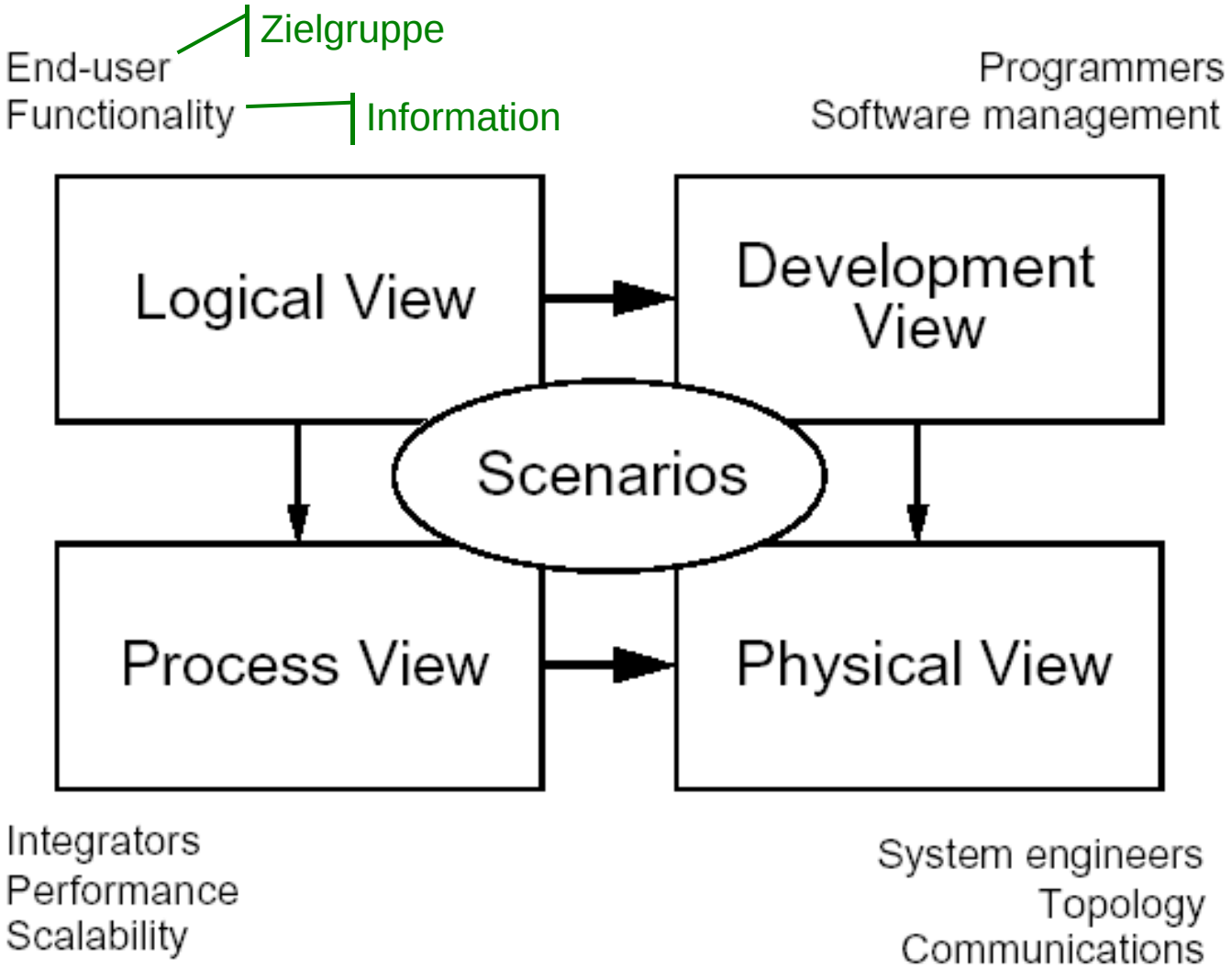
The 4+1 View Model of Architecture [Kruchten]

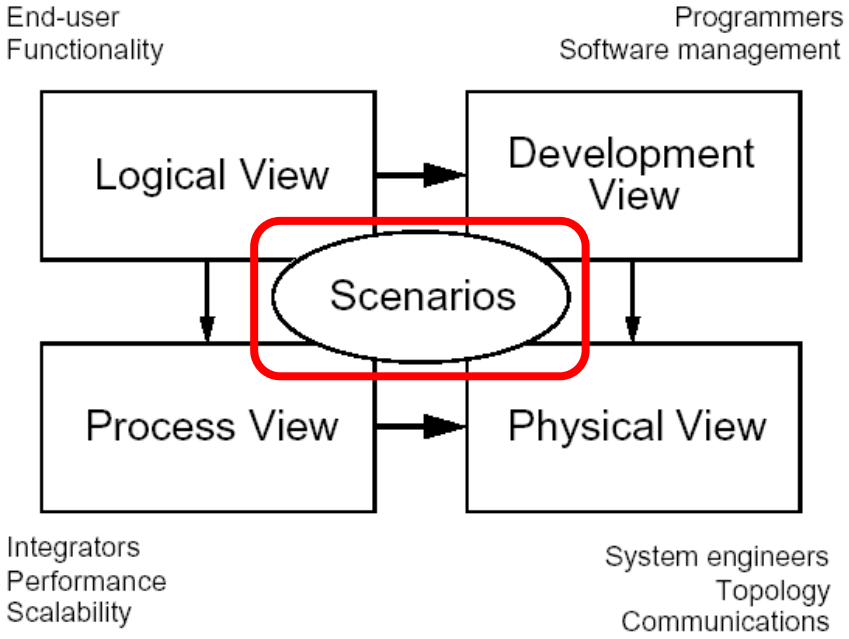
Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...



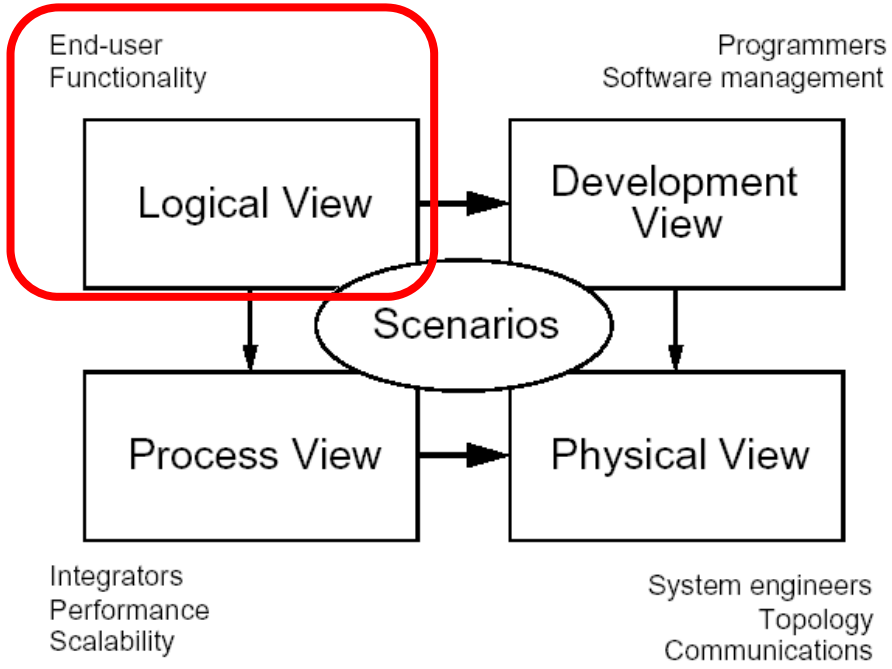
4 + 1 Views

- Jede Sicht behandelt unterschiedliche Aspekte (*Concerns*), die für unterschiedliche Zielgruppen relevant sind
- Die 4 *Views* werden mit Szenarien / Use Cases illustriert: 5. *View*
  - Repräsentation wichtiger Anforderungen
  - Szenarien (= konkrete Ausprägungen von Use Cases)
  - "Treiber" für den Architektur-Entwurf
  - Illustration der Architektur
- Jede Sicht enthält
  - Elemente
  - Pattern etc.
  - Begründungen (*rationales*) und Einschränkungen (*constraints*)



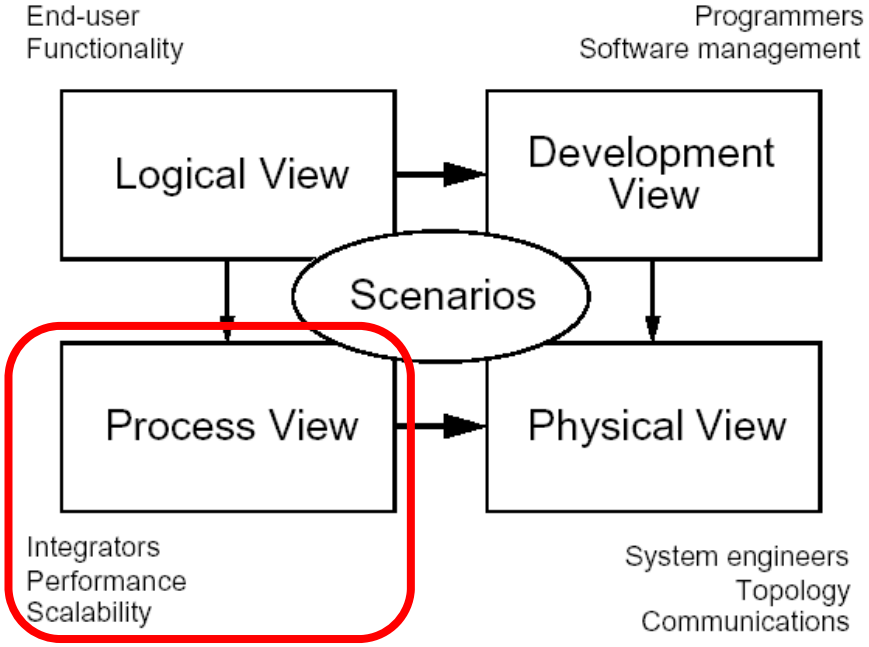
4 + 1 Views

- Logische Sicht
  - Zielgruppe: End-User
  - Betrifft funktionale Anforderungen und deren Gruppierung / Aufteilung
  - Elemente/Darstellung: objektorientiert, Klassendiagramme (urspr. Booch-Notation)



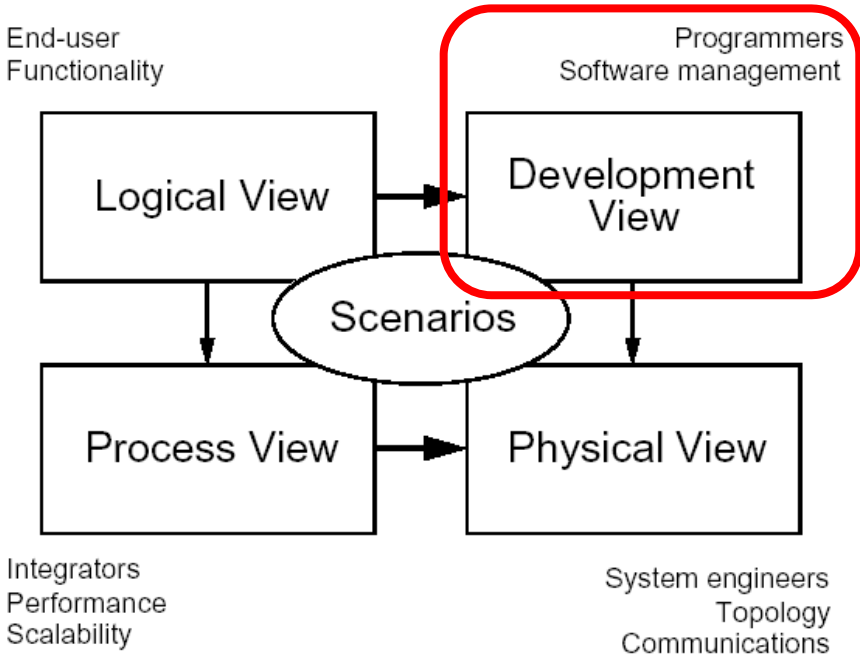
4 + 1 Views

- Prozess-Sicht
  - Zielgruppe: Software-Integratoren
  - Betrifft nicht-funktionale / Qualitäts-Anforderungen: Parallelität, Verteilung, Synchronisation, Performance, Skalierbarkeit
  - Elemente/Darstellung: Prozesse, Nachrichten etc. (urspr. keine formal festgelegte Notation)



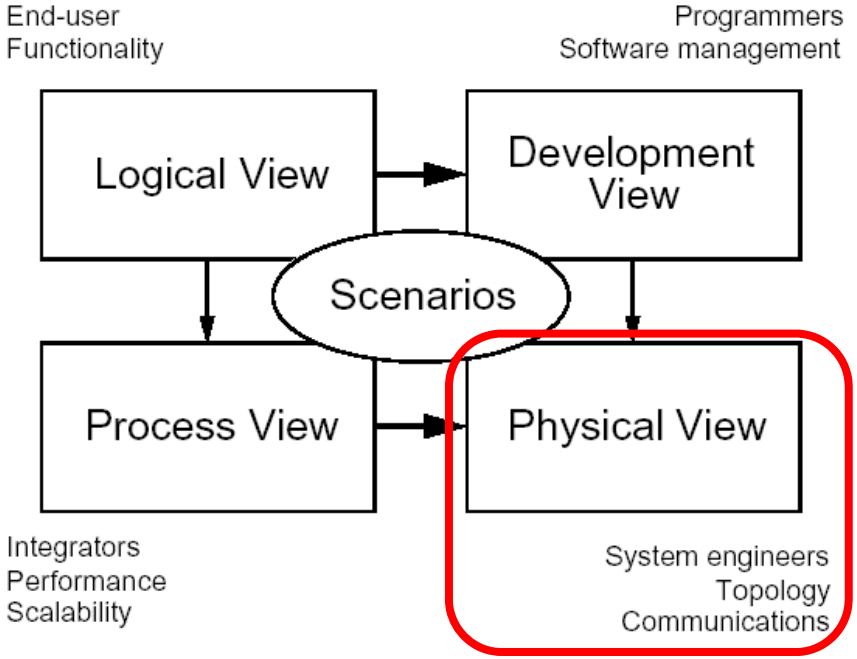
4 + 1 Views

- Entwickler-Sicht (traditionelles Design-Dokument)
  - Zielgruppe: Entwickler und Software-Manager
  - Betrifft Datei-Organisation, Produkt-Organisation, Planung
  - Elemente/Darstellung: Komponenten, Subsysteme, Layer; Klassen, Export- und Import-Beziehungen (urspr. keine formal festgelegte Notation)



