- Die EAI-Zerlegung eines Geschäftsprozesses (DFD) (Farbzerlegung mit Essenz, Administration, Infrastruktur/Middleware)
- EAI wurde in der SA erfunden, kann aber für alle Entwicklungsmethoden angewandt werden



Beispiel EAI-Zerlegung eines Geschäftsprozesses “Kurantrag”

- Der EAI-Sichtenbildungsraum für SA-DFD

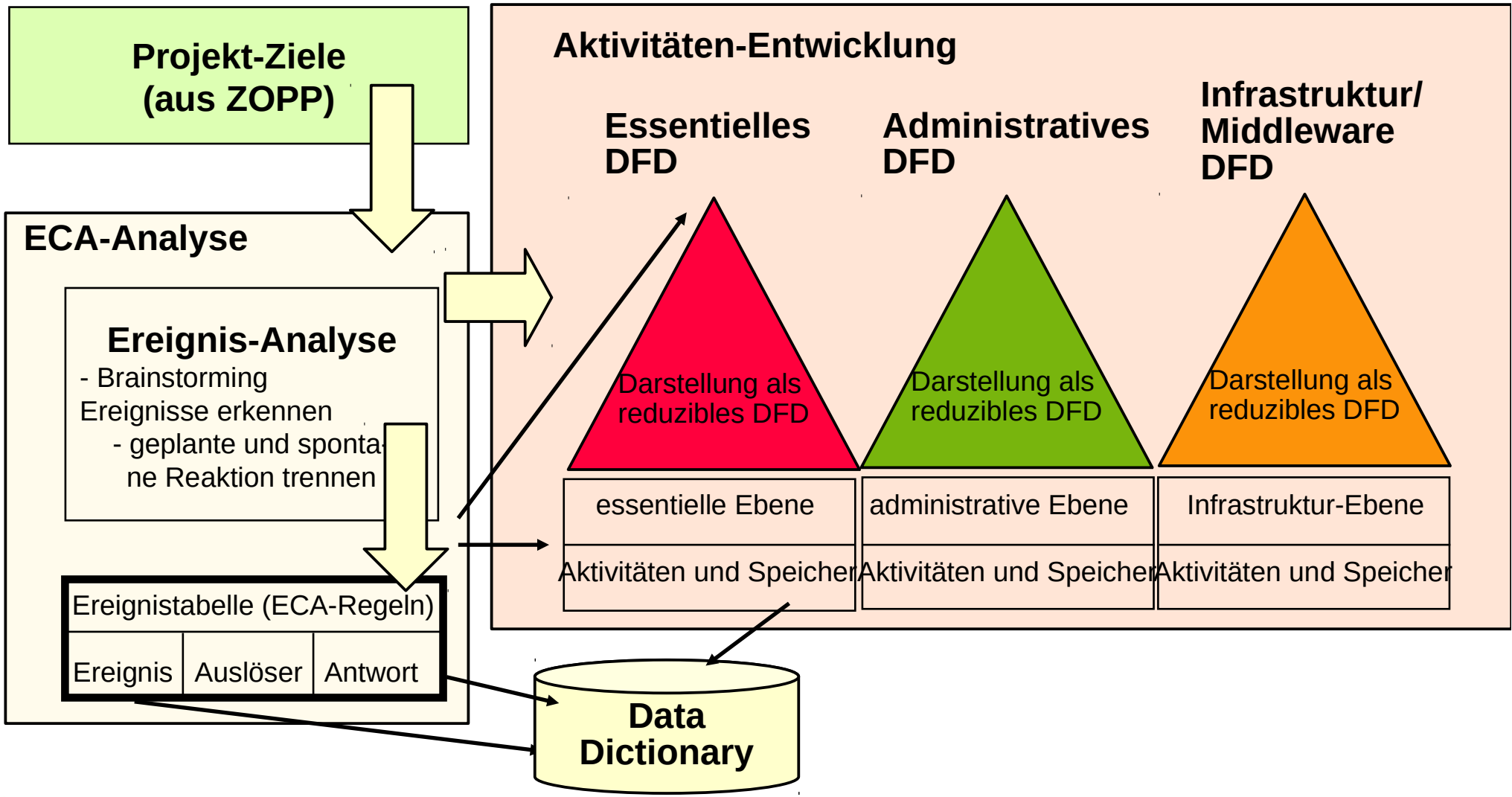


Essentielle Strukturierte Analyse (ESA)

- In DFD geht die Einteilung in Aspekte (Essenz, Administration, Infrastruktur) sehr einfach: Aspekte zerlegen das DFD in Scheiben einer Farbe (graph slicing).
  - Strombasierte DFD lassen sich leicht zerlegen
  - Aktivitätendiagramme der UML ähnlich
  - Petrinetze auch
- Für Module, COTS-Komponenten etc. geht das nicht ganz so gut, denn sie sind aufrufbasiert
- **Essentielle Strukturierte Analyse** ist ein Entwicklungsprozess, der ECA-basierten Entwurf mit EAI-Zerlegung kombiniert

Vorgehen zur essentiellen strukturierten Analyse (ESA)

- ESA combines ZOPP, ECA, EAI and ZOPP



Quelle: Raasch, J.: Systementwicklung mit Strukturierten Methoden; Hanser Verlag (2.Auf .) München 1992)

