14.–17. 09. 2009
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Immer schön den Durchblick behalten

Nachvollziehbarkeit von der Analyse bis zur Software-Architektur in der modellbasierten Entwicklung

Dr. Stefan Queins, André Pflüger

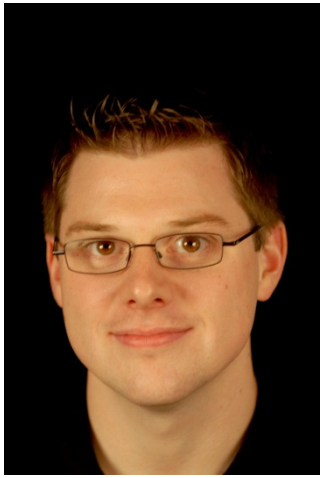
SOPHIST GmbH

Wer sind wir?

SOPHIST

Dr. Stefan Queins:

- Berater und Trainer
- Spezialist in den Bereich RE/RM , OOA/OOD
- Wissen aus den Bereichen Forschung und Entwicklung
- Autor von UML-Glasklar und Requirements-Engineering und -management



André Pflüger:

- Berater im Bereich Analyse und Architektur mit der UML
- Doktorand im Bereich Unterstützung der Entwicklung durch Modellierung

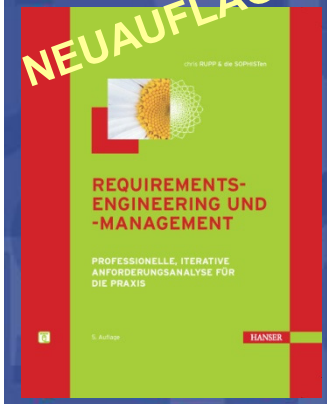
Wer schreibt der bleibt

Die Bücher der SOPHISTen

SOPHIST

REQUIREMENTS ENGINEERING & MANAGEMENT

NEUAUFLAGE!



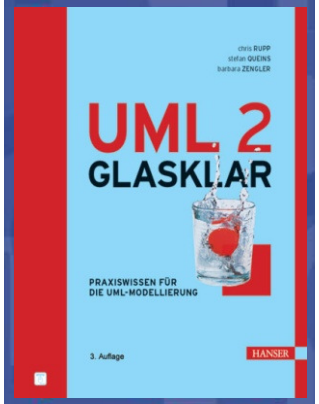
Chris Rupp &
die SOPHISTen

5. Auflage

seit Juli 2009 im
Handel

ISBN:
978-3-446-41841-7

UML 2 GLASKLAR



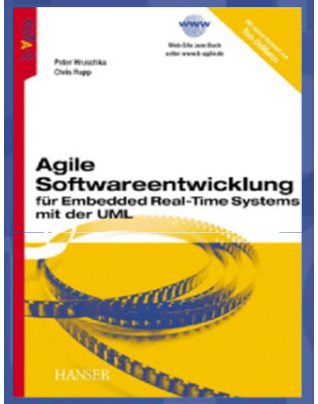
Chris Rupp,
Dr. Stefan Queins &
Barbara Zengler

3. Auflage

seit August 2007
im Handel

ISBN:
978-344-641-1180

AGILE SOFTWARE-ENTWICKLUNG



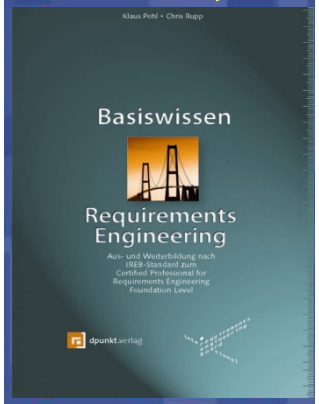
Chris Rupp &
Peter Hruschka

seit Februar 2002
im Handel

ISBN:
978-344-621-9977

BASISWISSEN REQUIREMENTS ENGINEERING

NEU!

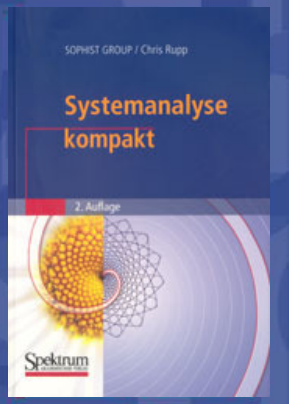


Chris Rupp &
Klaus Pohl

seit März 2009
im Handel

ISBN:
978-389-864-6130

SYSTEMANALYSE KOMPAKT



Chris Rupp &
die SOPHISTen

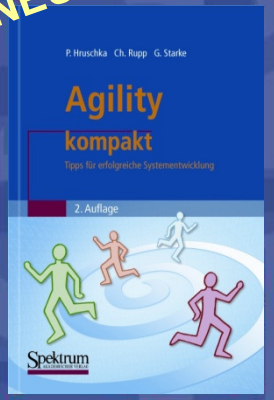
2. Auflage

seit März 2008
Im Handel

ISBN:
978-382-741-9361

AGILITY KOMPAKT

NEUAUFLAGE!



Chris Rupp,
Peter Hruschka &
Gernot Starke

2. Auflage

seit April 2009
im Handel

ISBN:
978-382-742-0923

SOPHIST

E

Einleitung

U

Unterstützung durch die UML

V

Validieren von Design-entscheidungen

Z

Zusammen-fassung und Ausblick

Immer schön den Durchblick behalten

SOPHIST

- Radarsysteme
- Entwicklungsprozess

Einleitung

Radarsysteme

Beispiele für Einsatzgebiete

- Airport Surveillance Radar (ASR)

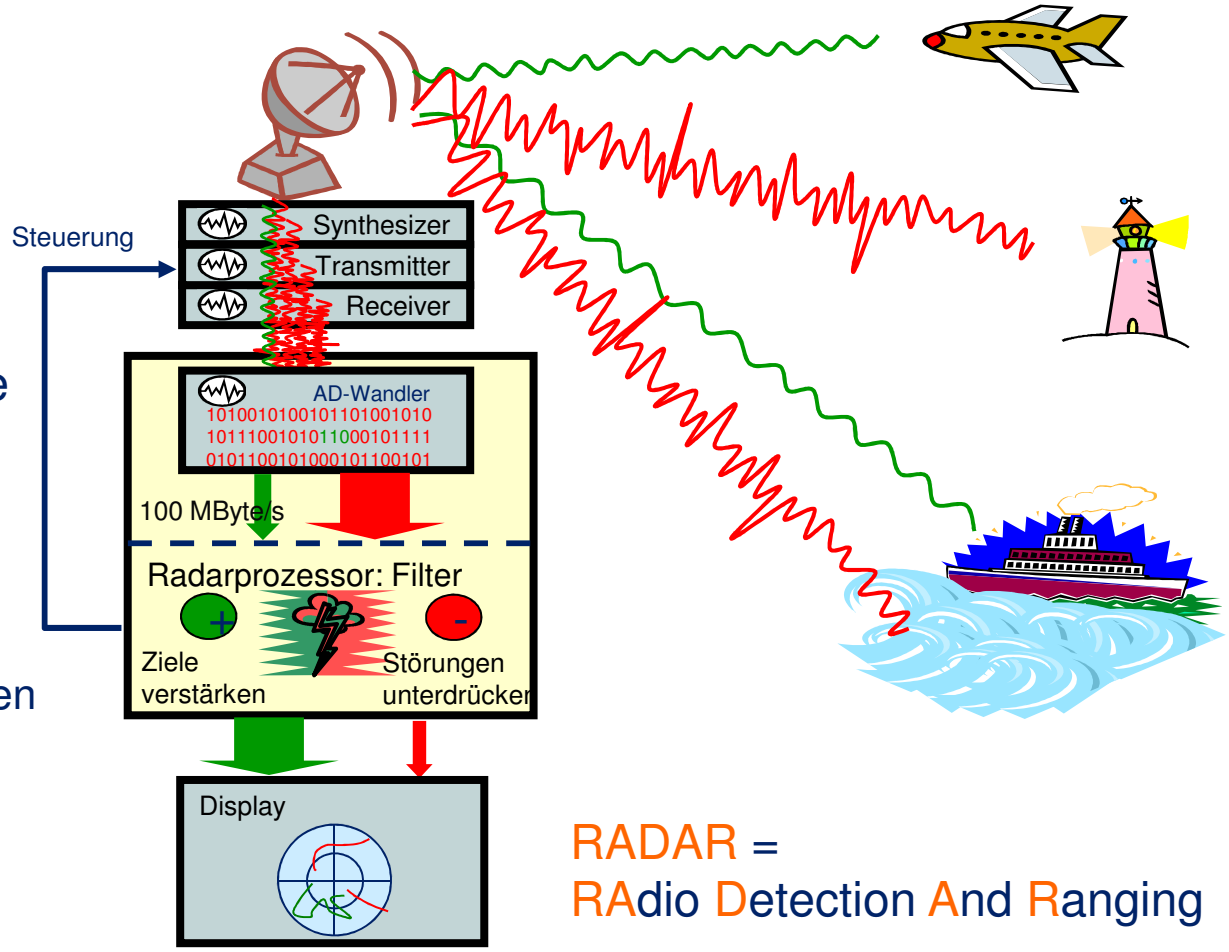
- BodenÜberwachungsRadar (BÜR)

- Fregatte vom Typ F-125

Radarsysteme

Funktionsweise

- Signale aussenden und empfangen
- Verarbeitung der empfangenen Signale
  - A/D-Wandlung
  - Filterung
  - Verstärkung
  - Störungen unterdrücken
  - Daten verarbeiten