4 + 1 Views

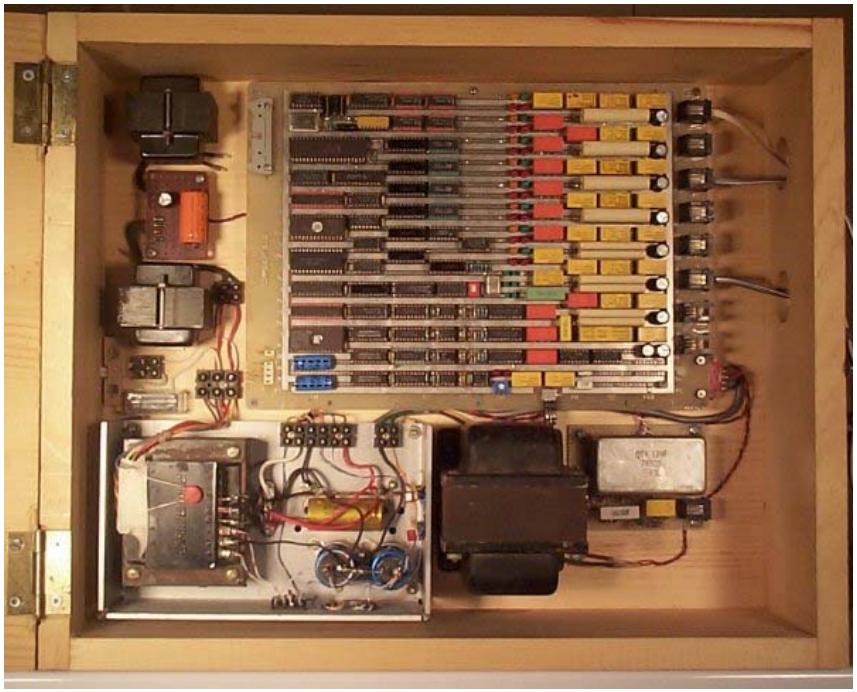
- Physikalische Sicht
  - Zielgruppe: *System*-Ingenieure
  - Betrifft nicht-funktionale / Qualitäts-Anforderungen: Verfügbarkeit, Zuverlässigkeit, Performance, Skalierbarkeit
  - Elemente/Darstellung: einfache Diagramme (Prozessoren, gerichtete Kommunikation) (urspr. keine formal festgelegte Notation)



Nochmal in der Übersicht: *4 + 1 Views*

- Logische Sicht
  - Zielgruppe: End-User
  - Betrifft: Funktionale Anforderungen und deren Gruppierung / Aufteilung
  - Elemente/Darstellung: objektorientiert, Klassendiagramme (urspr. Booch-Notation)
- Prozess-Sicht
  - Zielgruppe: Software-Integratoren
  - Betrifft: Nicht-funktionale Anforderungen (Parallelität, Verteilung, Synchronisation, Performance, Skalierbarkeit)
  - Elemente/Darstellung: Prozesse, Nachrichten etc.
- Entwickler-Sicht (traditionelles Design-Dokument)
  - Zielgruppe: Entwickler und Software-Manager
  - Betrifft: Datei-Organisation, Projekt-Organisation/-Planung
  - Elemente/Darstellung: Komponenten, Subsysteme, Layer; Export- und Import-Beziehungen
- Physikalische Sicht
  - Zielgruppe: *System*-Ingenieure
  - Betrifft: Nicht-funktionale Anforderungen (Verfügbarkeit, Zuverlässigkeit, Performance, Skalierbarkeit)
  - Elemente/Darstellung: einfache Diagramme (Prozessoren, gerichtete Kommunikation)

Beispiel [Kruchten]: PABX



PABX – *Private Automatic Branch Exchange*
Ein privater Telefonverteiler, der ortspezifische Dienste und einen Zugang zu einem lokalen Kommunikationsnetz zur Verfügung stellt.

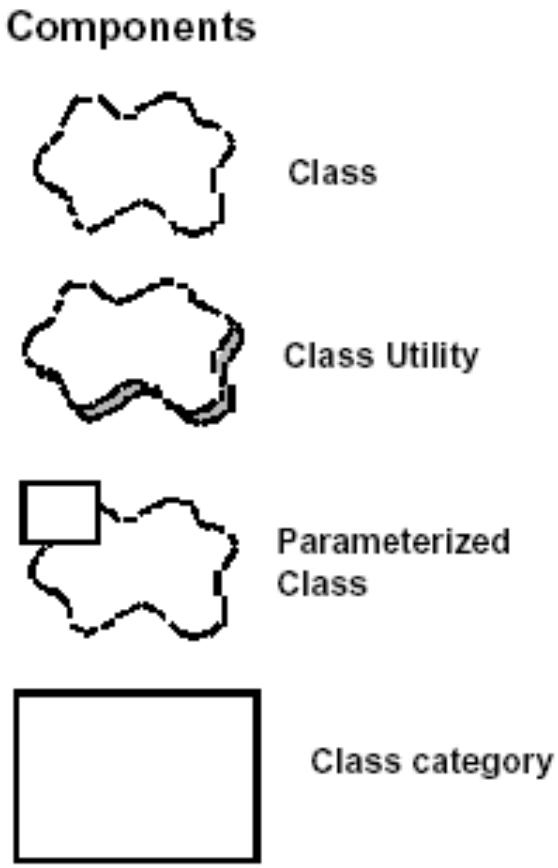
Sieht heute anders aus... ☺

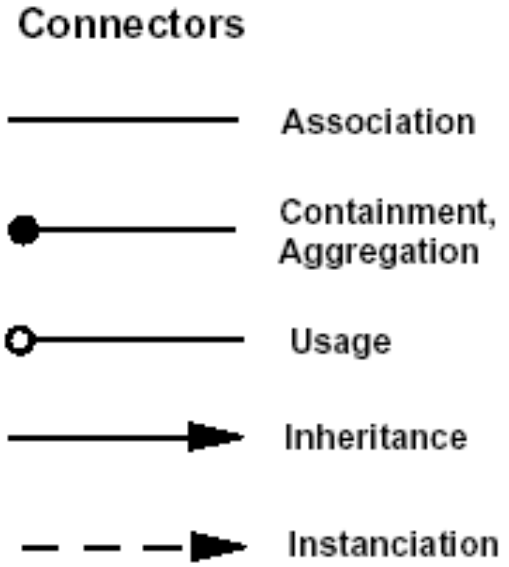
Notation [Booch] (Paper von 1995)

Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...

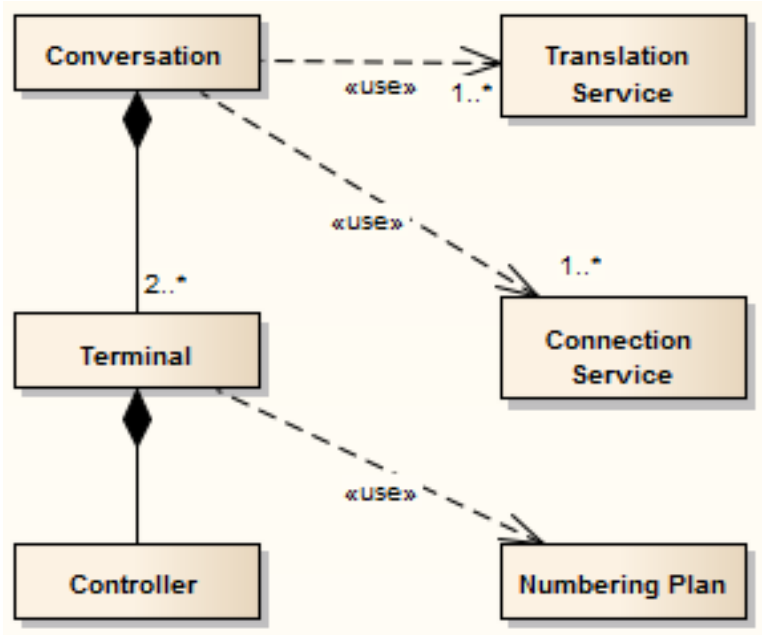
Animation





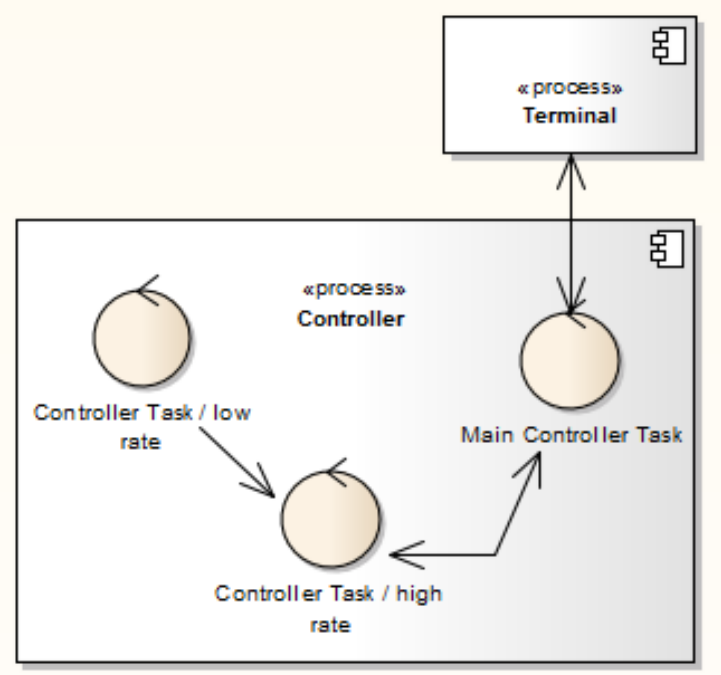
Beispiel PABX: Logische Sicht

- A PABX establishes communications between *terminals*
  - Examples for terminals: data line, ISDN line, PABX to PABX line
  - A *controller* handles one interface card for one kind of line
  - A *numbering plan* maps digits to lines
- A *conversation* represents a set of terminals engaged in a conversation, it uses
  - *connection services* to establish a voice path between the terminals and
  - *translation services* (directory, logical to physical address mapping, routes)



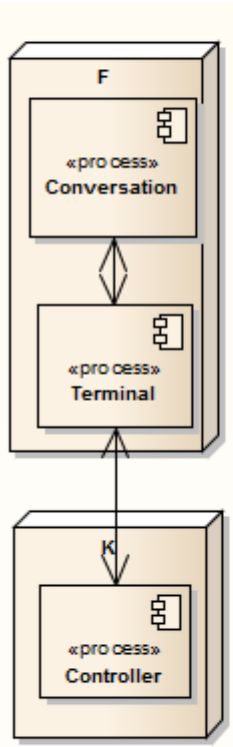
Beispiel PABX: Prozess-Sicht

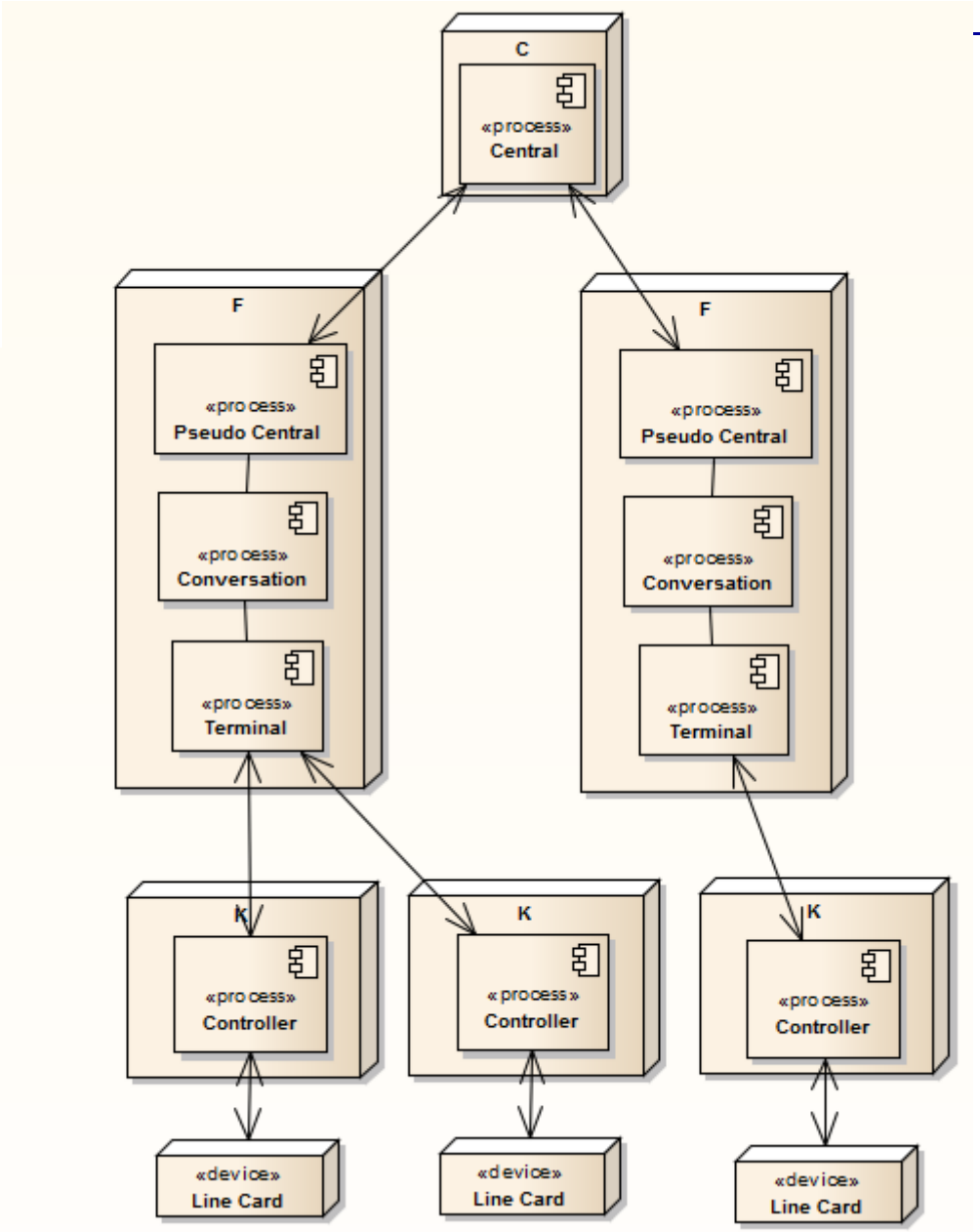
- All terminals are handled by a single *terminal process*, which is driven by messages in its input queues
- A *low cycle rate task* (200 ms) scans all inactive terminals, puts any terminal becoming active in the scan list of a high cycle rate task
- The *high cycle rate task* (10ms), which detects any significant change of state, passes them to the main controller task
- The *main controller task* interprets the changes and communicates them by message to the corresponding terminal
- The controller objects are executed in *one controller process*; message passing within the controller process is done via shared memory



Beispiel PABX: Physikalische Sicht kombiniert mit Prozess-Sicht

Two mappings of the process architecture on two different physical architectures, corresponding to a large and a small PABX





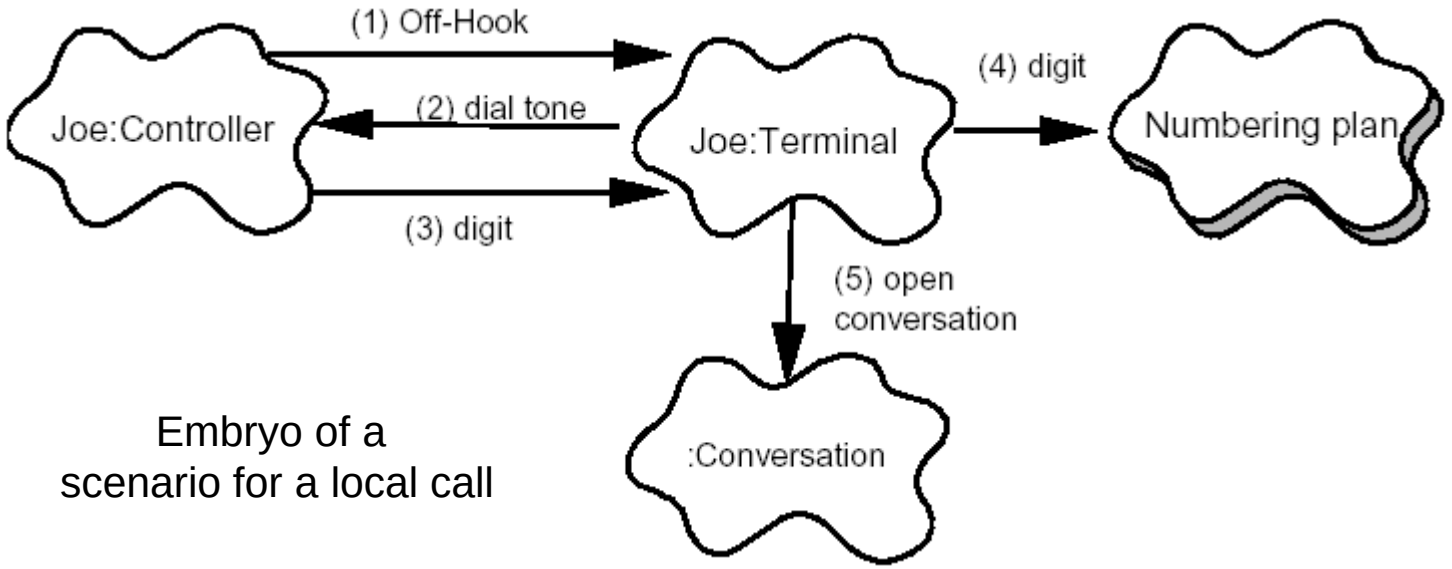
Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...