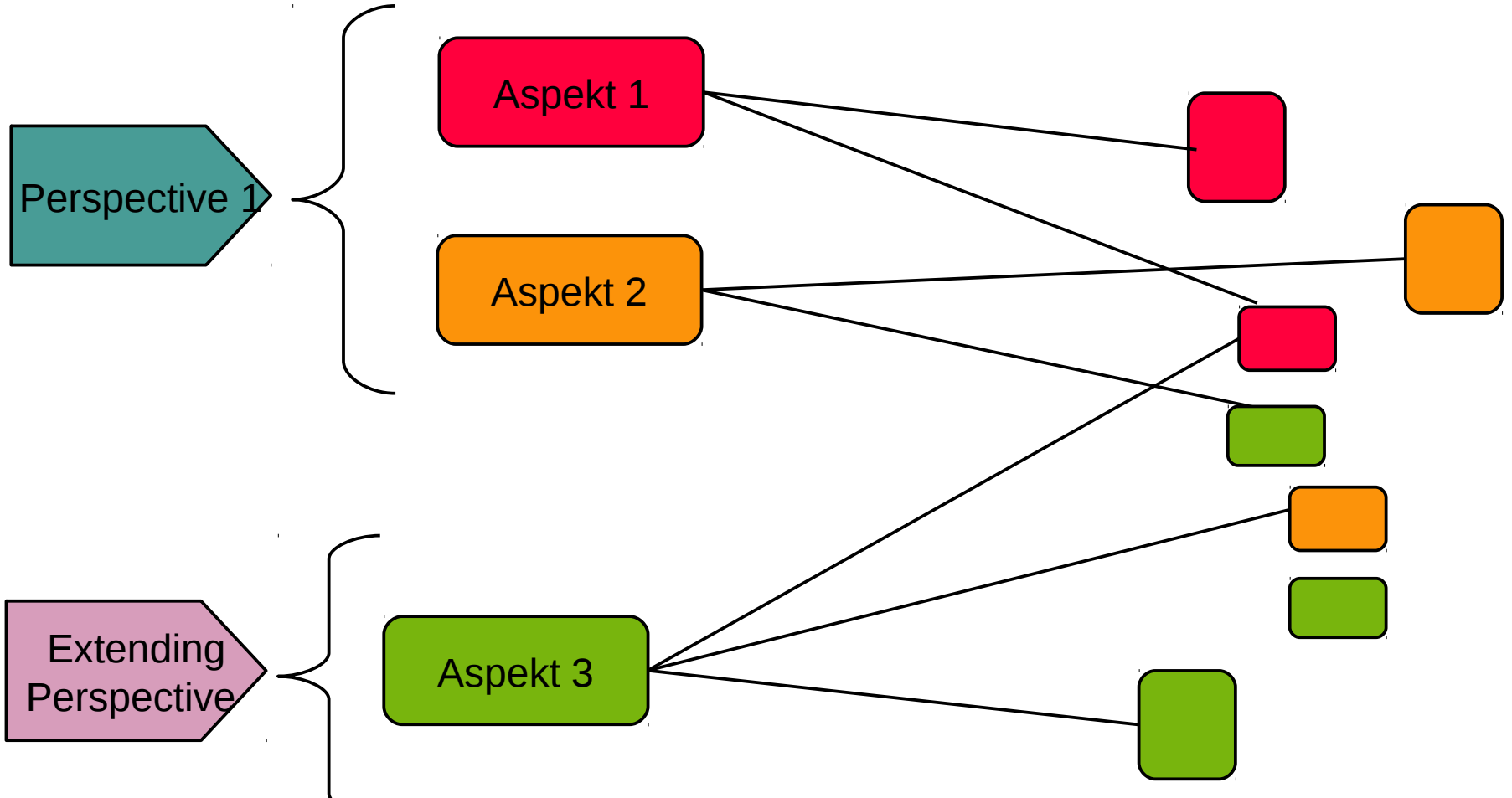
Methodik der essentiellen strukturierten Analyse (ESA)

1) **Ziele** des neuen Systems festlegen (z.B. Mit ZOPP)
2) **Essentielle Aktivitäten finden** (mit EAI)
   1) Aufstellen der Ereignisse in einer Ereignistabelle mit ECA-Regeln, die über Eingabe-Datenflüsse Ausgabe-Datenflüsse und damit einen Prozess initiieren. Aus der vollständigen Ereignistabelle ist das essentielle DFD (essentieller Aspekt) zu erstellen
   2) **Erstellen des essentiellen Kontextdiagramm**
   3) **Essentielle Speicher des Systems finden** über Analyse der Ein- und Ausgabewerte von Datenflüssen
   4) **Verfeinerung des Kontextdiagramms**
3) **Administrations-(Verwaltungs-)aktivitäten finden**, die zur Erstellung, Konsistenzsicherung und Pflege essentieller Speicher (Laden, Änderung, Löschen,...) notwendig sind
4) **Infrastrukturaktivitäten finden** (Kommunikation, Datenaustausch, Datenwandlung)
5) Vorläufiges integriertes Modell aus allen 3 Aspekten erstellen
6) Wiederholung und Verfeinerung bis zu den MiniSpecs

28.3. Extension by Aspect-Orientation

Aspect-Oriented Extensions

- With Aspect-Orientations and an appropriate extension mechanism, software can be extended also in unforeseen ways
- Extension works by adding a new aspect with a new slice to the system



The End – Was haben wir gelernt?

- Wie dekomponiert eine Aspektzerlegung eine Software in mehrere Belange?
  - Was ist ein “Concern space” (Belang-Raum)?
- Welche Bestandteile enthält ein Perspektivenmodell?
- Beispiele?
  - EAI-Zerlegung
  - RM-ODP, Zachmann, Kruchten 4+1
  - Blutgruppen von Siedersleben (siehe ST-1)
  - Werkzeug-Effektkategorien (siehe SEW)