RADAR =
RAdio Detection And Ranging

Radarsysteme

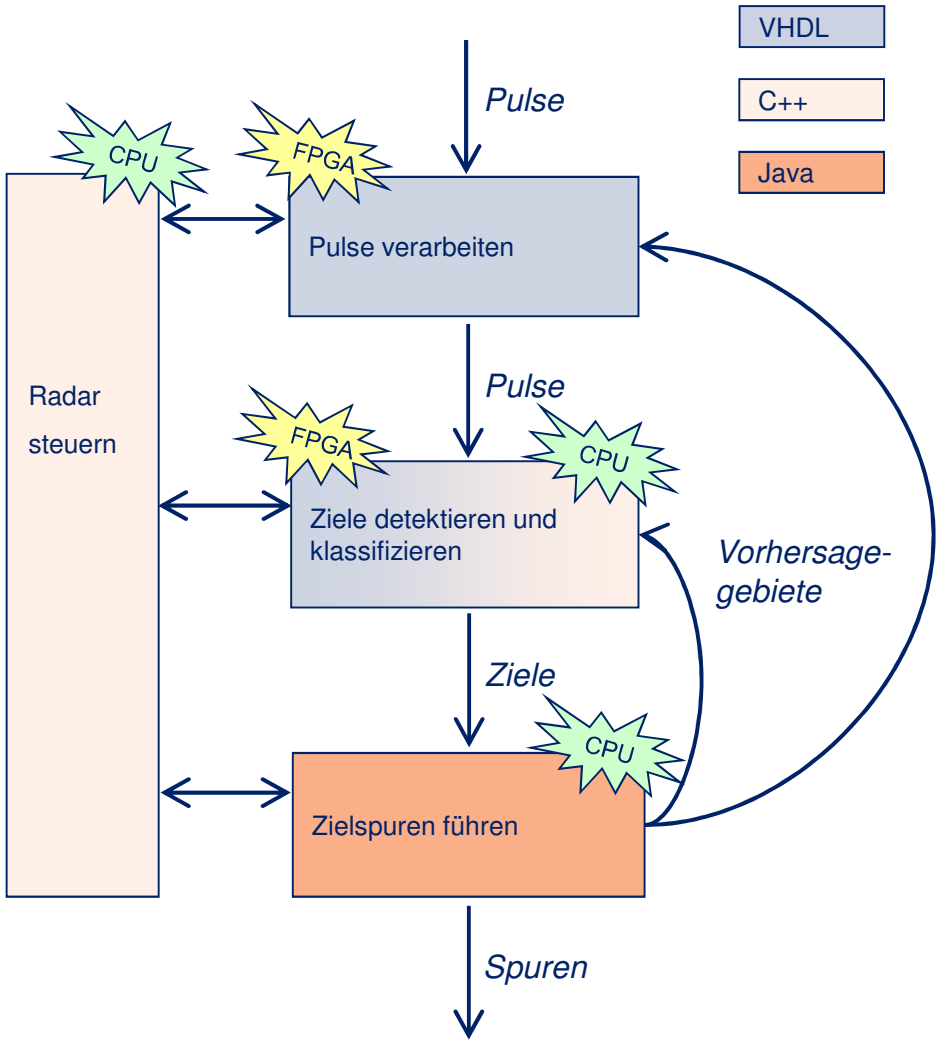
Radarverarbeitung

- Aufgaben der Radarverarbeitung
  - Ziele detektieren
  - Ziele klassifizieren
  - Ziele verfolgen
- Abtastung eines 4-dimensionalen Raums
  - Entfernung x Azimut x Elevation x Zeit
- Dopplerauswertung liefert weitere Dimension: Dopplerfrequenz
- Suche nach Mustern von Zielen und Störungen in einem 5-dimensionalen Raum
  - Entfernung x Azimut x Elevation x Zeit x Frequenz
  - in Echtzeit
  - bei Datenraten von ca. 100 MByte/s

Radarsysteme

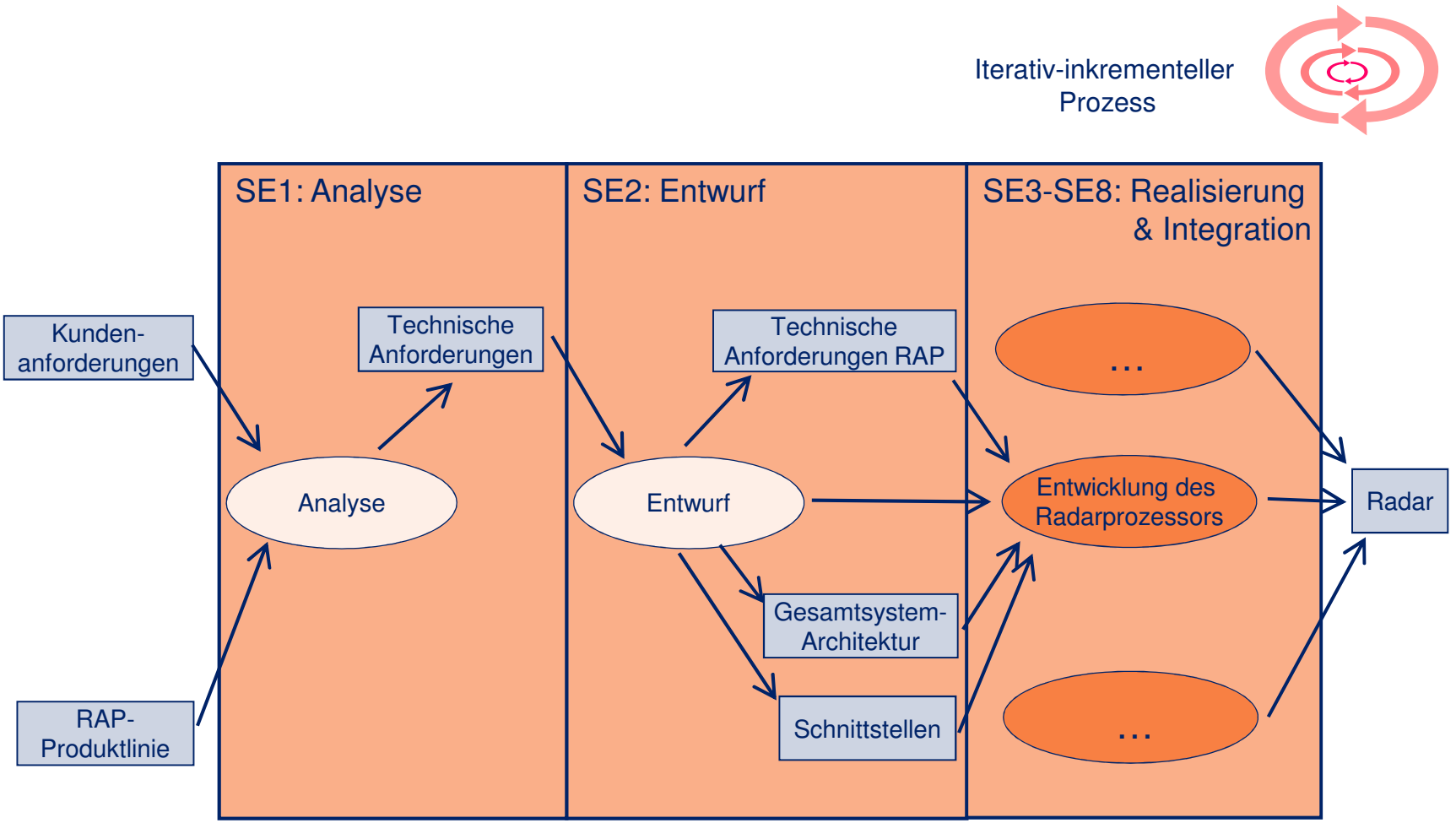
Common Radar Processor

- Gemeinsame Plattform (Hardware, Software, Werkzeuge)
  - Für verschiedene Projekte
  - Für alle funktionalen Komponenten
- Gemeinsame Entwicklungsmethodik
- 3 Abteilungen beteiligt, aber trotzdem ein RAP-Team