Leider enthält das Papier kein Beispiel für die Entwicklersicht...

Solche Sichten kennen Sie aber!

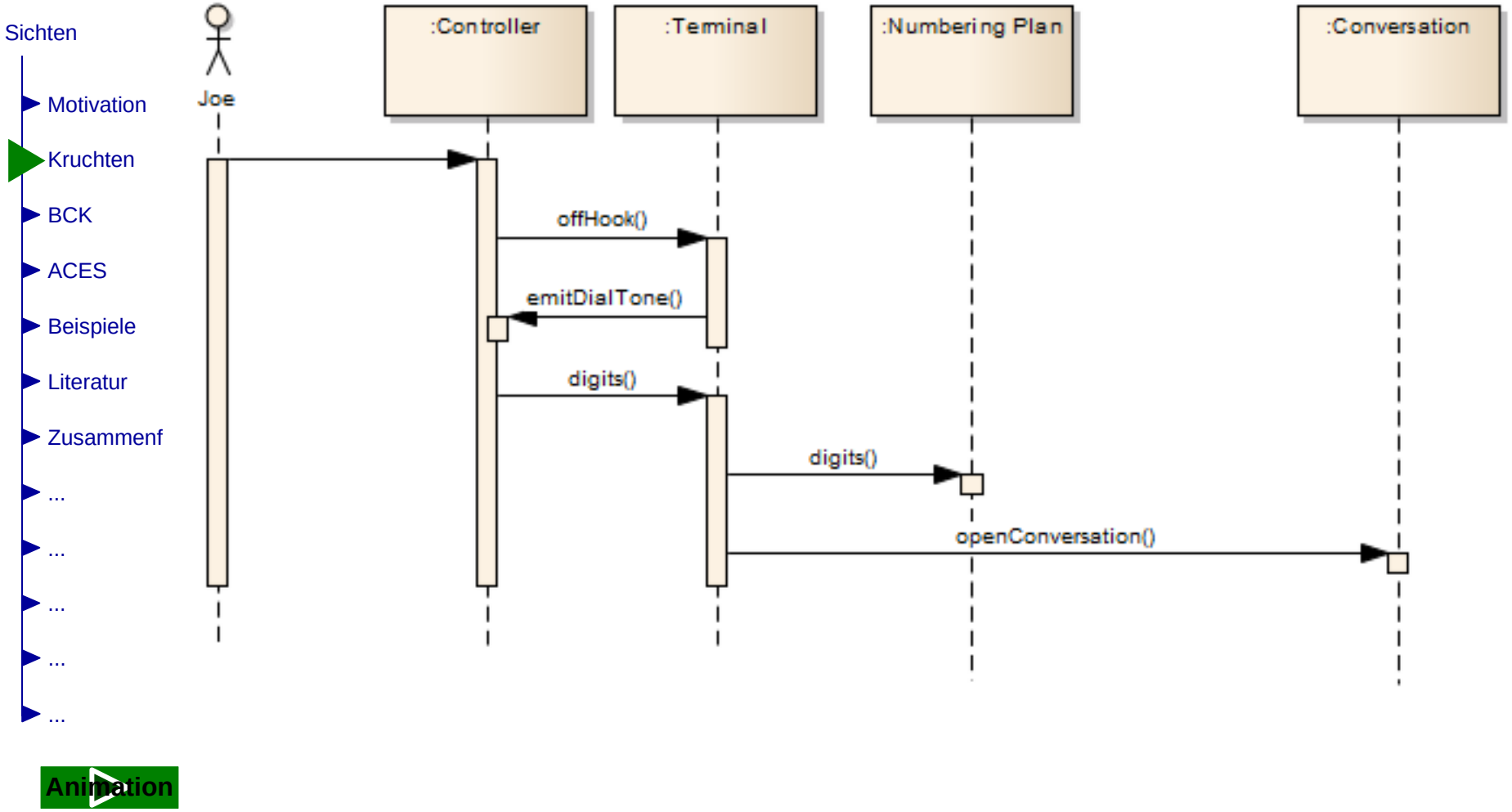
Example Scenario: the "... + 1" View



Embryo of a scenario for a local call

1. The controller of Joe's phone detects the transition from on-hook to off-hook and sends a message to wake up the corresponding terminal object
2. The terminal tells the controller to emit some dial-tone
3. The controller receives encoded digits and transmits them to the terminal
4. The terminal uses the numbering plan to analyze the digit flow.
5. When a valid sequence of digits has been entered, the terminal opens a conversation

Example Scenario: the "... + 1" View

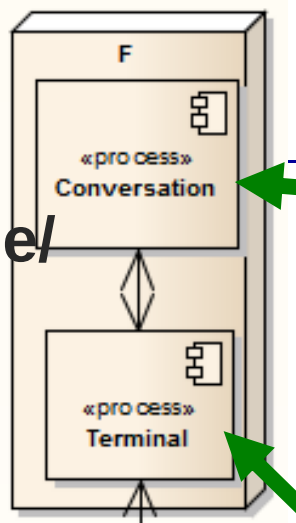


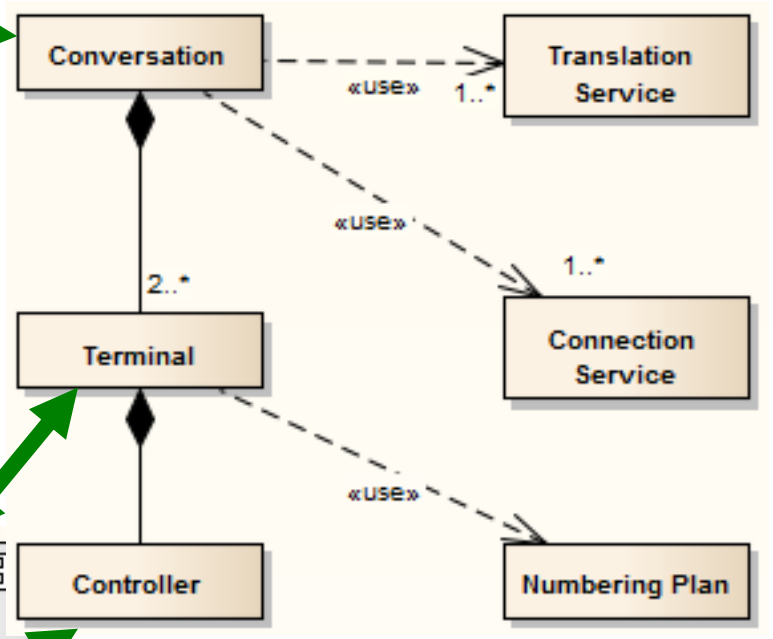
Abhängigkeiten im PABX-Beispiel

Sichten

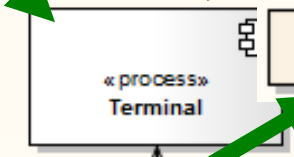
- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...

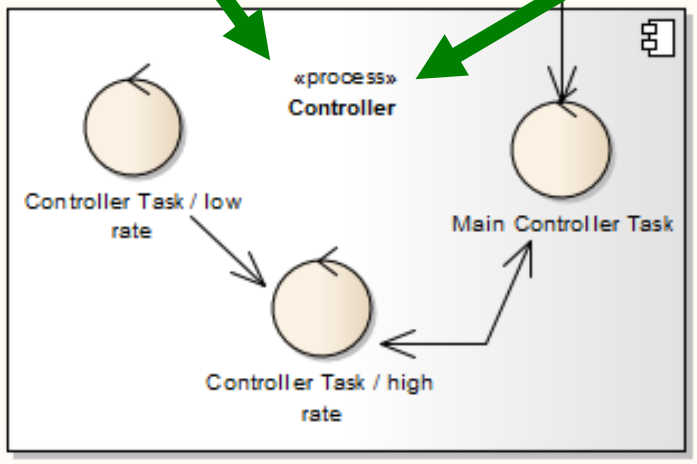
Physikalische/ Prozess Sicht





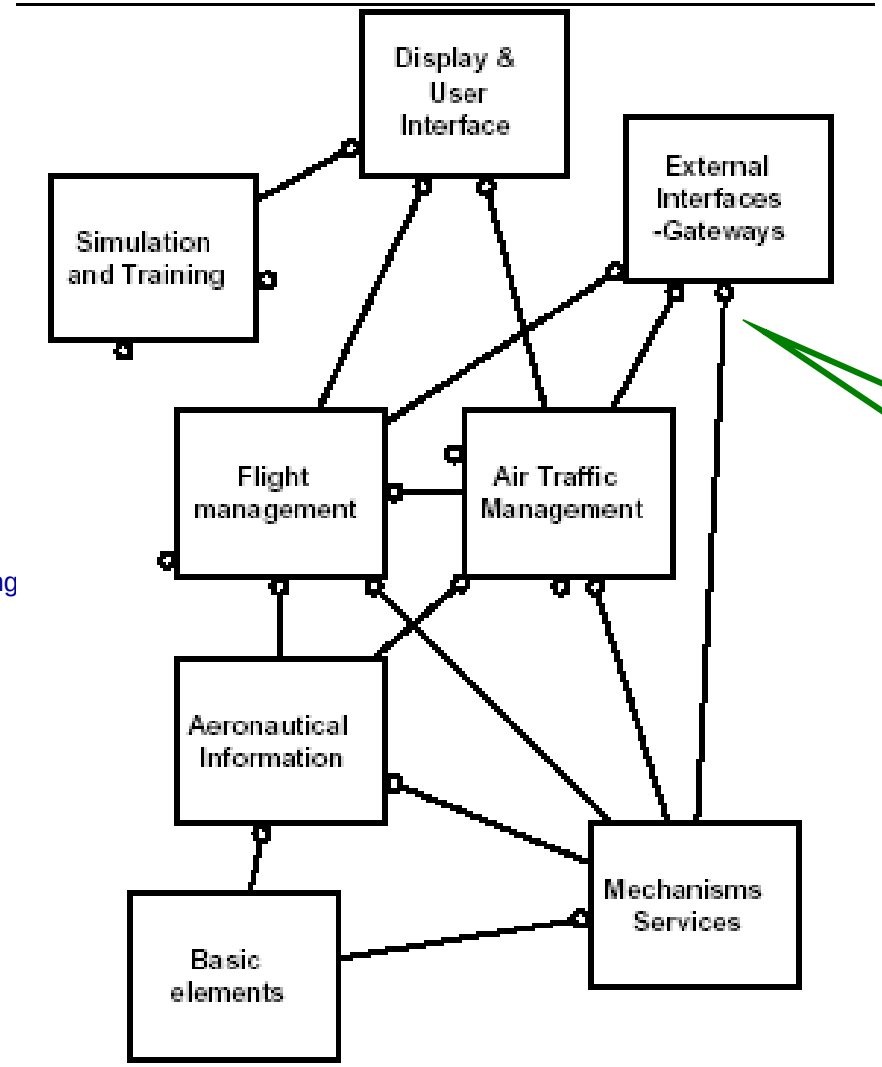
Logische Sicht





Prozess-Sicht

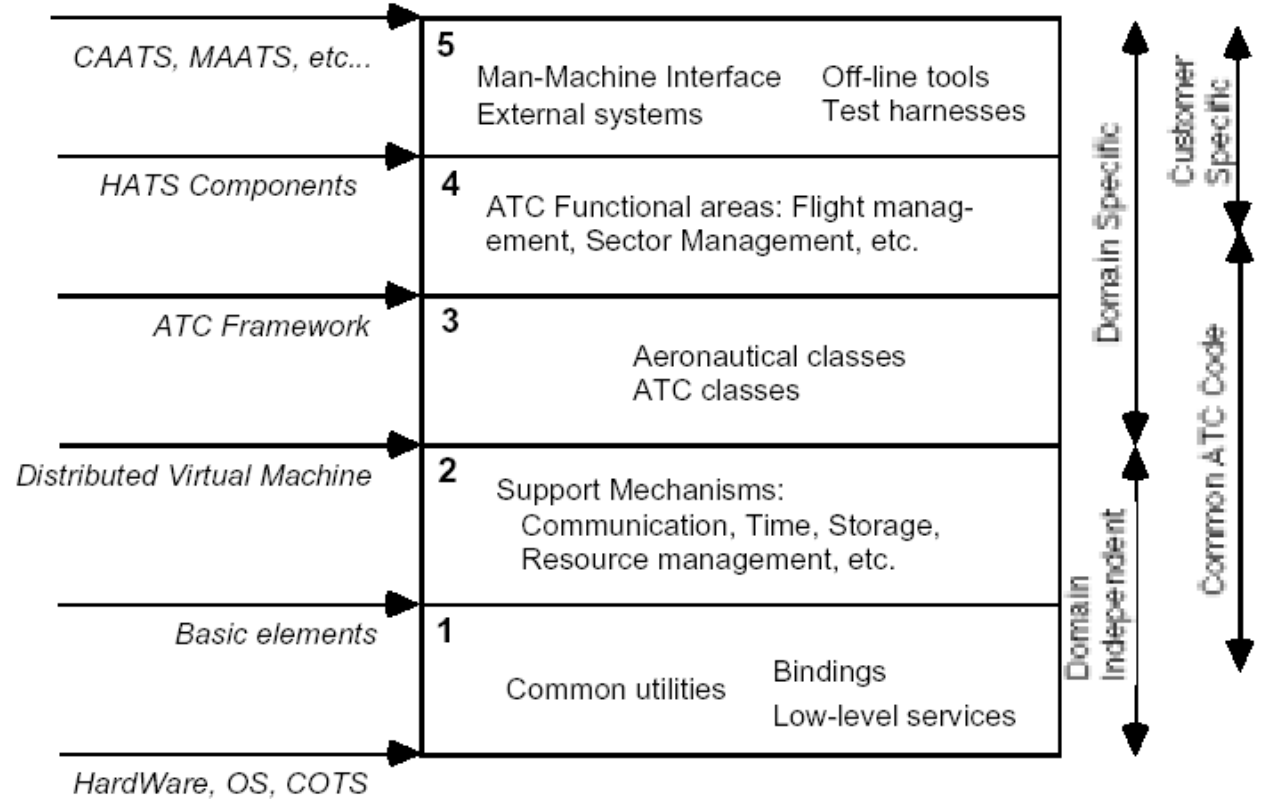
Another Example: Air Traffic Control System, Logical View



- Top level class diagram containing 8 class categories
- Classes of architectural significance

"Usage" relation

Air Traffic Control System: Development View



Product line of Air Traffic Control systems developed by Hughes Aircraft of Canada

Organization in five layers

- Layers 1 and 2: domain-independent, encapsulates hardware, OS, DBMS etc.
- Layer 3: domain-specific ATC framework
- Layer 4: functionality based on layer 3
- Layer 5: customer-, product-dependent, user interfaces, interfaces to external systems

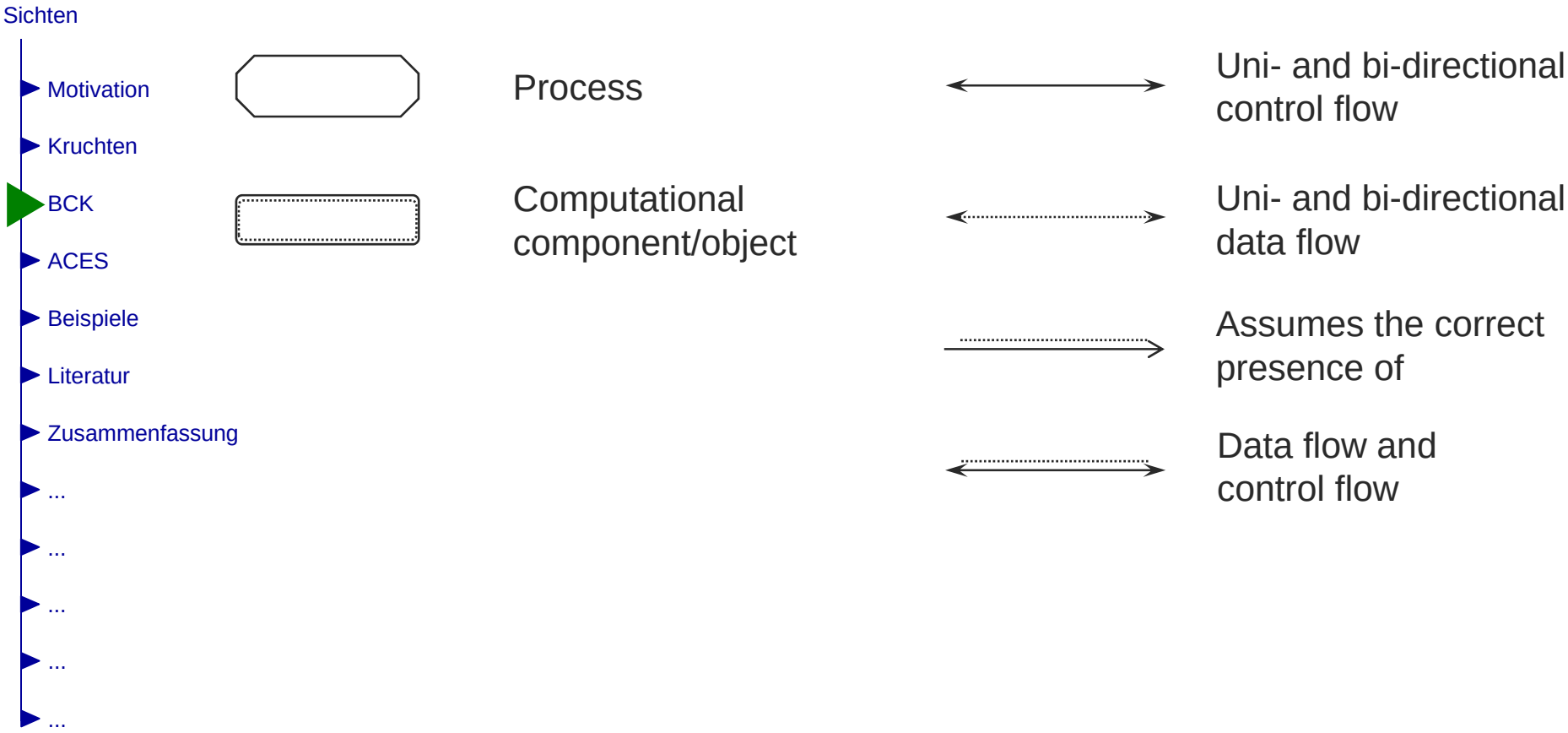
Architectural Structures [Bass, Clements, Kazman]

Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...

- Module
  → Modifizierbarkeit, Entwicklung von Teilsystemen
- Process
  → Parallelität, Verteilung, Synchronisation, Performance, Skalierbarkeit
- Uses
  → Entwicklung von Teilsystemen
- Calls
  → Beobachtung des Kontrollflusses (Debugging!), Testbarkeit, Wartbarkeit
- Data flow
  → Verteilung der Funktionalität, Performance, Korrektheit, Genauigkeit
- Class
  → Projektplanung, Wartbarkeit
- Physical
  → Verfügbarkeit, Zuverlässigkeit, Performance

Notation:
Komponenten und Verbindungen bei BCK (Auszug)



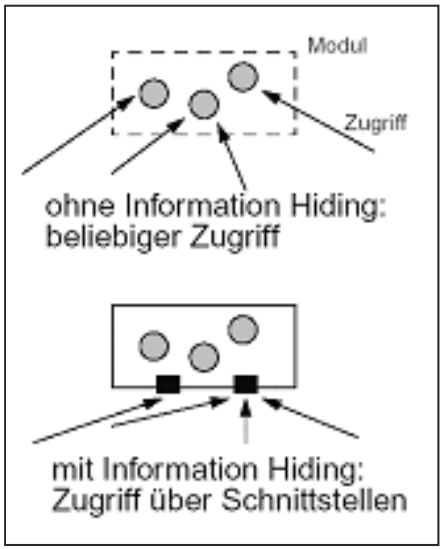
Beispiel: A-7E Corsair II Aircraft

Hintergrund, Anforderungen 1/2

- Ein kleiner, langsamer Bordcomputer für Navigation und Waffen-Zielführung
- 32 KByte Speicher, Assembler, optimiert für Performance
- Berechnungen müssen 25 Mal/s laufen
- Extreme Modifizierbarkeit: Beispiel für *Information Hiding* (Parnas, 1977)

Begriff: *Information Hiding* [Parnas, 1972]

We have tried to demonstrate by these examples that it is almost always incorrect to begin the decomposition of a system into modules on the basis of a flowchart. We propose instead that one begins with a list of difficult design decisions or design decisions which are likely to change. Each module is then designed to hide such a decision from the others. Since, in most cases, design decisions transcend time of execution, modules will not correspond to steps in the processing. To achieve an efficient implementation we must abandon the assumption that a module is one or more sub-routines, and instead allow subroutines and programs to be assembled collections of code from various modules.

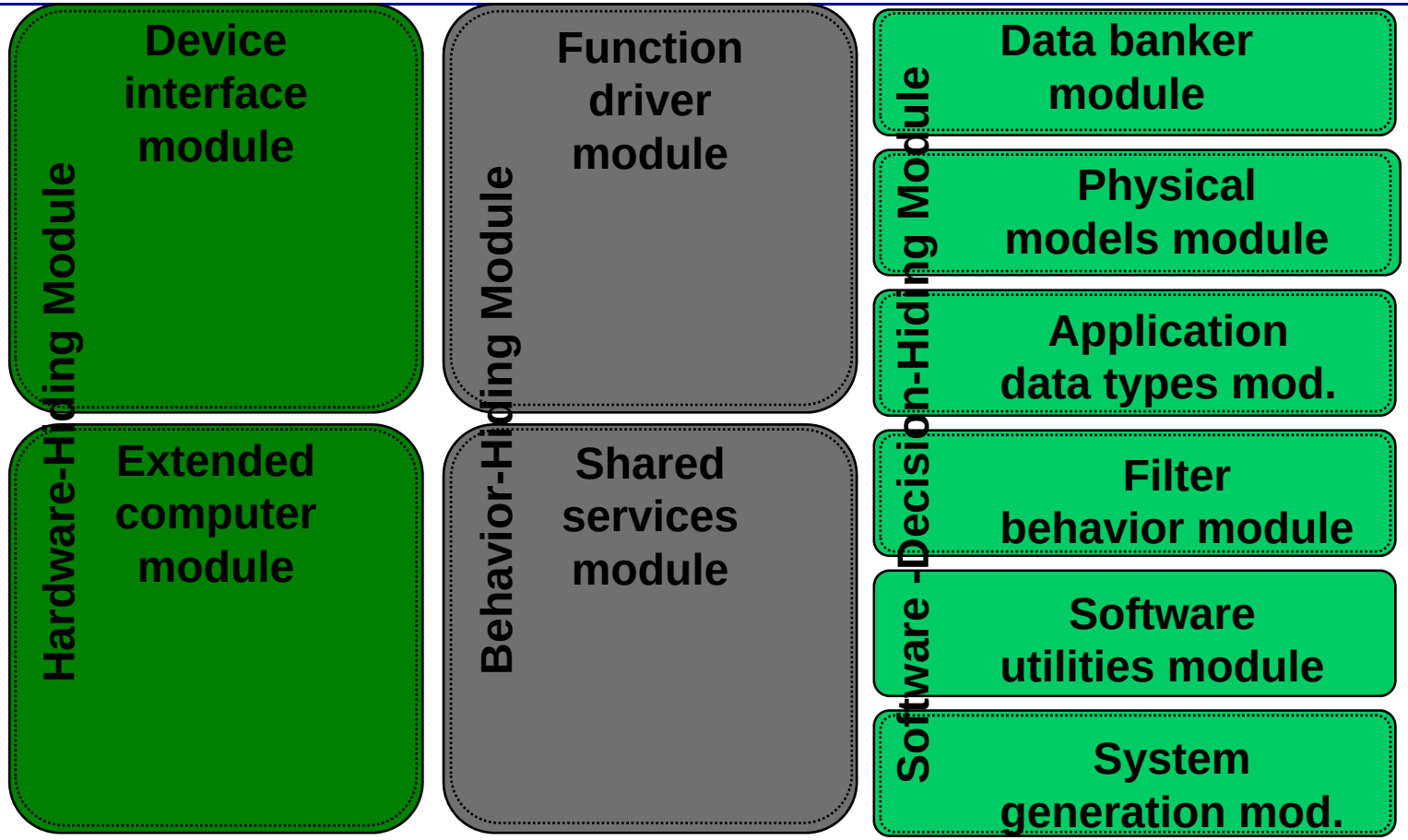


http://www.software-kompetenz.org/

Hintergrund, Anforderungen 2/2

- Konzentration auf 3 Sichten
  - Modul-Sicht → Modifizierbarkeit, Teilsysteme ausliefern, Expertenzuordnung
  - Uses-Sicht → Teilsysteme ausliefern
  - Prozess-Sicht: → Portierbarkeit, Performance Tuning
- *Information Hiding* als übergeordnetes Prinzip (→ Modul-Sicht)
  - Bereiche potenzieller Änderungen Modulen zuordnen
  - Sich ändernde Aspekte in der Modul-Implementierung kapseln
  - Konstante Aspekte als Modul-Schnittstellen anbieten
  - Zugriffe auf Module sind ausschließlich über die Schnittstellen erlaubt
  - Datenstrukturen, Algorithmen und andere änderbare Aspekte verbergen
- 3 Arten von Änderungen
  - Hardware: Sensoren/Aktoren, Computer
  - Verhalten: Funktionalität
  - Software: Prozess-Scheduling, Datentypen, Daten-Aktualisierung

Beispiel: Module Structure 1/2
A-7E Avionics System



- Komponenten: Module, Arbeitspaket-Zuordnungen
- Beziehungen: “is a submodule of,” “shares a secret with”
- Verwendung: Team-Strukturierung, Ressourcen-Planung
- Beeinflusste Qualitätsattribute: Wartbarkeit, Verständlichkeit

Hardware-Hiding Module

- **Extended Computer Module**
  - Data Module
  - Input/Output Module
  - Computer State Module
  - Parallelism Control Module
  - Program Module
  - Virtual Memory Module
  - Interrupt Handler Module
  - Timer Module
- **Device Interface Module**
  - Air Data Computer Module
  - Angle of Attack Sensor Module
  - Audible Signal Device Module
  - Computer Fail Device Module
  - Doppler Radar Set Module
  - Flight Information Displays Module
  - Forward Looking Radar Module
  - Head-Up Display Module
  - Inertial Measurement Set Module
  - Input/Out Representation Module
  - Master Function Switch Module
  - Panel Module
  - Projected Map Display Set Module
  - Radar Altimeter Module
  - Shipboard Inertial Navigation System Module
  - Slew Control Module
  - Switch Bank Module
  - TACAN Module
  - Visual Indicators Module
  - Waypoint Information Systems Module
  - Weapon Characteristics Module
  - Weapon Release System Module
  - Weight on Gear Module

Behavior-Hiding Module

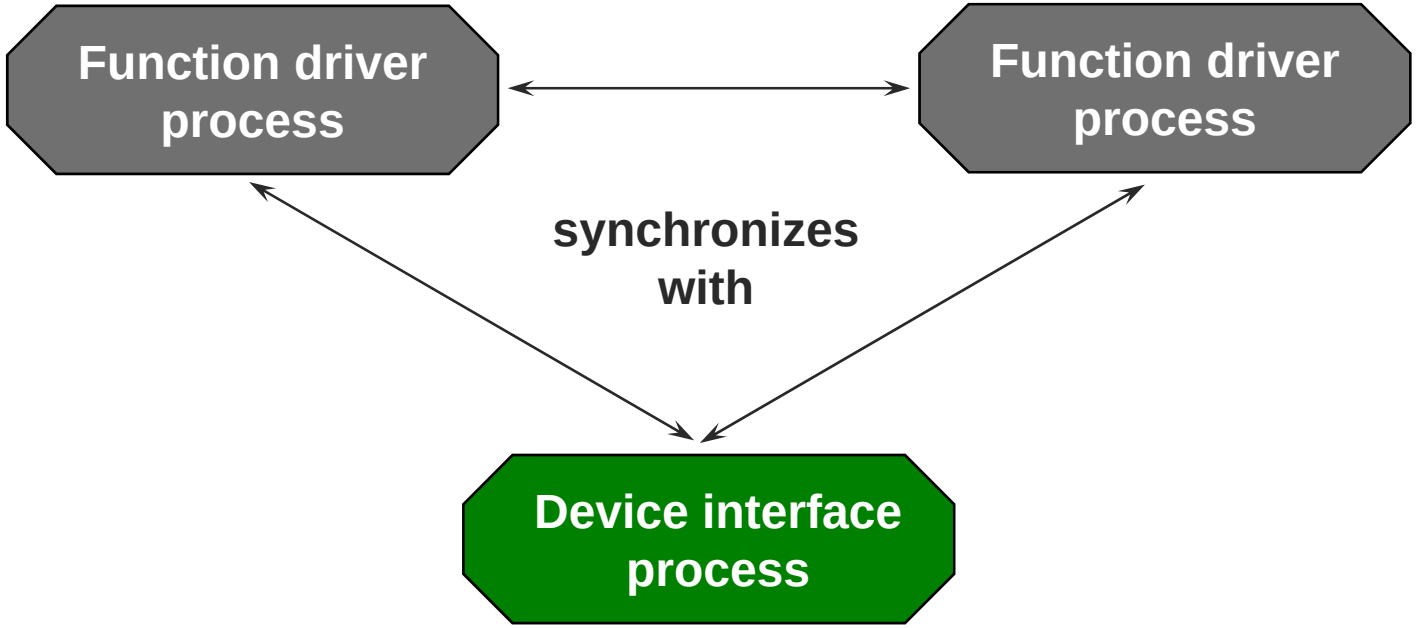
- **Function Driving Module**
  - Air Data Computer Module
  - Audible Signal Module
  - Computer Fail Signal Module
  - Doppler Radar Set Module
  - Flight Information Display Module
  - Forward Looking Radar Module
  - Head-Up Display Module
  - Inertial Measurement Set Module
  - Panel Module
  - Projected Map Display Set Module
  - Shipboard Inertial Navigation System Module
  - Visual Indicator Module
  - Weapon Release System Module
  - Ground Test Module
- **Shared Services Module**
  - Mode Determination Module
  - Panel I/O Support Module
  - Shared Subroutine Module
  - Stage Director Module
  - System Value Module