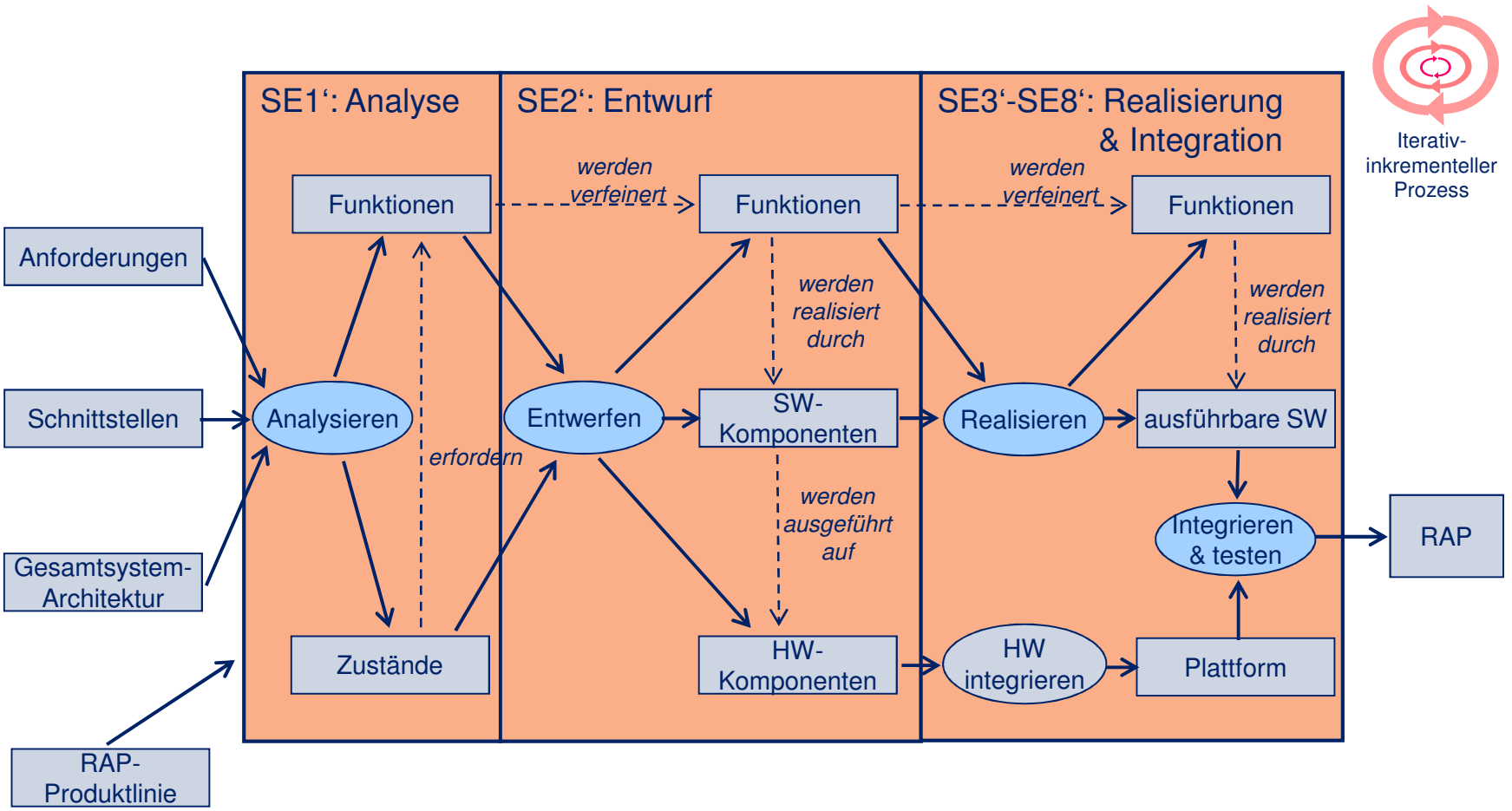
Entwicklungsprozess

Wo wir uns einordnen



Entwicklungsprozess

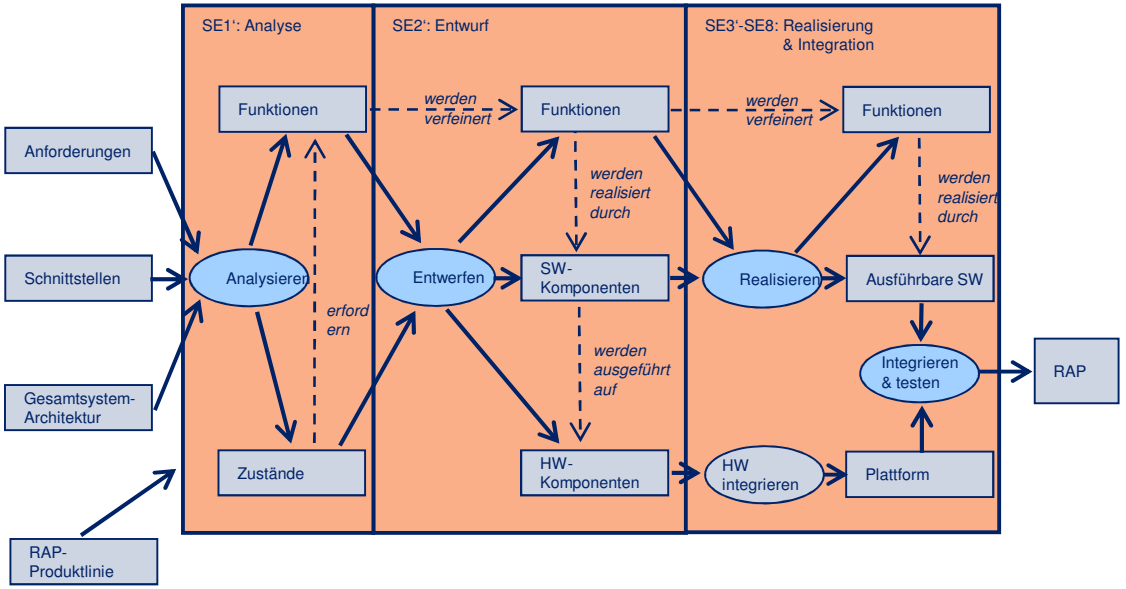
Erneuter Start im V-Modell bei SE1 für den Radarprozessor



Unterstützung durch die UML

Was unterstützen wir und was haben wir zu beachten?

- Unterstützung des Gesamtprozesses durch die UML
- Analyse
  - Funktionen
  - Zustände
- Entwurf
  - Funktionen
  - Struktur
  - Schnittstellen
  - Verteilung der Funktionen
- Realisierung & Test
  - Ähnlich zu Entwurf
  - Releases



- Randbedingungen / Ziele
  - Unsicherheit bezüglich Realisierung
  - Sich ändernde Anforderungen und Randbedingungen
  - Wiederverwendbarkeit über Projekte hinweg
  - Vollständigkeit in der Entwicklung
  - Automatische Analysen und Synthesen

SOPHIST

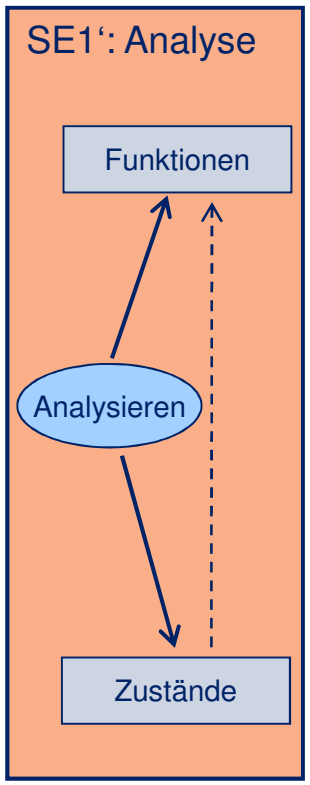
- Analysephase
- System-Architektur

Unterstützung durch die UML

Analysephase

Ziele und Vorgehen

SOPHIST

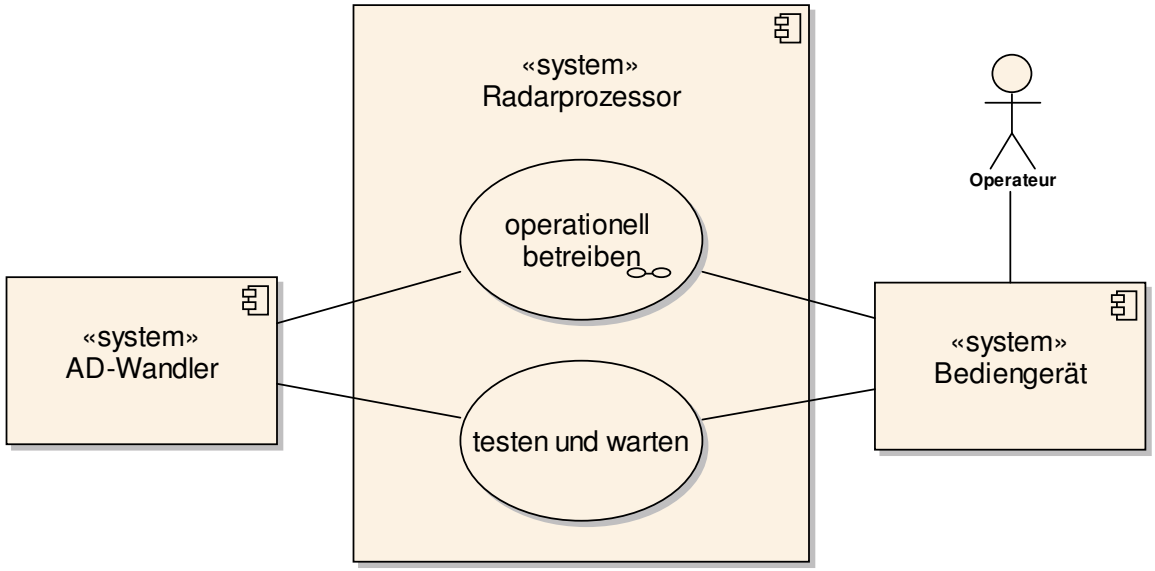


- Ziele
  - Untersuchung der Anforderungen
  - Festlegen der fachlichen Abläufe
- Vorgehen bei der Analyse
  - Definition der Anwendungsfälle
  - Verfeinerung durch Aktivitätsdiagramme
  - Systemzustände als Vorbedingungen
  - Begriffsmodelle zur Kommunikation

Analysephase

Anwendungsfälle des RAP

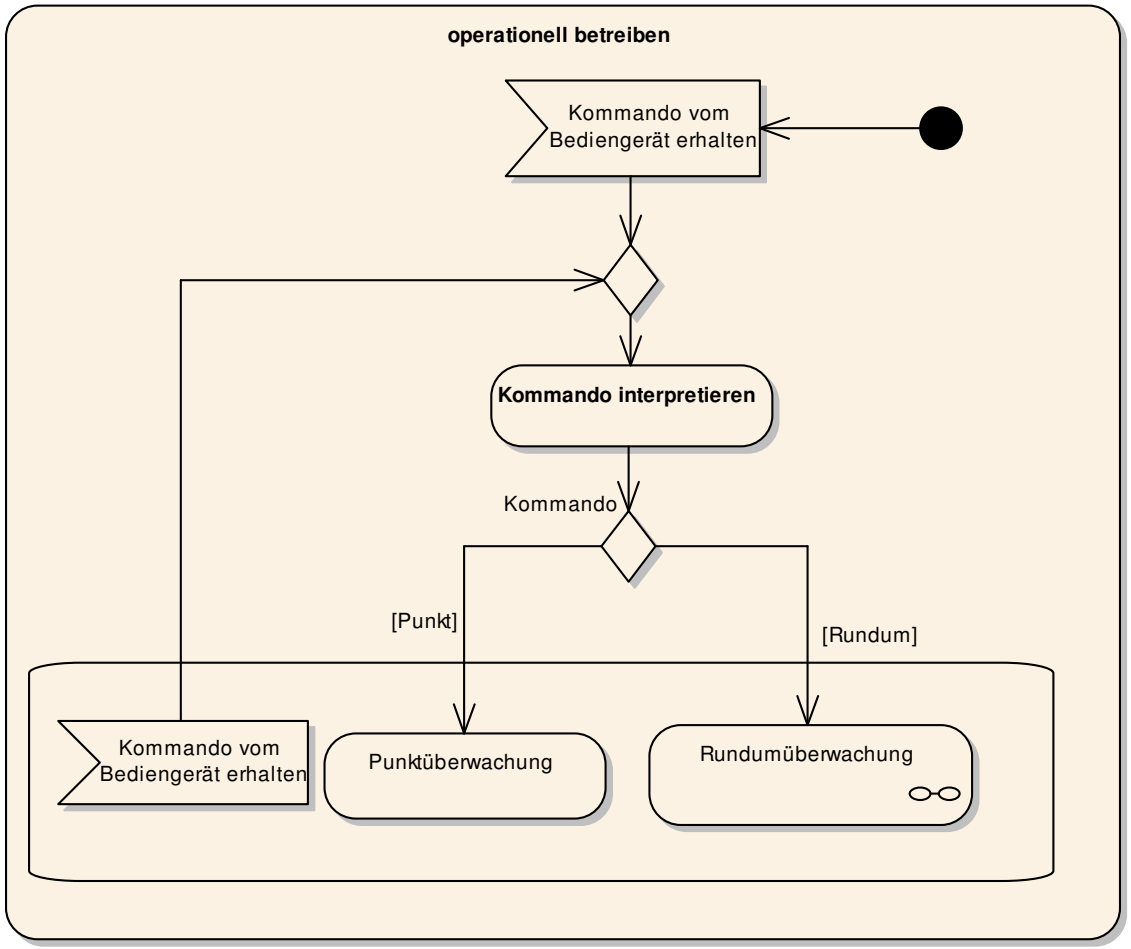
- Nach außen zur Verfügung stehende Funktionalitäten
- Darstellung der an den Use-Cases beteiligter Akteure
- Black-Box-Sicht des Systems