Software Decision Module

- **Application Data Type Module**
  - Numeric Data Type Module
  - State Transition Event Module
- **Data Banker Module**
  - Singular Values Module
  - Complex Event Module
- **Filter Behavior Module**
- **Physical Models Module**
  - Aircraft Motion Module
  - Earth Characteristics Module
  - Human Factors Module
  - Target Behavior Module
  - Weapon Behavior Module
- **Software Utility Module**
  - Power-Up Initialization Module
  - Numerical Algorithms Module
- **System Generation Module**
  - System Generation Parameter Module
  - Support Software Module

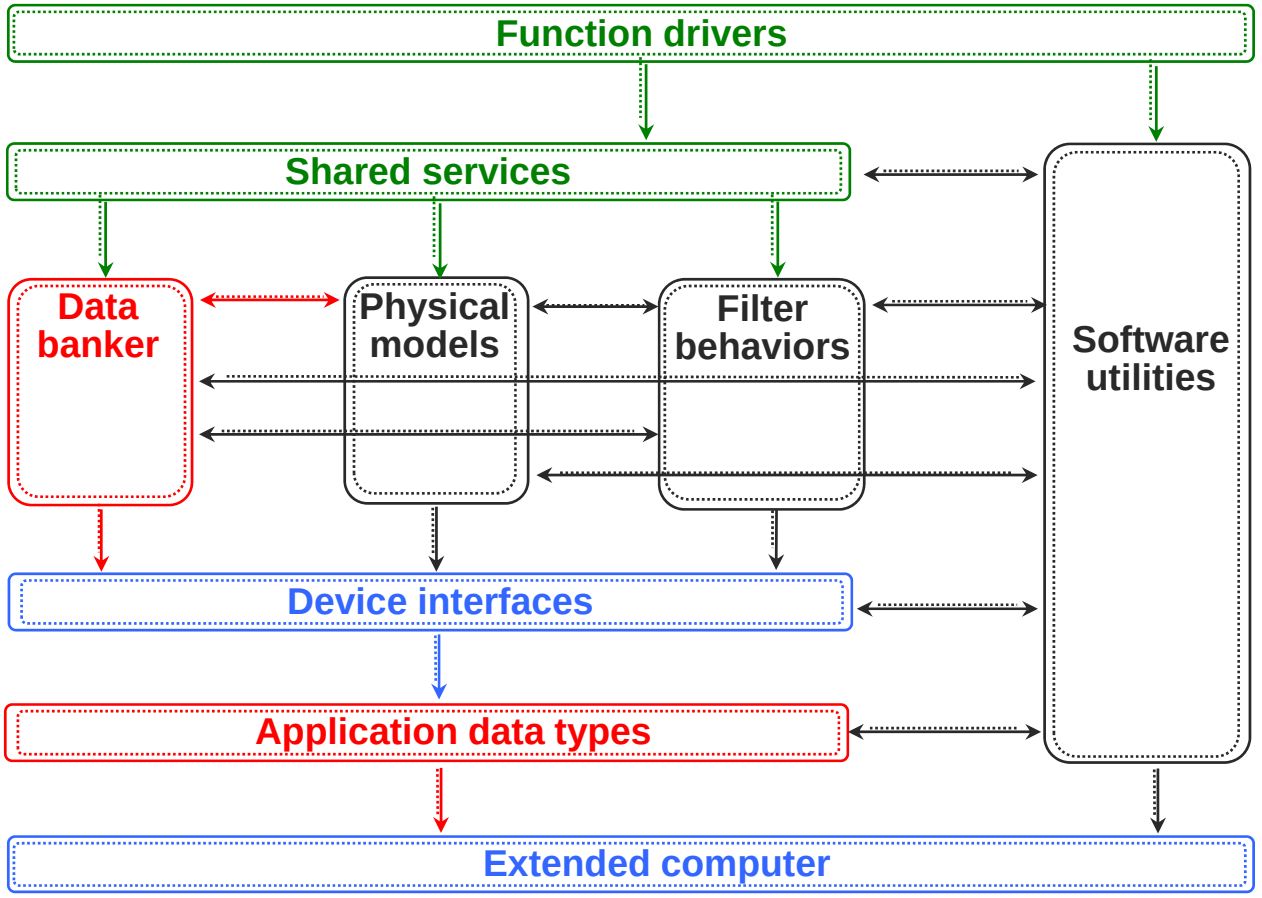
Beispiel: Module Structure 2/2
A-7E Avionics System

Process Structure, Beispiel



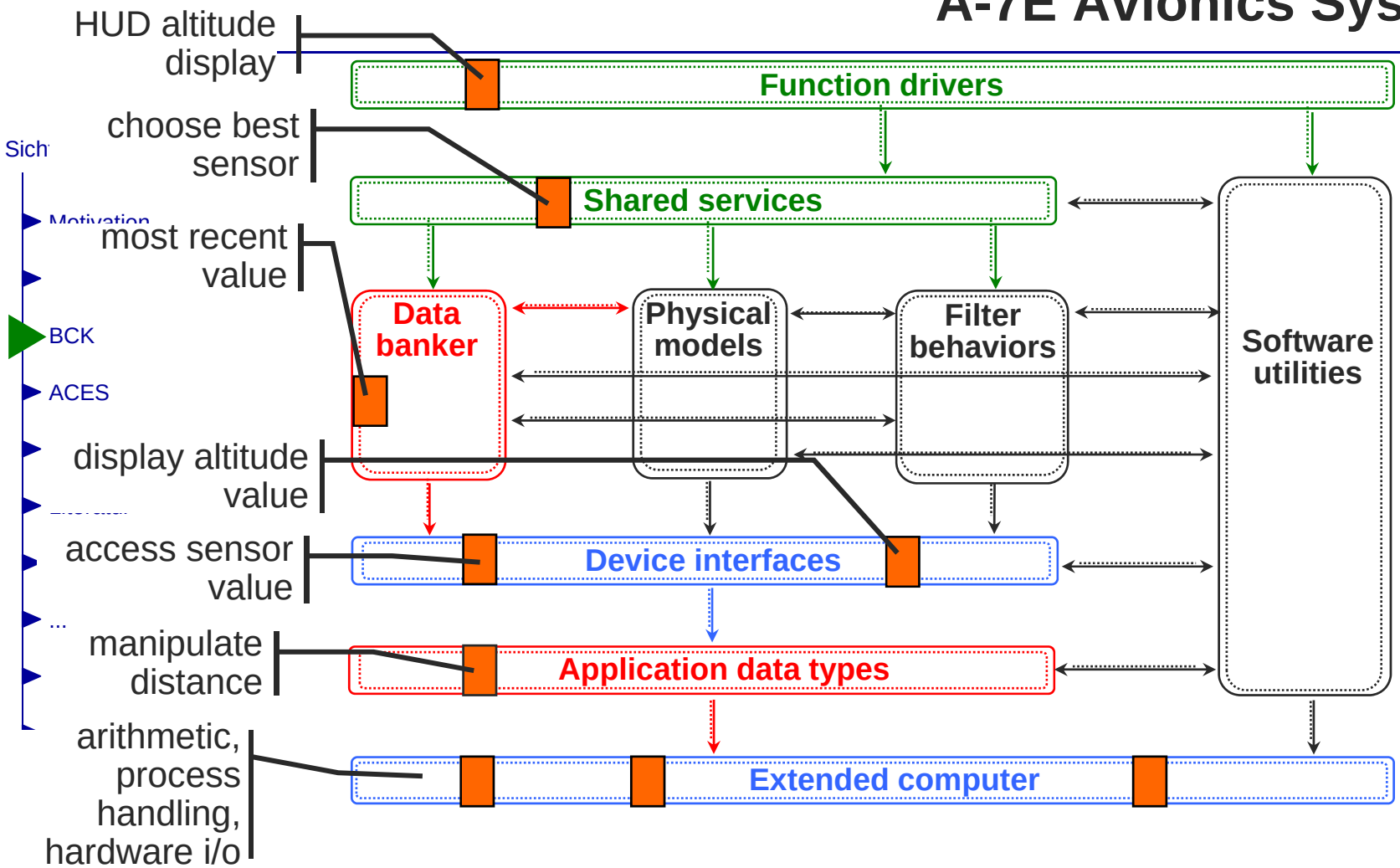
- Komponenten: Tasks, Prozesse
- Beziehungen: “synchronizes with,” “excludes,” “preempts”
- Verwendung: Laufzeit-Performance-Planung, Ausnutzen von (Multi-Prozessor-)Hardware
- Beeinflusste Qualitätsattribute: Performance

Beispiel: Uses Structure
A-7E Avionics System



- Komponenten: Prozeduren
- Beziehungen: “assumes the correct presence of”
- Verwendung: Definition von Teilmengen/Obermengen von Funktionalität
- Beeinflusste Qualitätsattribute: Wiederverwendbarkeit, Testbarkeit, inkrementelle Entwicklung

Beispiel: Uses Structure A-7E Avionics System



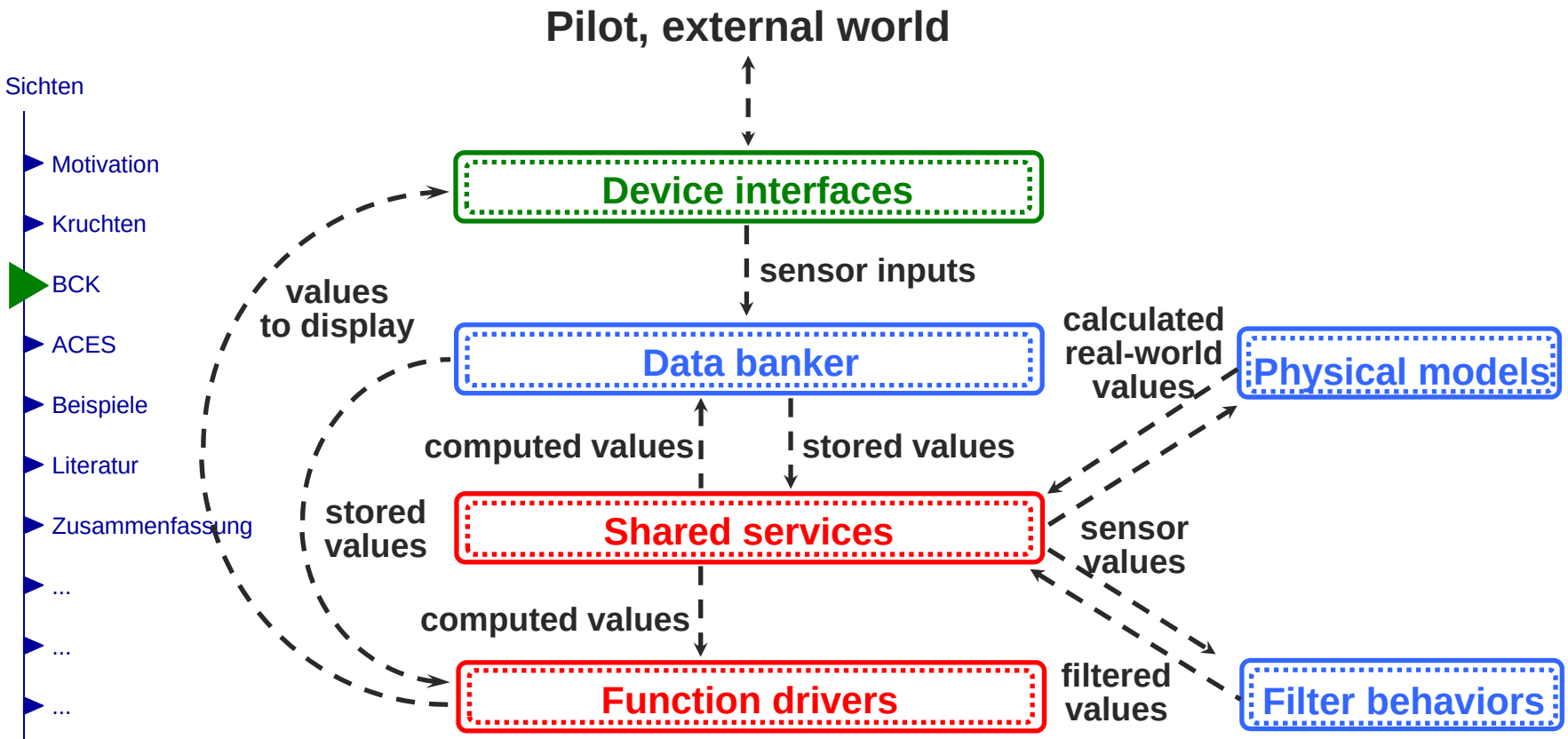
Calls Structure

- Komponenten: Prozeduren
- Beziehungen: "invokes"
- Verwendung: Debugging, Verfolgung von Kontrollfluss
- Beeinflusste Qualitätsattribute: "Baubarkeit" (buildability), Testbarkeit, Wartbarkeit, Verständlichkeit

Abgrenzung von "Uses Structure"

- *Uses*-Beziehung “assumes the correct presence of”
  - Das Ergebnis einer Prozedur wird tatsächlich benutzt
  - Die Prozedur wird vielleicht von einem Parallel-Prozess aufgerufen
- *Calls*-Beziehung "invokes"
  - Eine Prozedur wird tatsächlich aufgerufen
  - Das Ergebnis der Prozedur wird eventuell (hier) nicht verwendet

Beispiel: Data flow Structure



- Komponenten: Programme, Module
- Beziehungen: “may send data to”
- Verwendung: Nachvollziehbarkeit von Funktionalität
- Beeinflusste Qualitätsattribute: Performance, Korrektheit, Genauigkeit

Class Structure

Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...