Analysephase

Aktivitätsdiagramme

- Auswahl aus den Standard-Notations-elementen
- Vollständig verfeinert bis auf Fehlerfälle

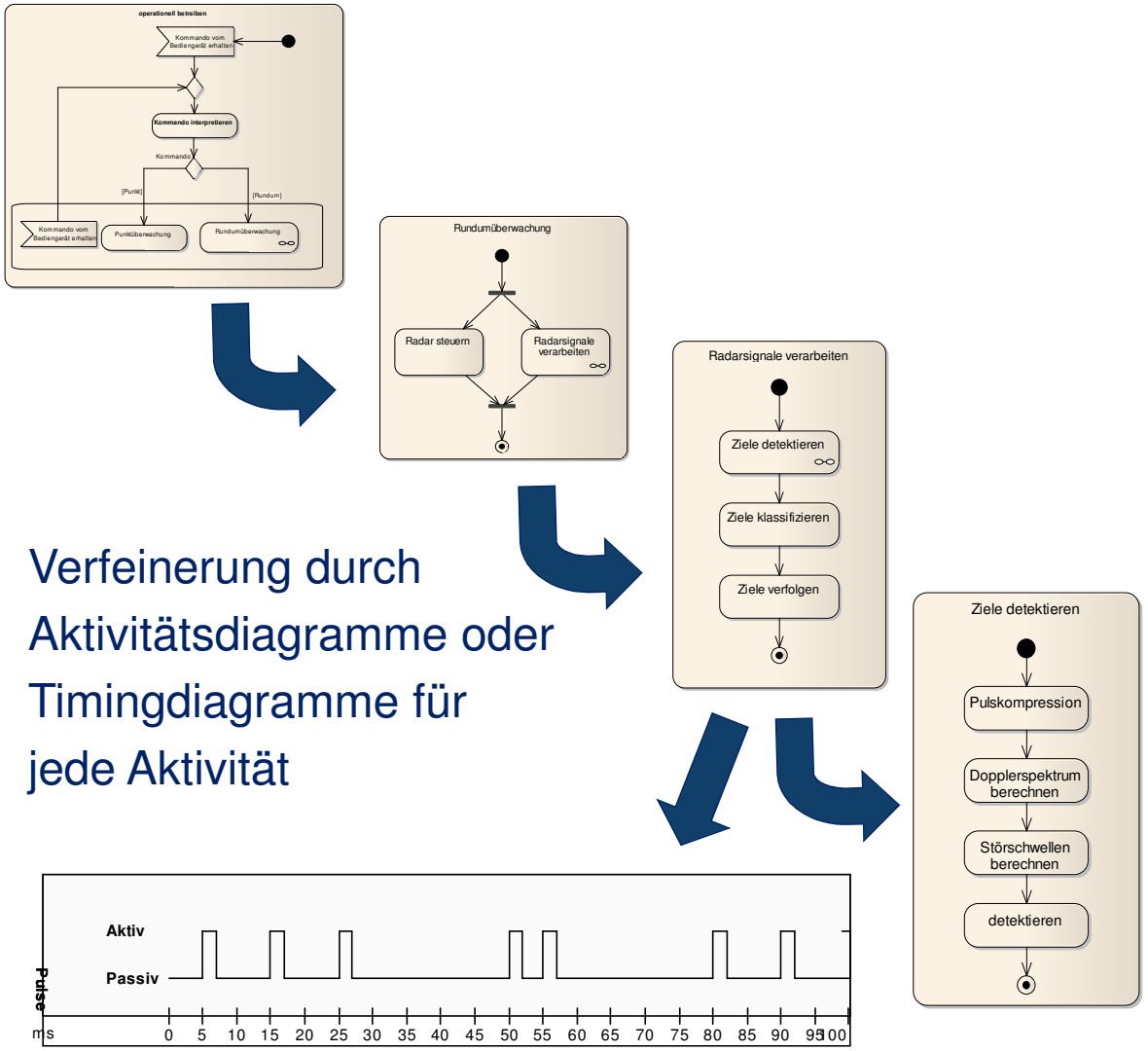


Analysephase

Aktivitäten verfeinern

- Fachliche Abläufe darstellen
- Den benötigten Rechenaufwand darstellen
- Ebene finden, die in der Architektur angewandt werden kann

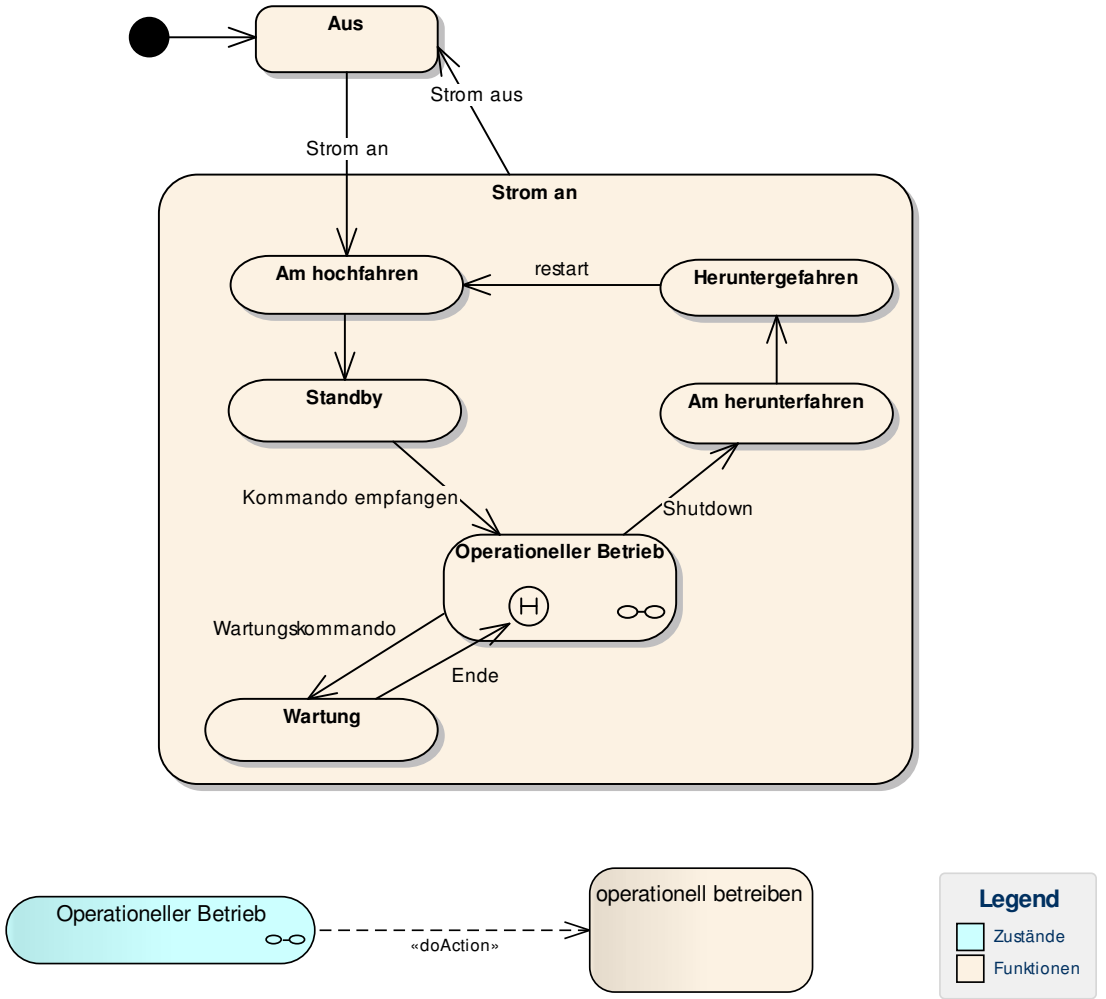
Grundlage der HW-Architektur und HW-SW-Zuordnung