- Komponenten: Objekte
- Beziehungen: “inherits from,” “is instance of”
- Verwendung: Ausnutzen von Ähnlichkeiten zwischen Objekten
- Beeinflusste Qualitätsattribute: Entwicklungszeit, Wartbarkeit
- Vergleichbar zu UML-Klassendiagrammen: Dargestellt wird ausschließlich die statische Struktur des Systems

Physical Structure

- Komponenten: Tasks, Prozesse, Prozessoren
- Beziehungen: “resides on same processor”
- Verwendung: Allokation von Prozessen auf Prozessoren
- Beeinflusste Qualitätsattribute: Performance, Verfügbarkeit, Sicherheit

- Dargestellt wird die Abbildung von Software auf Hardware, das entspricht der Physikalischen Sicht von Kruchten
- Interessant (nur) in verteilten oder Mehrprozessor-Systemen

Nochmal in der Übersicht: *Architectural Structures* [Bass, Clements, Kazman]

- Module
  → Modifizierbarkeit, Entwicklung von Teilsystemen
- Process
  → Parallelität, Verteilung, Synchronisation, Performance, Skalierbarkeit
- Uses
  → Entwicklung von Teilsystemen
- Calls
  → Beobachtung des Kontrollflusses (Debugging!), Testbarkeit, Wartbarkeit
- Data flow
  → Verteilung der Funktionalität, Performance, Korrektheit, Genauigkeit
- Class
  → Projektplanung, Wartbarkeit
- Physical
  → Verfügbarkeit, Zuverlässigkeit, Performance

Decomposition Dimensions by Fraunhofer IESE

- **Decomposition Dimensions**
  address the characteristics of a software system at different **abstraction levels**:

How will the system interact with its environment?

Context — System seen as black box in its environment → Decomposing the system within its environment

How will the system work at runtime?

Runtime — System seen as Runtime entities → Decomposing the system into Runtime entities

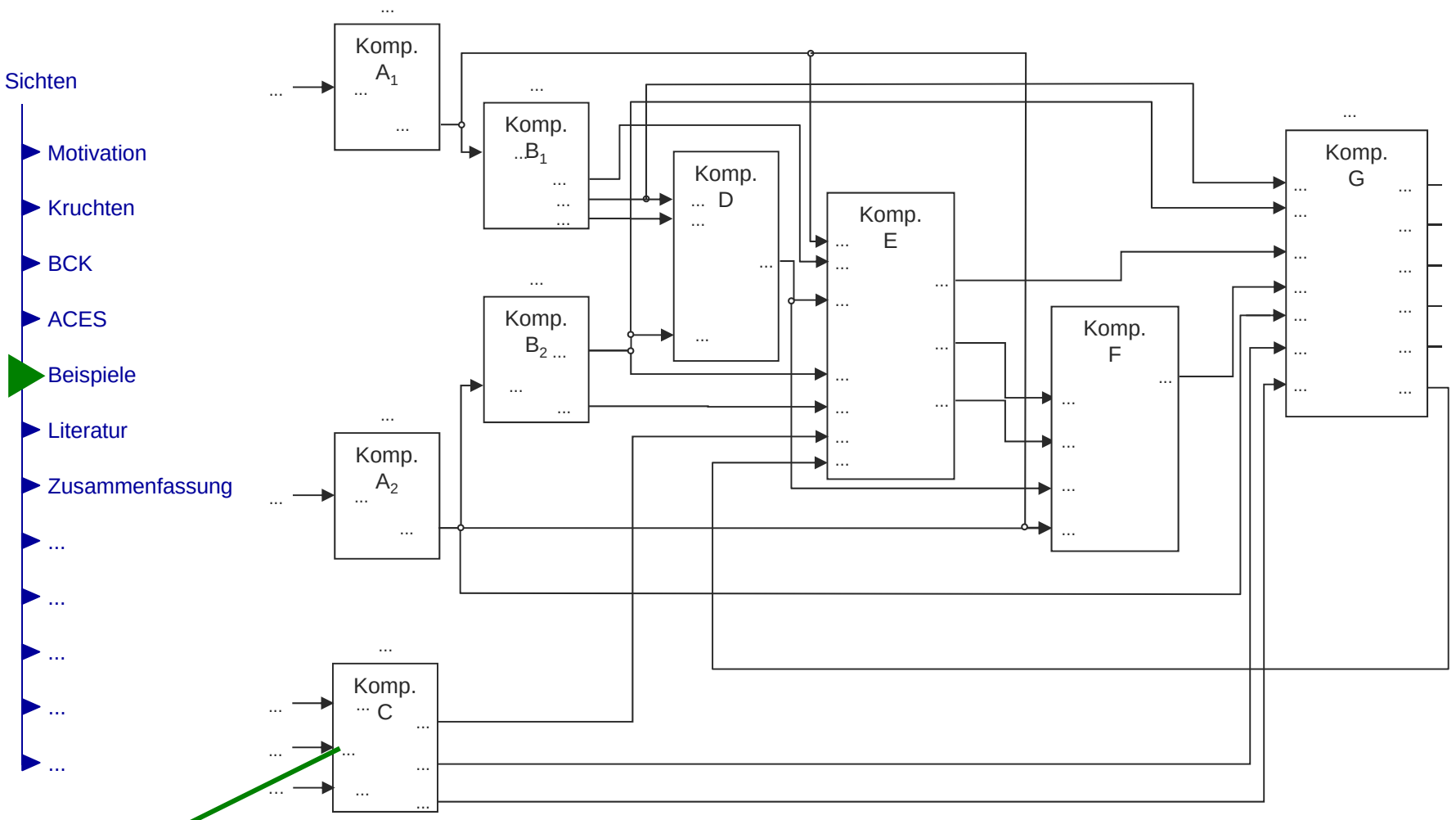
How will the system be implemented?

Development-Time — System seen as Development-time entities → Decomposing the system in Development-time artifacts

1

Fraunhofer IESE

Beispiel: Statische vs. Dynamische Sicht

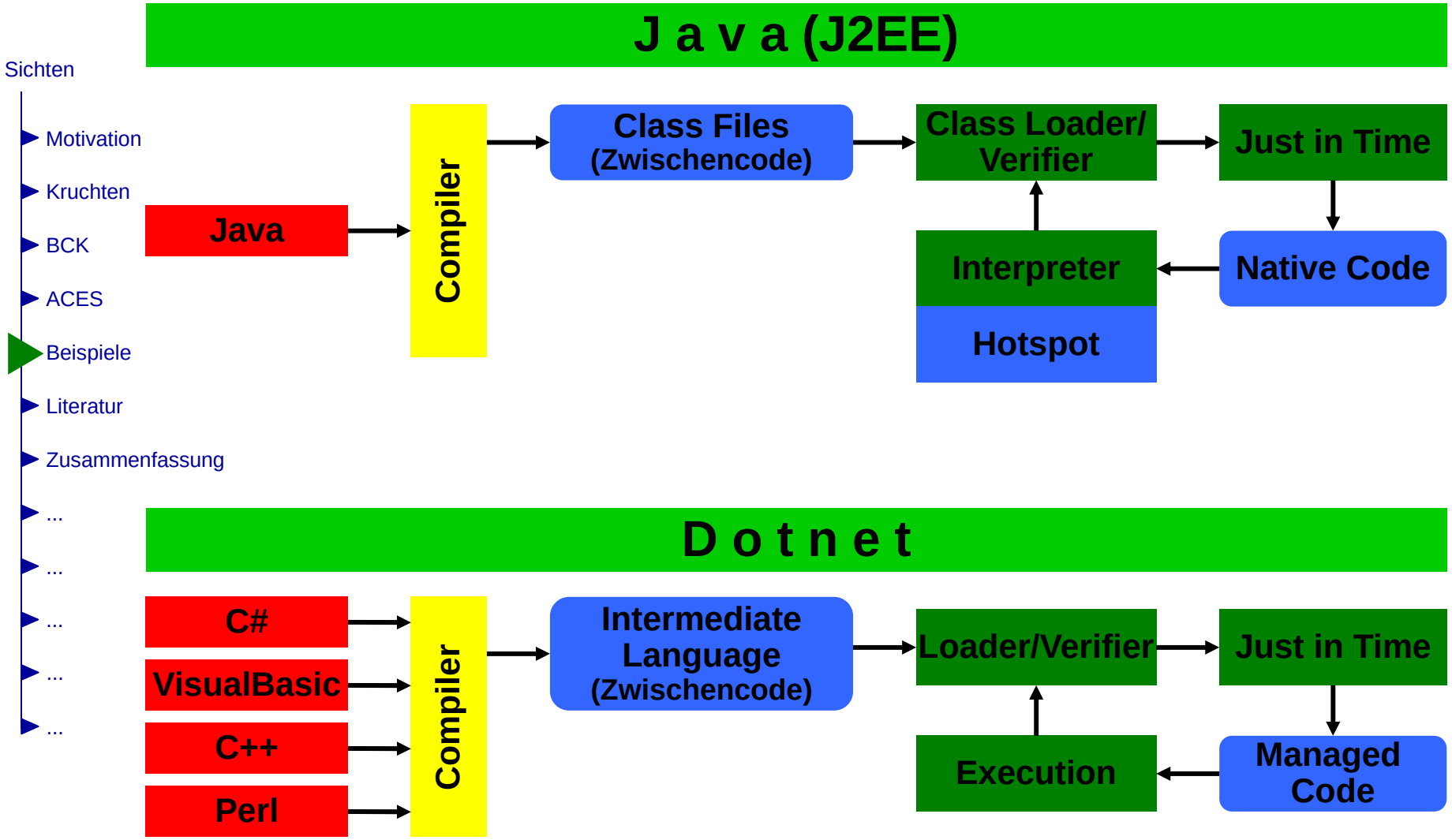


Komponente mit eingehenden und ausgehenden Daten samt Beschreibung und zugehörigen „Ports“

Zur Laufzeit: *viele* kleine "Tasks"/ Funktionen

J2EE als Vorlage für MS-Dotnet (.net)

[COMPUTER ZEITUNG 22/2002]

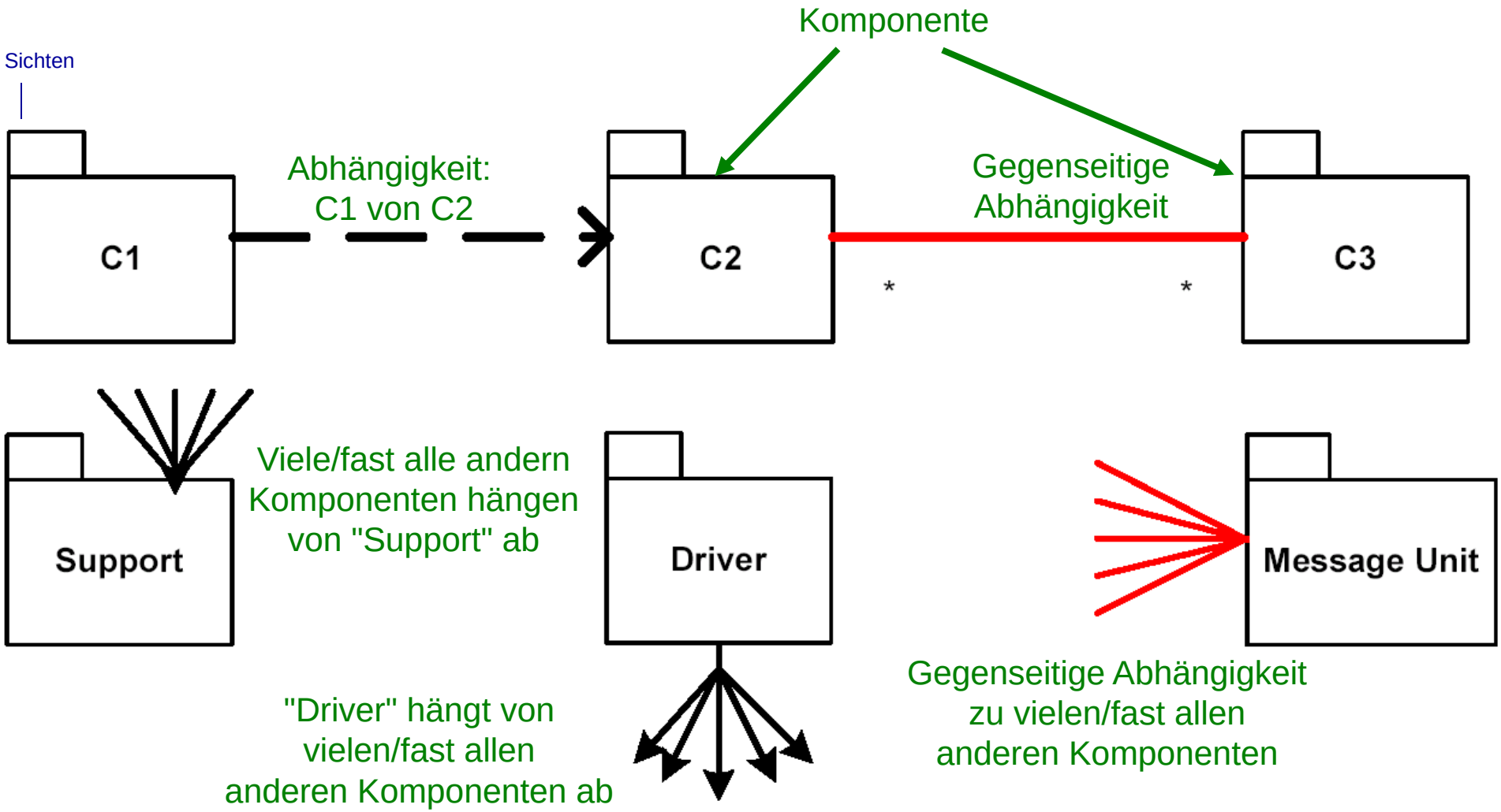


Welche Sicht ist das?