Analysephase

Verwendung von Zustandsautomaten

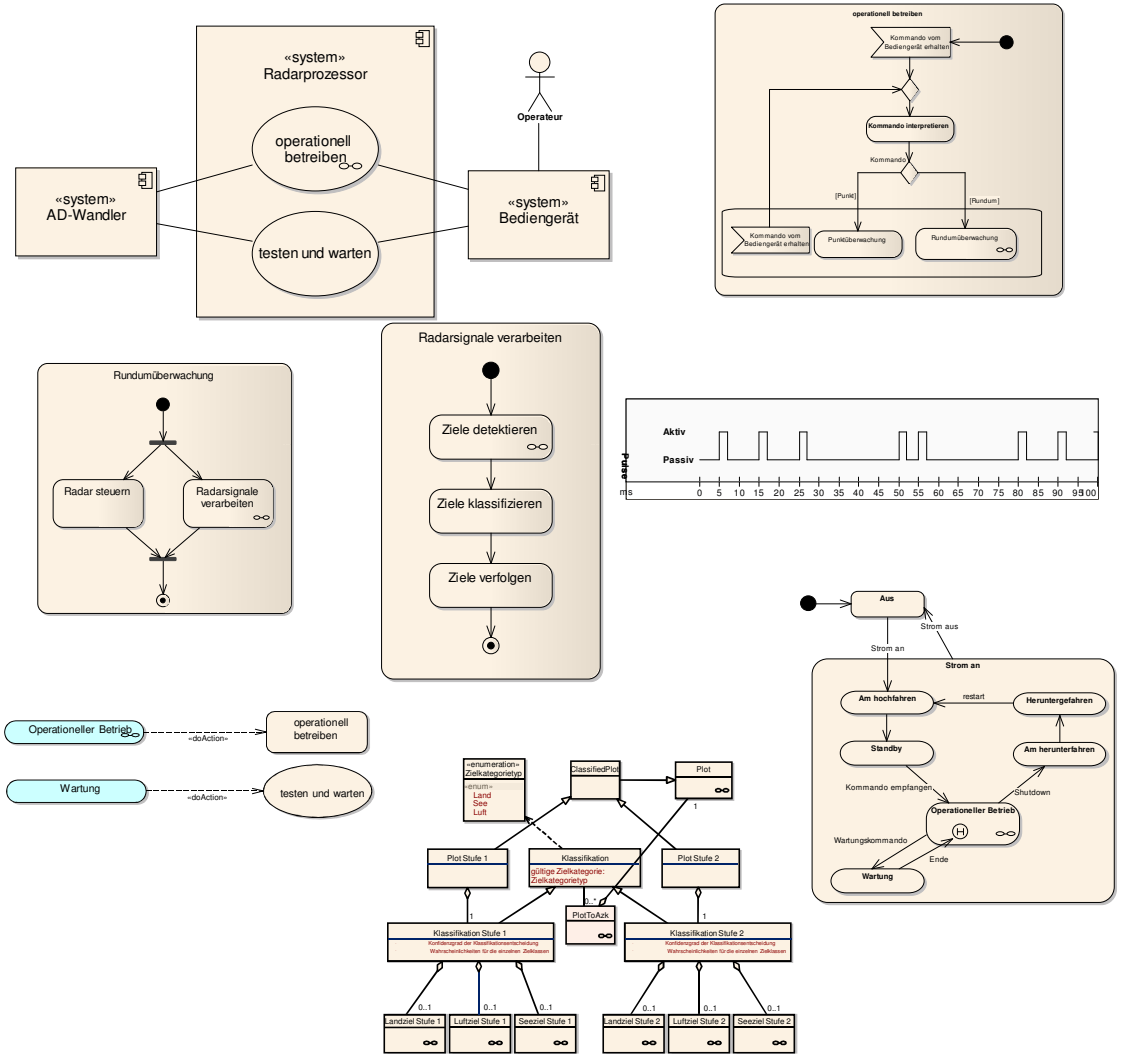
- Parallel zu der Use-Case-Modellierung
- Zustände repräsentieren die Systemzustände
- Übergänge werden durch Kommandos „von außen“ ausgelöst
- Zustände geben Vorbedingungen für Funktionen an



Analysephase

Zusammenfassung Analyse

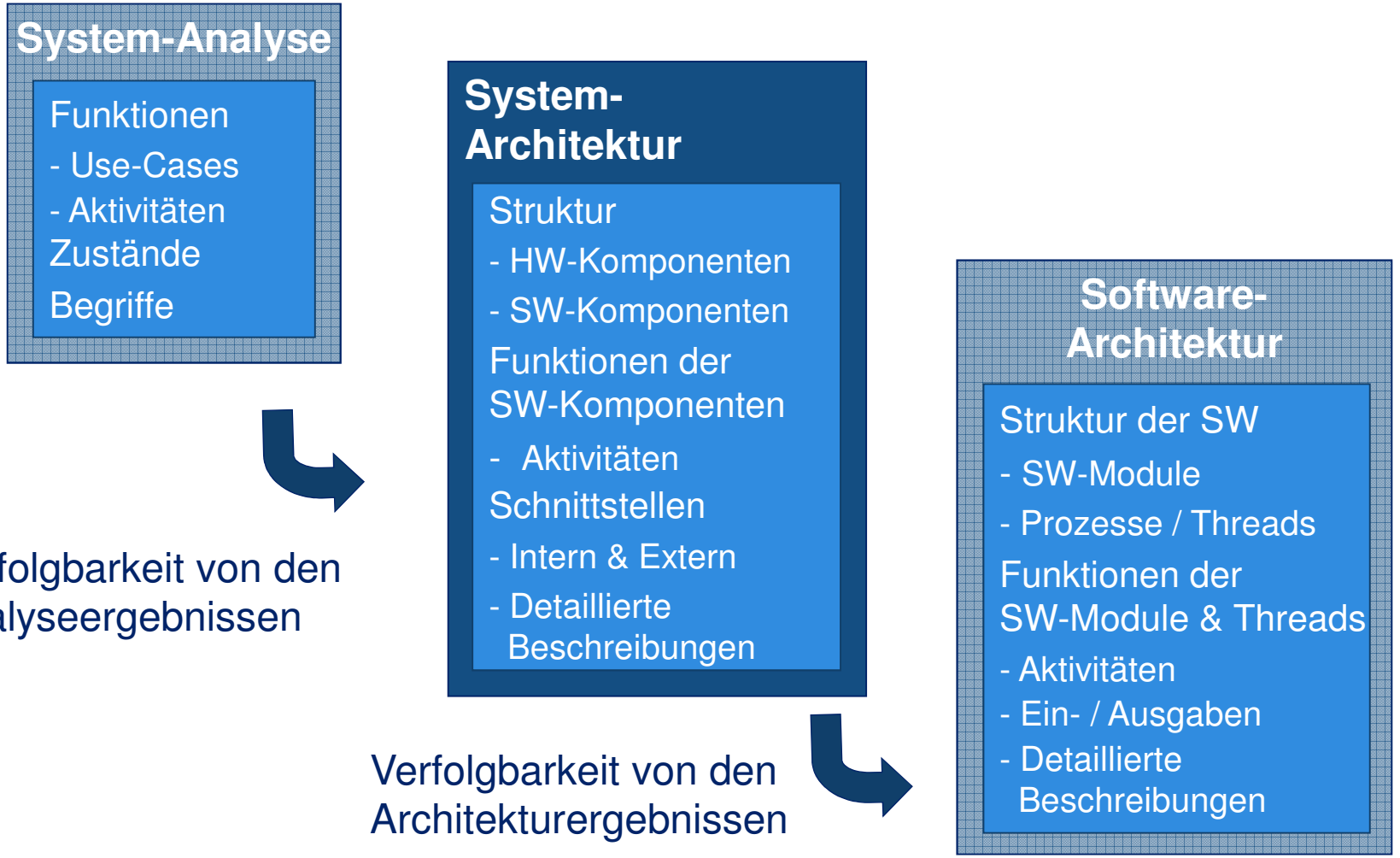
- Klassische Use-Case-Analyse angereichert mit weiteren Diagrammen
- Wir konnten (moderiert) während der inhaltlichen Besprechungen die (Zwischen-) Ergebnisse in UML modellieren



System-Architektur

System- und Softwareentwicklung

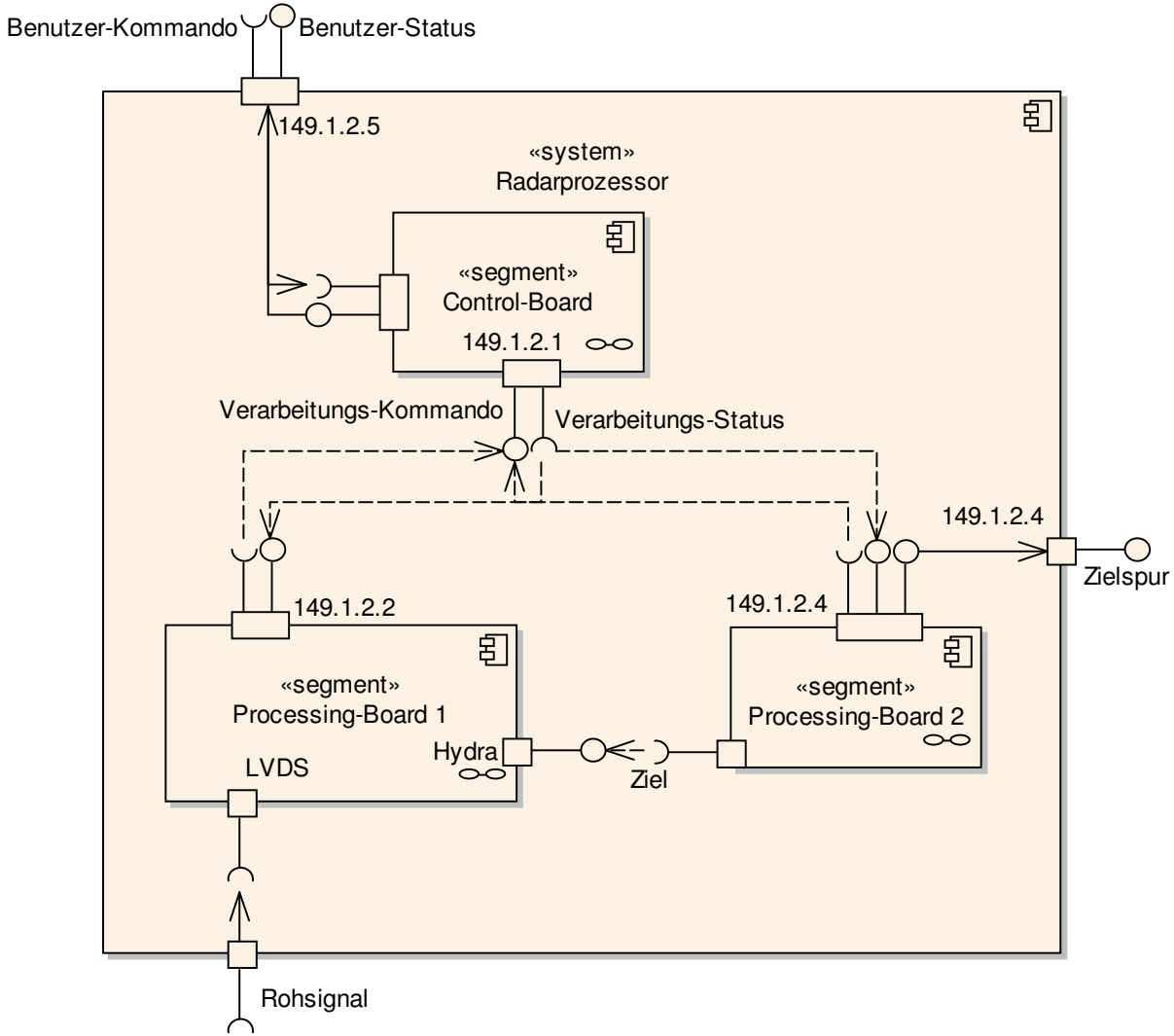
SOPHIST



System-Architektur

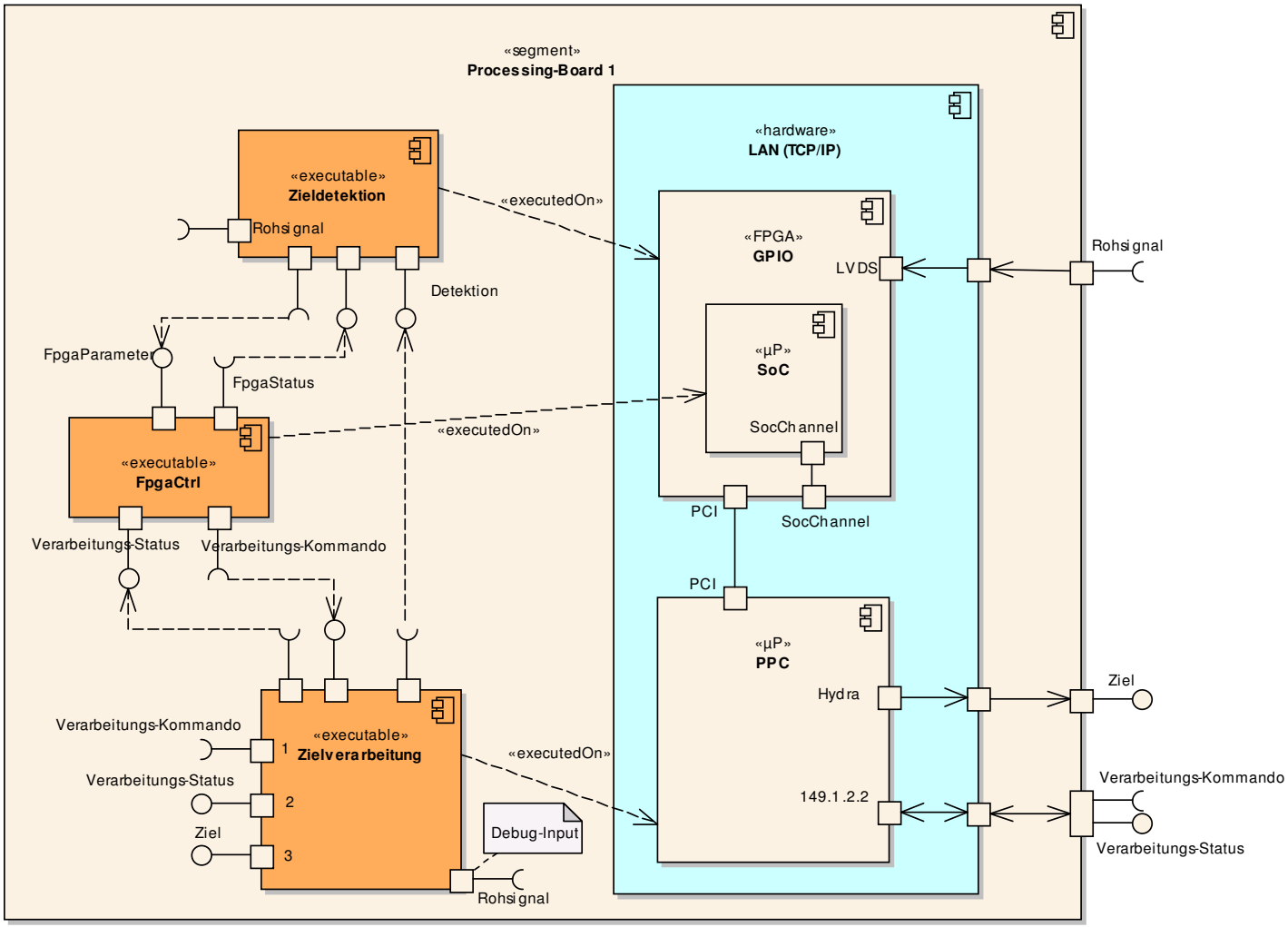
Physikalische Sicht

- Zeigt die Struktur des RAP
- RAP wird verfeinert bis zu reinen Software- und Hardware-Komponenten
- Externe und interne Schnittstellen durch Ports und Interfaces



System-Architektur

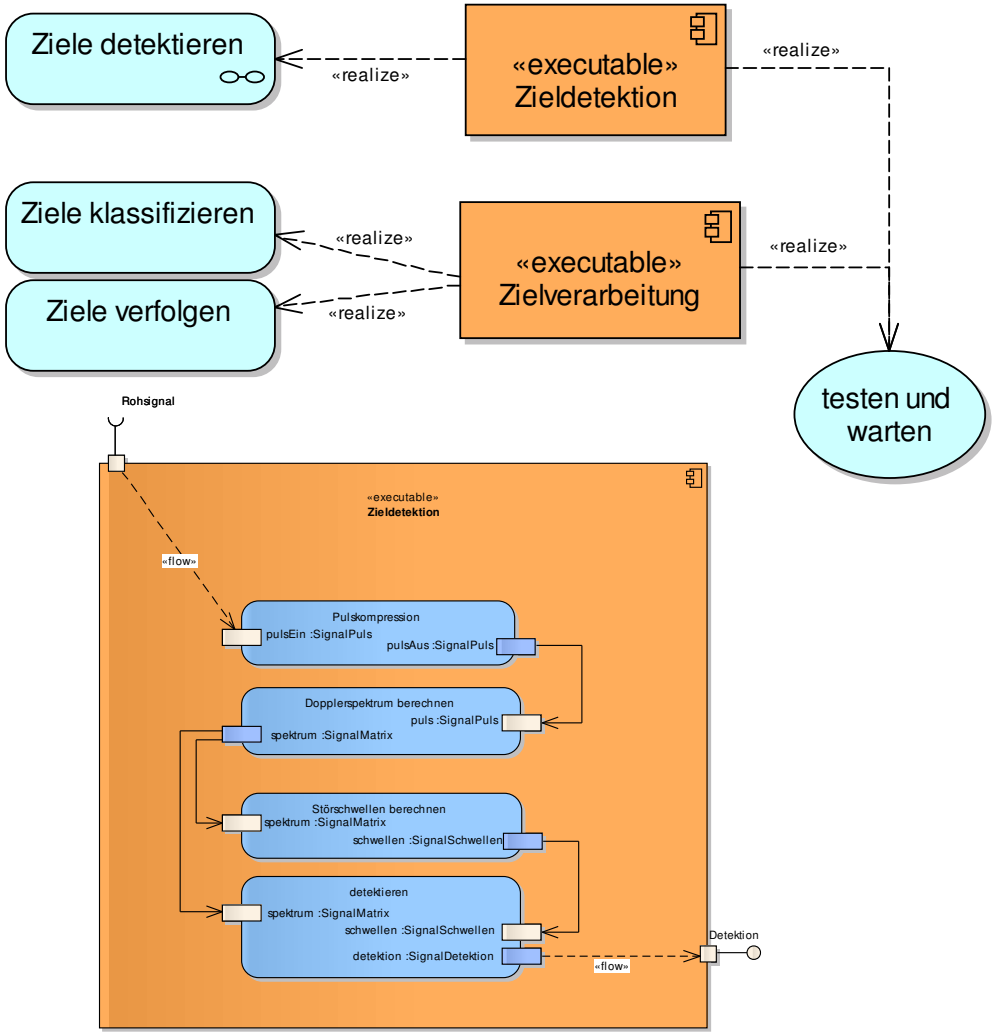
Physikalische Sicht, unterste Ebene



System-Architektur

Funktionen → SW-Komponenten

- Eindeutige Zuordnung aller in der Analyse gefundenen Funktionen (Use-Cases und Aktivitäten) zu Software-Komponenten
- Zuordnung der Schnittstellen der Komponente zu Ein-/ Ausgaben der Aktivitäten
- Verfolgbarkeit der Daten über Komponenten hinweg