Große Software-Systeme

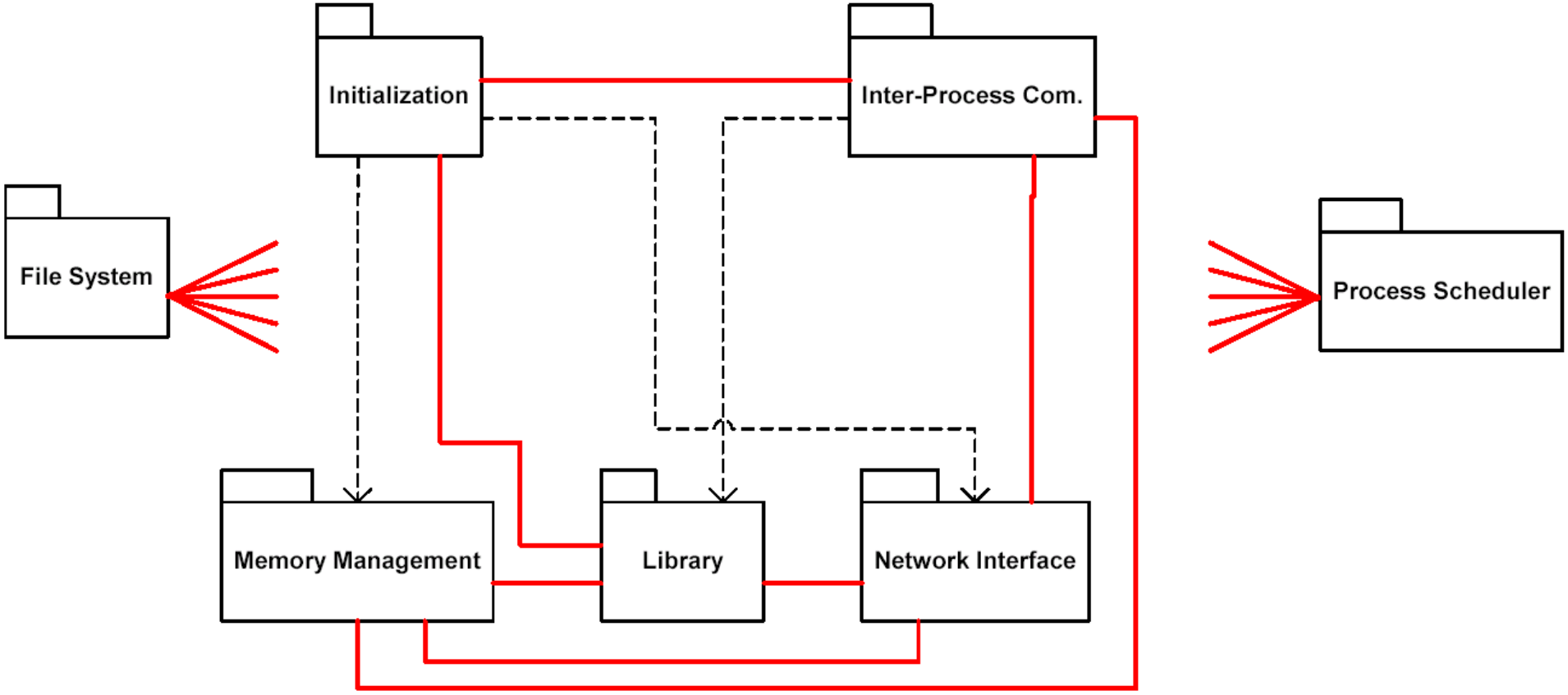
System	Domäne	Größe [kLoC]	Anzahl der Dateien	Anzahl der Komponenten (Top-Level)	Anzahl der Level
tcsh	Unix Shell			5 (4)	2
Apache	Web Server			21 (9)	2
AOL Server	Web Server			22 (10)	3
Linux Kernel	OS			128 (7)	5

Eine mögliche Notation für die Darstellung der Entwickler-Sicht



Architektur großer Systeme: Linux Kernel

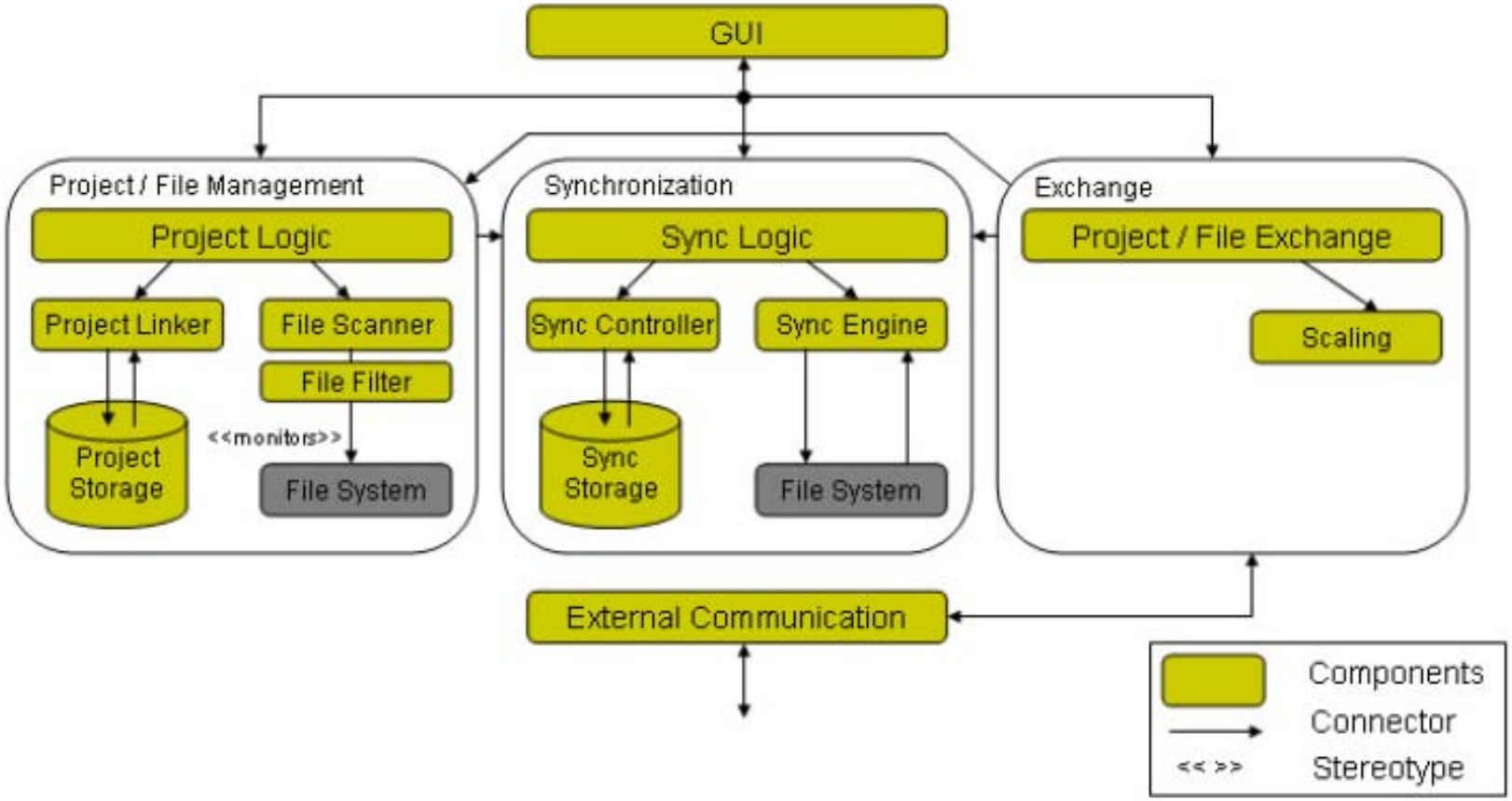
- 800 kLoC in C programmiert
- 557 Dateien
- 128 Komponenten (7 auf Top-Level, 5 Levels)

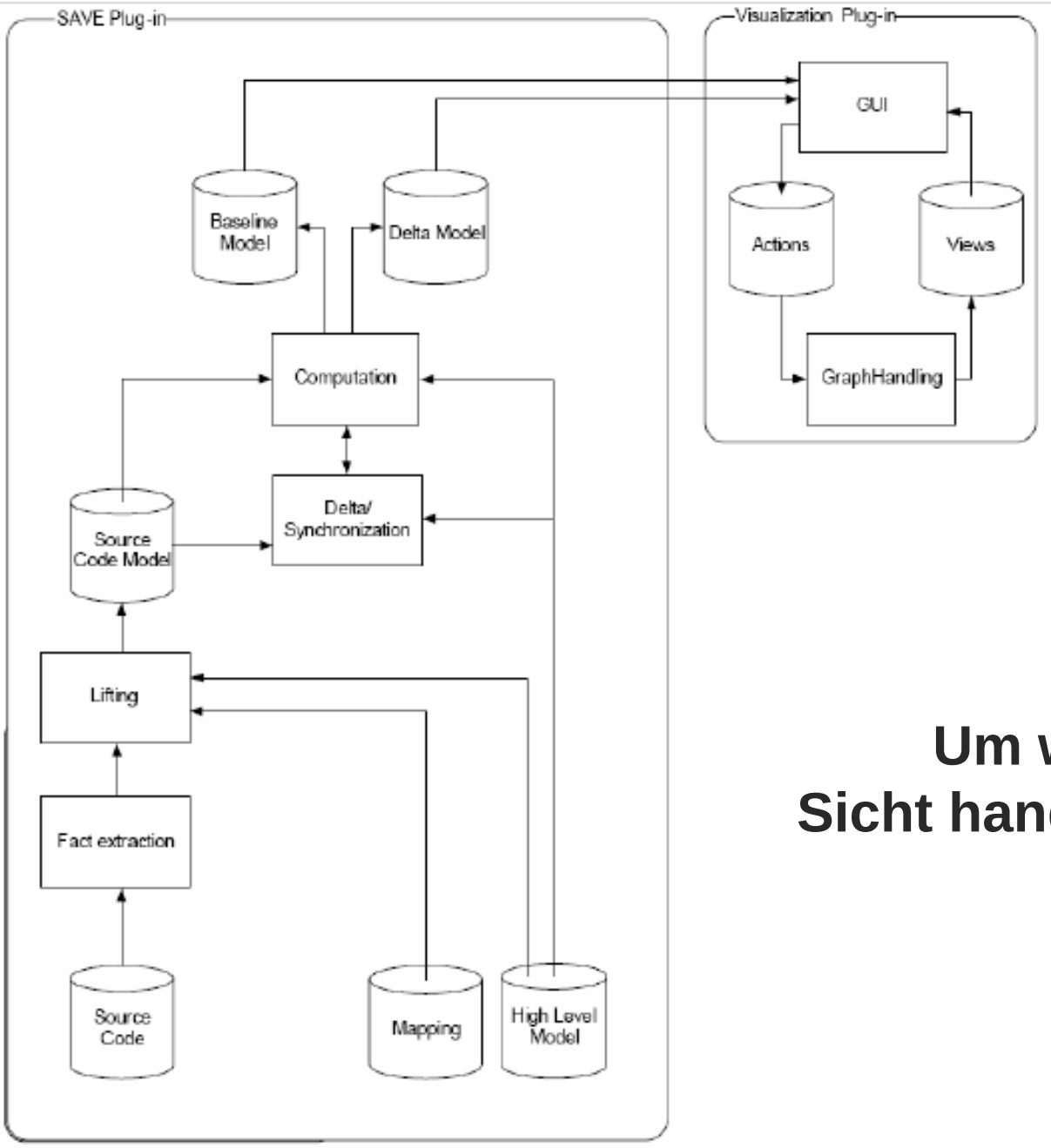


Um welche Sicht handelt es sich?

Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfas
- ...
- ...
- ...
- ...
- ...





Um welche Sicht handelt es sich?

Um welche Sicht handelt es sich?

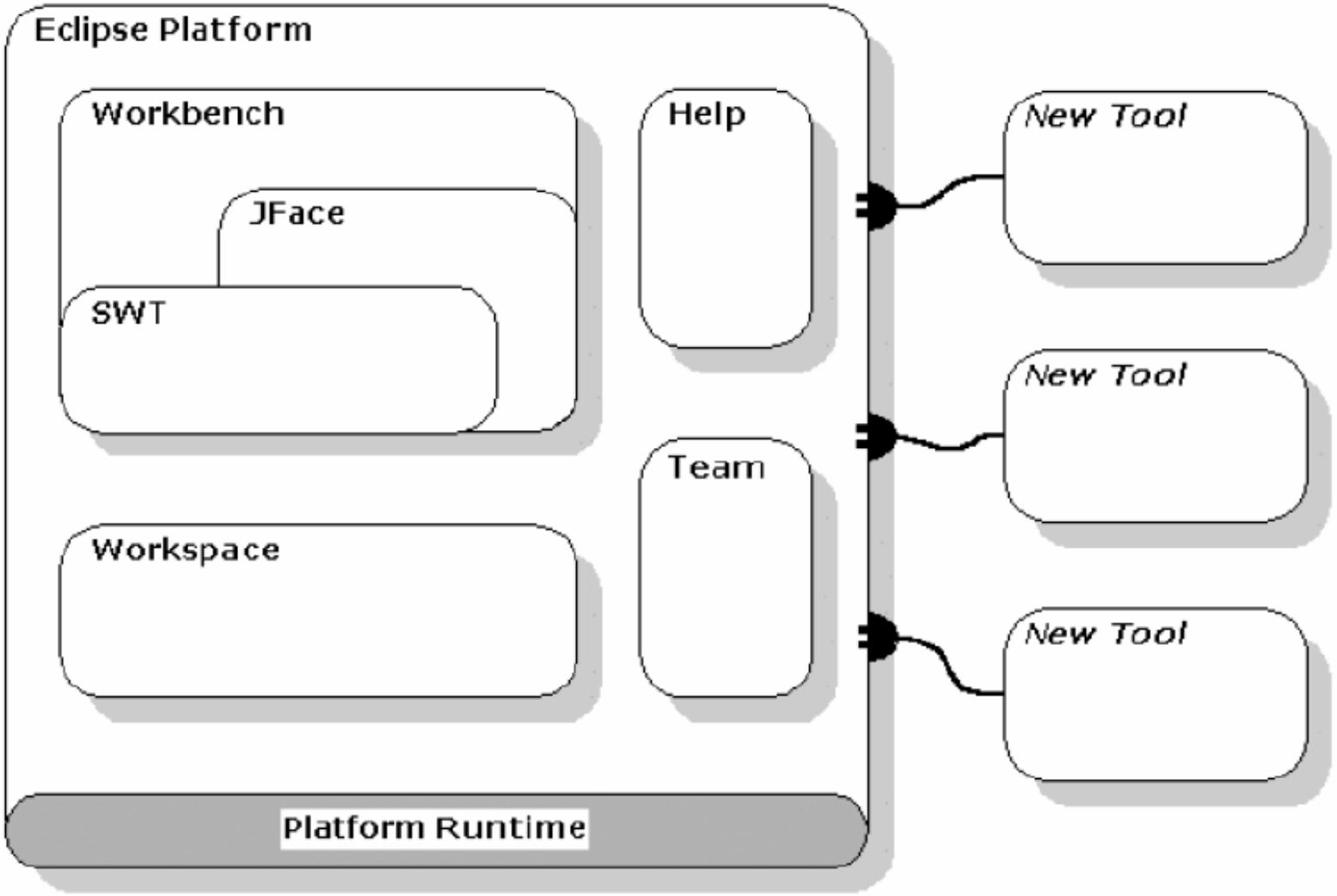
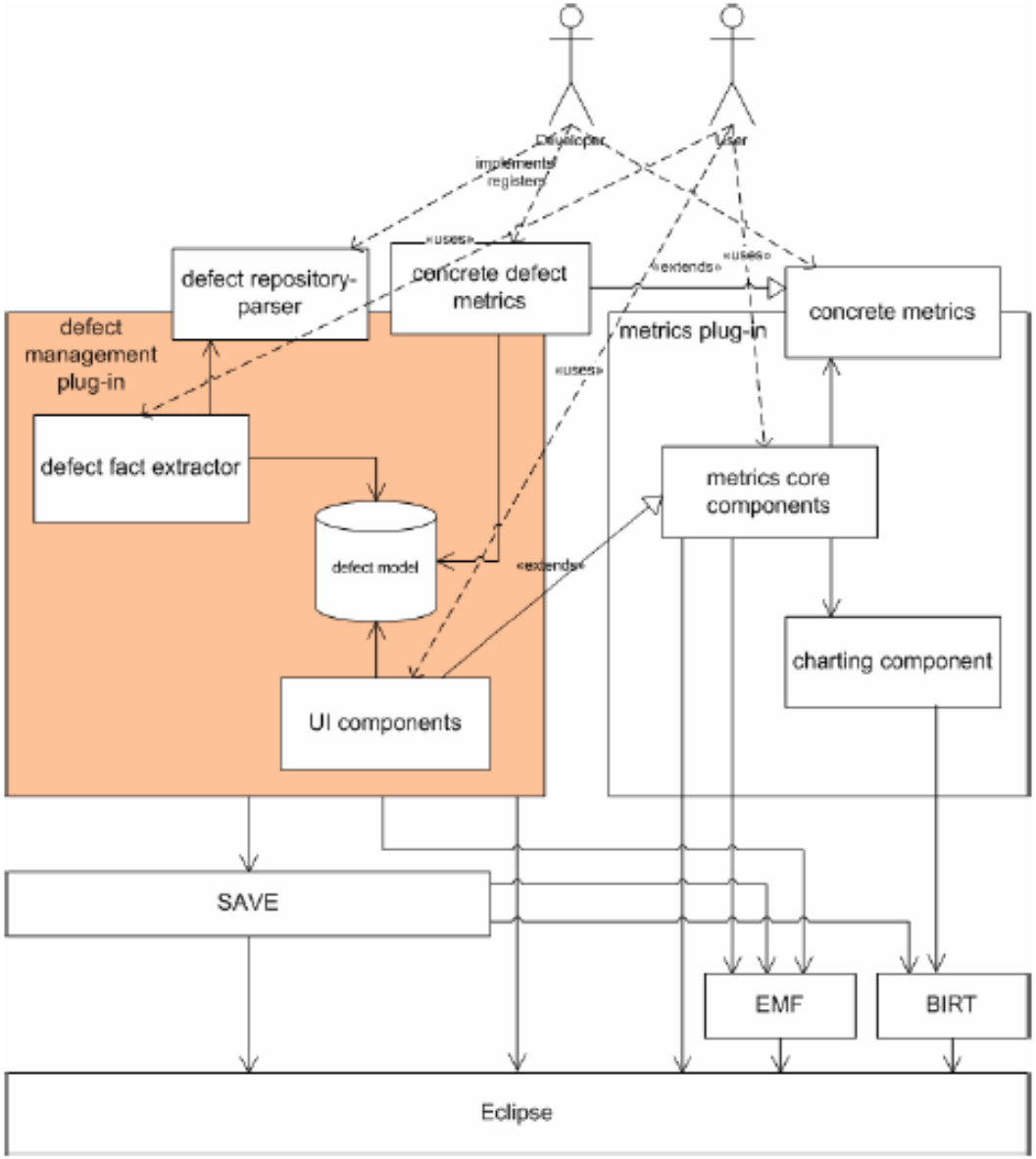