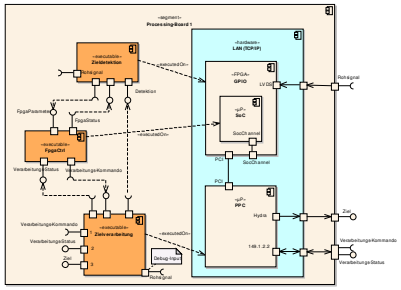
SOPHIST

- Einstieg
- Systematische Evaluation
- Beispiel

Validieren von Designentscheidungen

Einstieg

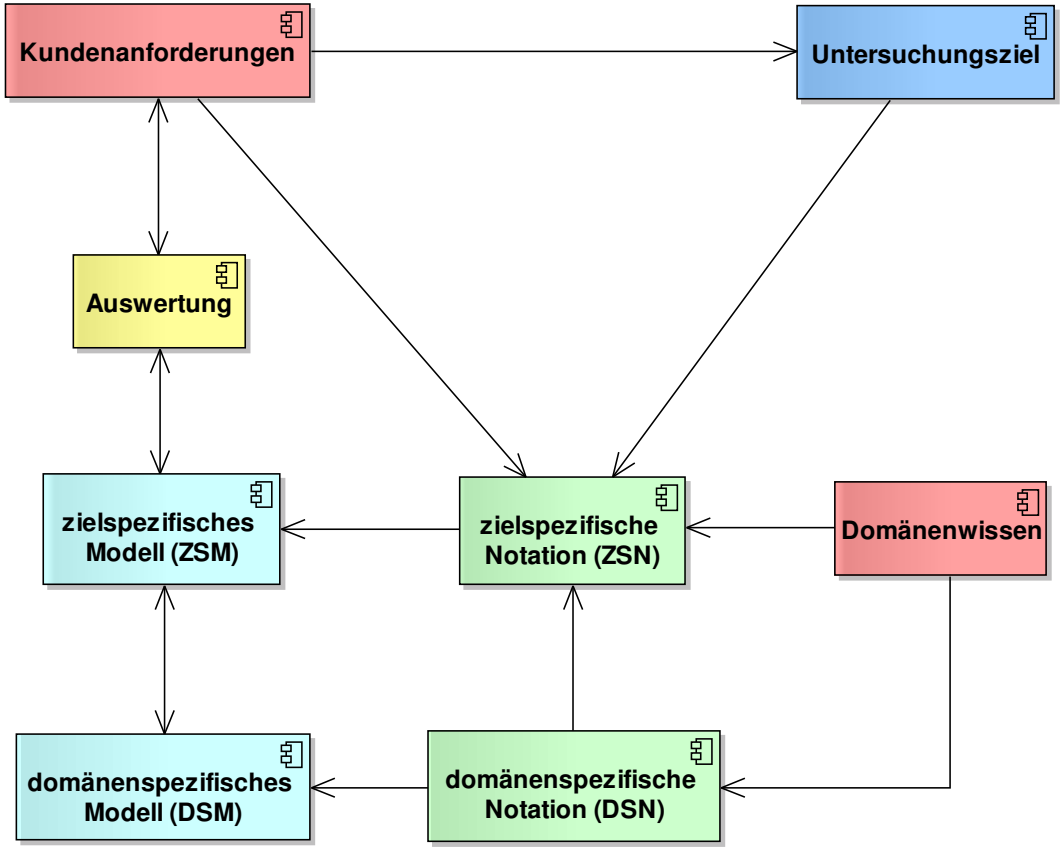
System-Architektur



- Zuordnung der SW-Komponenten auf Hardware-Elemente
- Was muss dabei berücksichtigt werden?
  - Rechenaufwand der einzelnen SW-Komponenten
  - Durch Abhängigkeiten der SW-Komponenten untereinander erzeugter Rechenaufwand
    - Kommunikation
    - Speicher
  - Gewählte Hardware-Architektur
    - Kommunikationsbandbreiten
    - Speichergröße und dessen Anbindung
  - Nicht funktionale Anforderungen
    - Maximale mittlere CPU-Last
    - Maximale Datenverarbeitungsdauer

Systematische Evaluation

Auftretende Daten und ihre Abhängigkeiten



- ZSN hängt vom Untersuchungsziel, dem Domänen-wissen, den Kunden-anforderungen und der DSN ab
- ZSM wird vom DSM, der ZSN und dem Auswertungsergebnis beeinflusst
- DSN und ZSM wirken auf das DSM

Systematische Evaluation

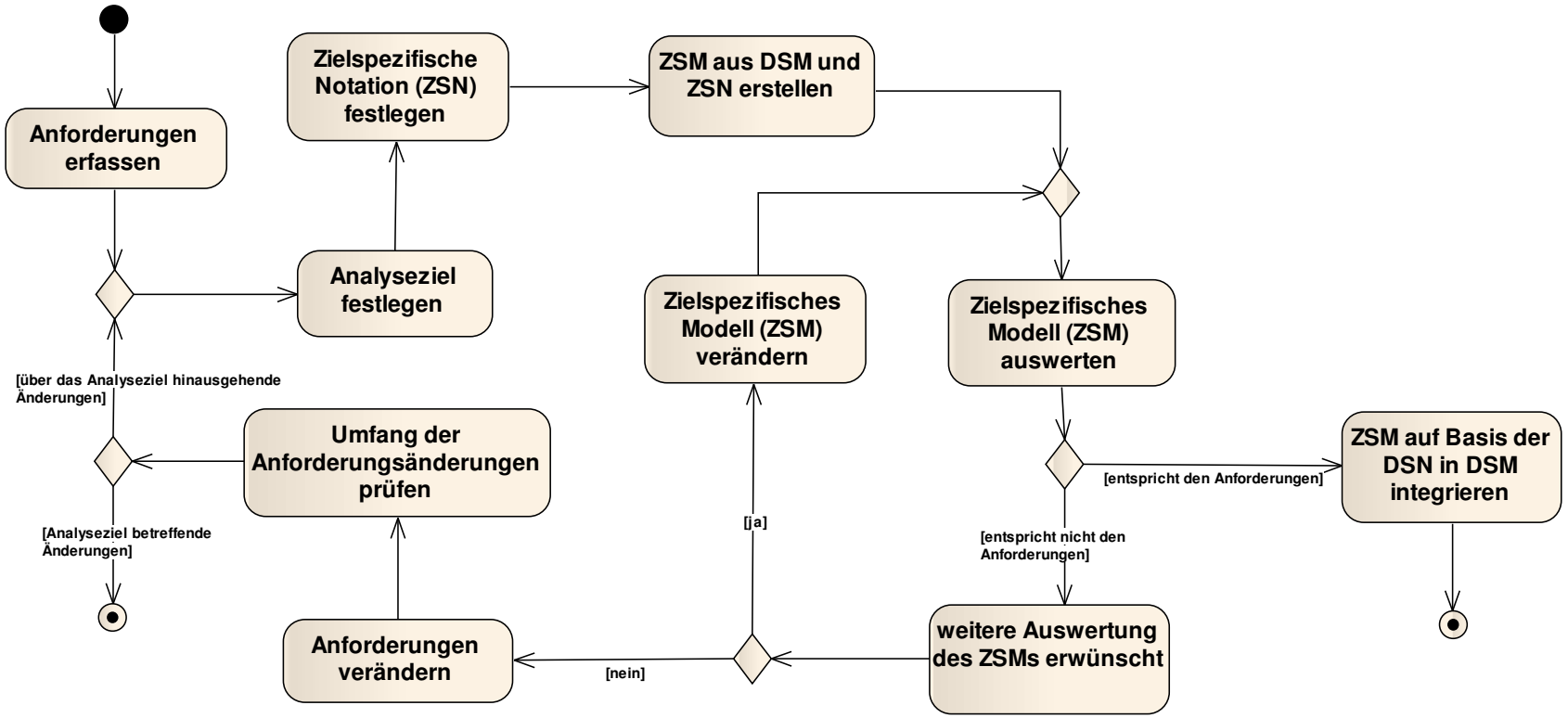
Was steckt hinter den Begriffen…

SOPHIST

- Domänenwissen: embedded systems, Radarsysteme
- Untersuchungsziel: mittlere CPU-Last
- Domänenspezifische Notation: angepasste UML, beschrieben in einem Leitfaden
- Domänenspezifisches Modell: Analyse- und Architekturergebnisse
- Zielspezifische Notation und zielspezifisches Modell: kommen später

Systematische Evaluation

Ablaufschema



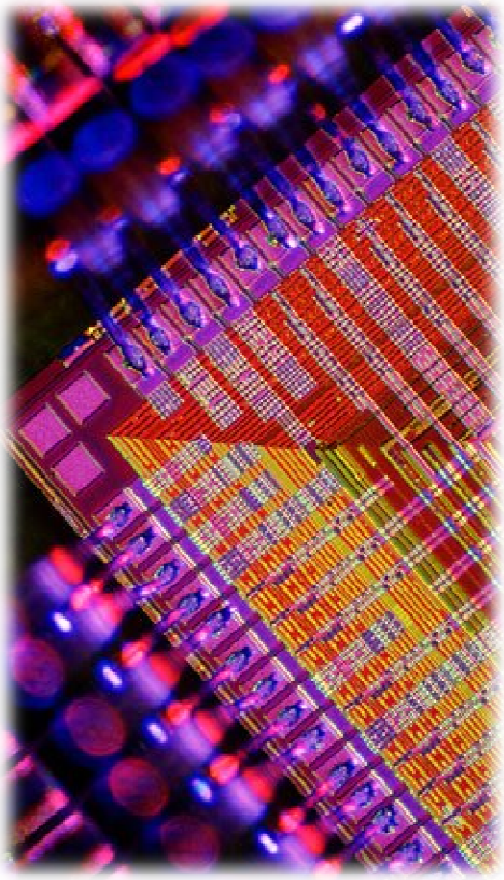
Progressive Verfeinerung von Anforderungen