Figure 5.1: Eclipse platform architecture [IBM'03]

Um welche Sicht handelt es sich?

Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...



Um welche Sicht handelt es sich?

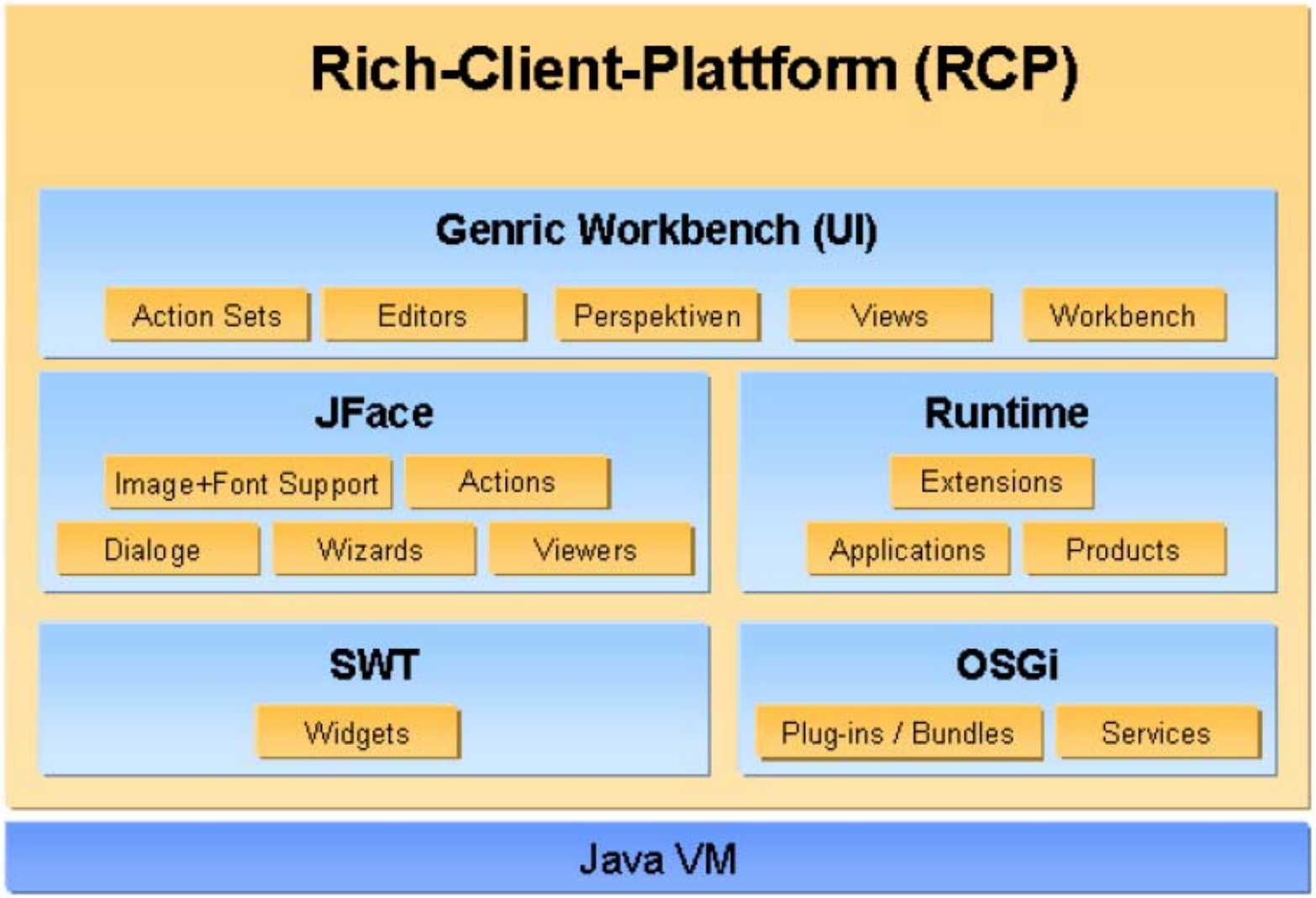


Abbildung 1: RCP-Architektur
(vgl. [Daum06], S. 12 und [McAffer05], S. 15)

Um welche Sicht handelt es sich?

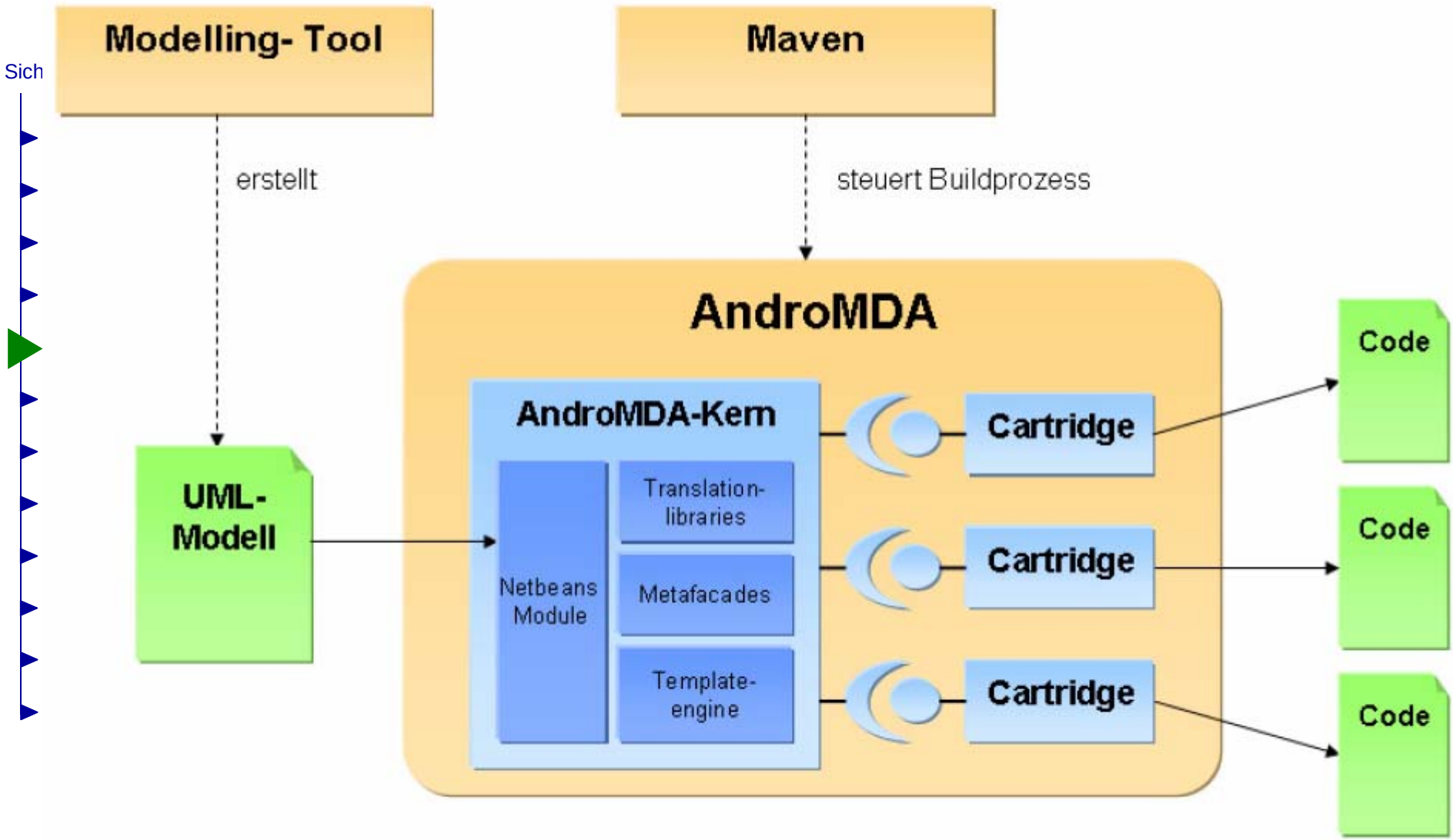
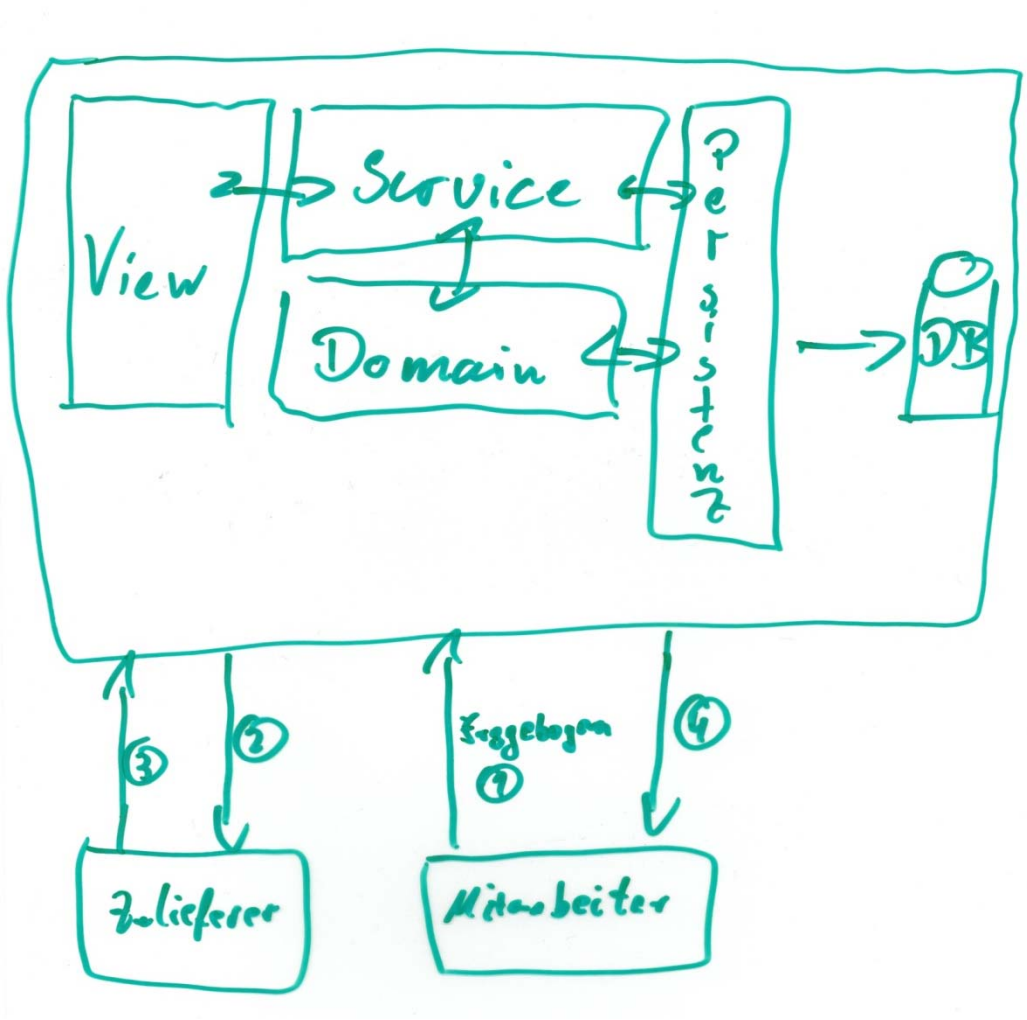


Abbildung 6: Aufbau von AndroMDA

Ihre System - Darstellungen

Sichten

- Motivation
- Kruchten
- BCK
- ACES
- Beispiele
- Literatur
- Zusammenfassung
- ...
- ...
- ...
- ...
- ...



Literatur

- Bass, Clements, Kazman: *Software Architecture in Practice*, 2nd Edition, Addison Wesley
- G. Booch: *Object-Oriented Design with Applications*, Benjamin Cummings, 1991
- J.-F. Girard: *ADORE-AR:Software Architecture Reconstruction with Partition and Clustering*, PhD Thesis, Universität Kaiserslautern, 2005
- Jens Knodel: *Lecture "Software Architecture"*, Hochschule Mannheim, 2012
- Philippe Kruchten: *The "4 + 1" View Model of Software Architecture*, IEEE Software 12 (6), November 1995, pp. 42-50
- Linda Northrop: *Let's Teach Architecting High Quality Software*, 19th Conference on Software Engineering Education and Training (CSEE&T), 2006
- D.L. Parnas: *On the Criteria To Be Used in Decomposing Systems Into Modules*, *CACM*, Vol. 15(12), 1972
- T. Keuler, J. Knodel, M. Naab: *Architecture-centric Software and System Engineering*, http://www.iese.fraunhofer.de/de/competencies/architecture/philosophy_architecture.html
- Software-Kompetenz.de: http://www.software-kompetenz.org/

Zusammenfassung

Verwirrend: Es gibt

viele Namen für

viele Sichten

Aber

- (Fast) alle Autoren ermutigen, (nur) *relevante* Sichten *auszuwählen*
- Es kommt darauf an, dass Sie "Ihr" System allen Stakeholdern präsentieren können;
  welche *Namen* die Sichten haben ist *nicht relevant*