Nutzen von vorhandenen Informationen im ZSM

Übernahme des ZSMs ins DSM bei positiver Auswertung

Beispiel

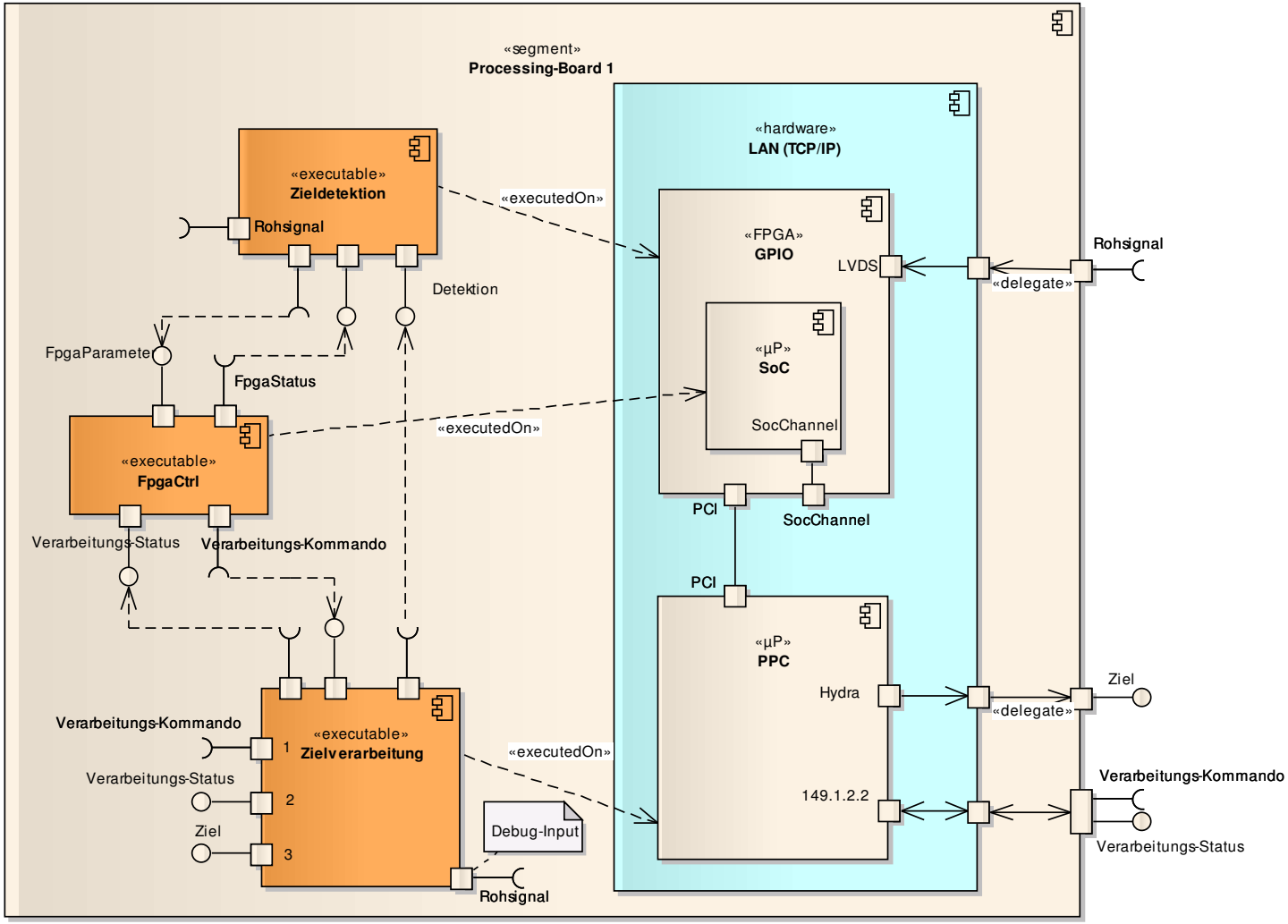
Ermittlung der CPU-Last



- Auswirkungen von SW-Komponenten auf die CPUs einer Hardwareplattform
  - Maßeinheit für CPU-Leistungsfähigkeit und Aufwand der SW-Komponenten: GFlop/s
- Funktionalitäten hängen voneinander ab/kommunizieren miteinander
  - Berücksichtigung des Kommunikationsmediums bei der Ermittlung der CPU-Last
  - Berücksichtigung der Datenaustauschmenge zwischen den Funktionalitäten

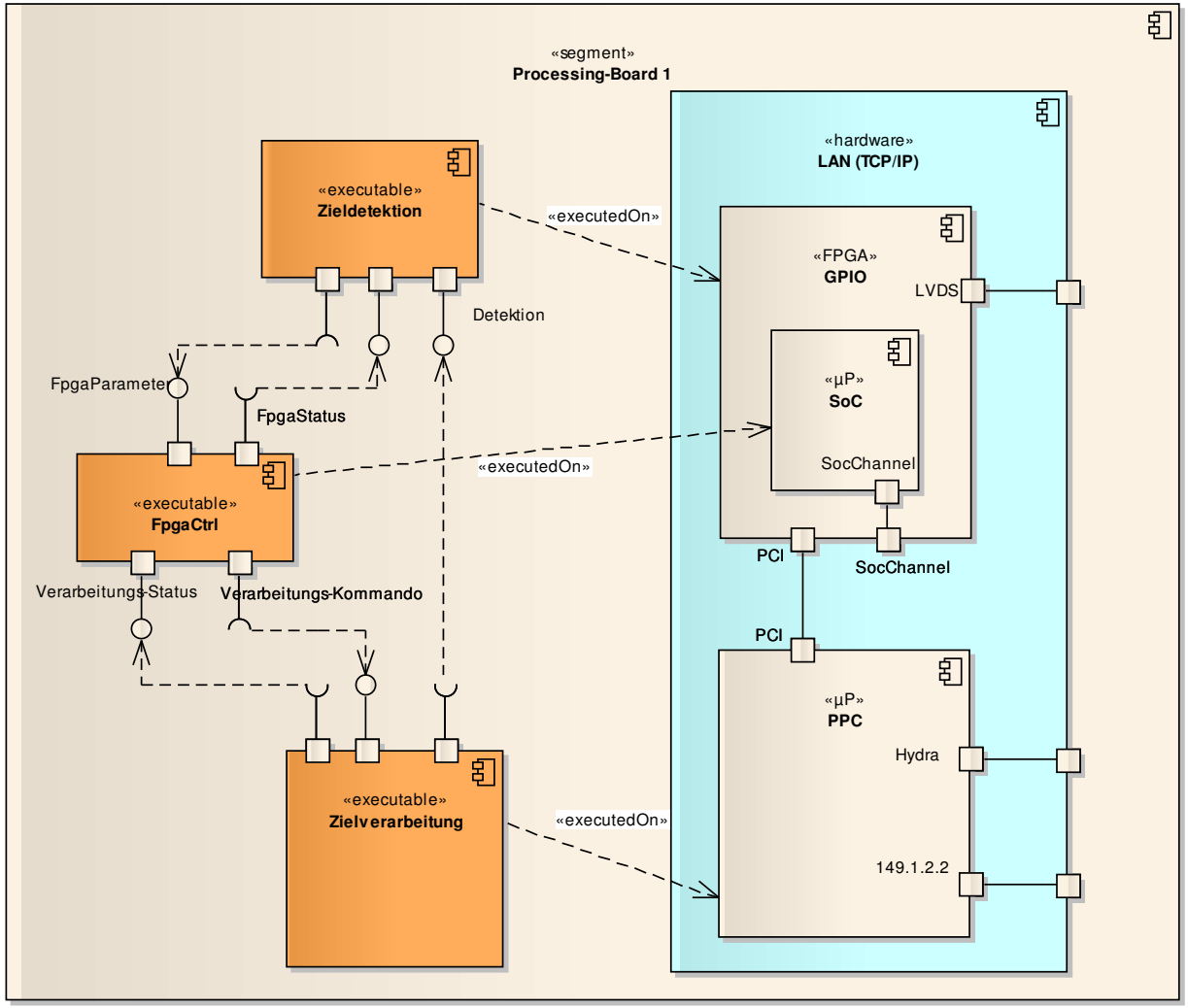
Beispiel

Physikalische Sicht, unterste Ebene



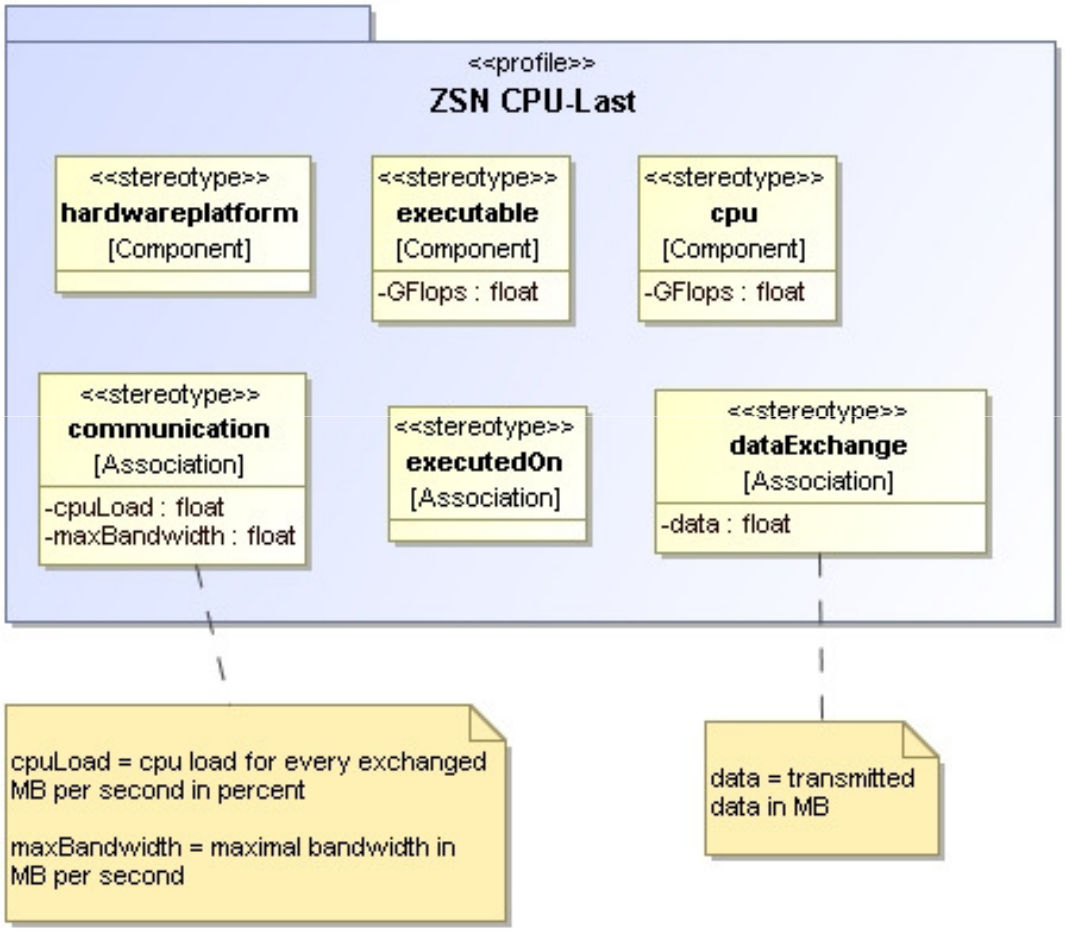
Beispiel

Physikalische Sicht, unterste Ebene



Beispiel

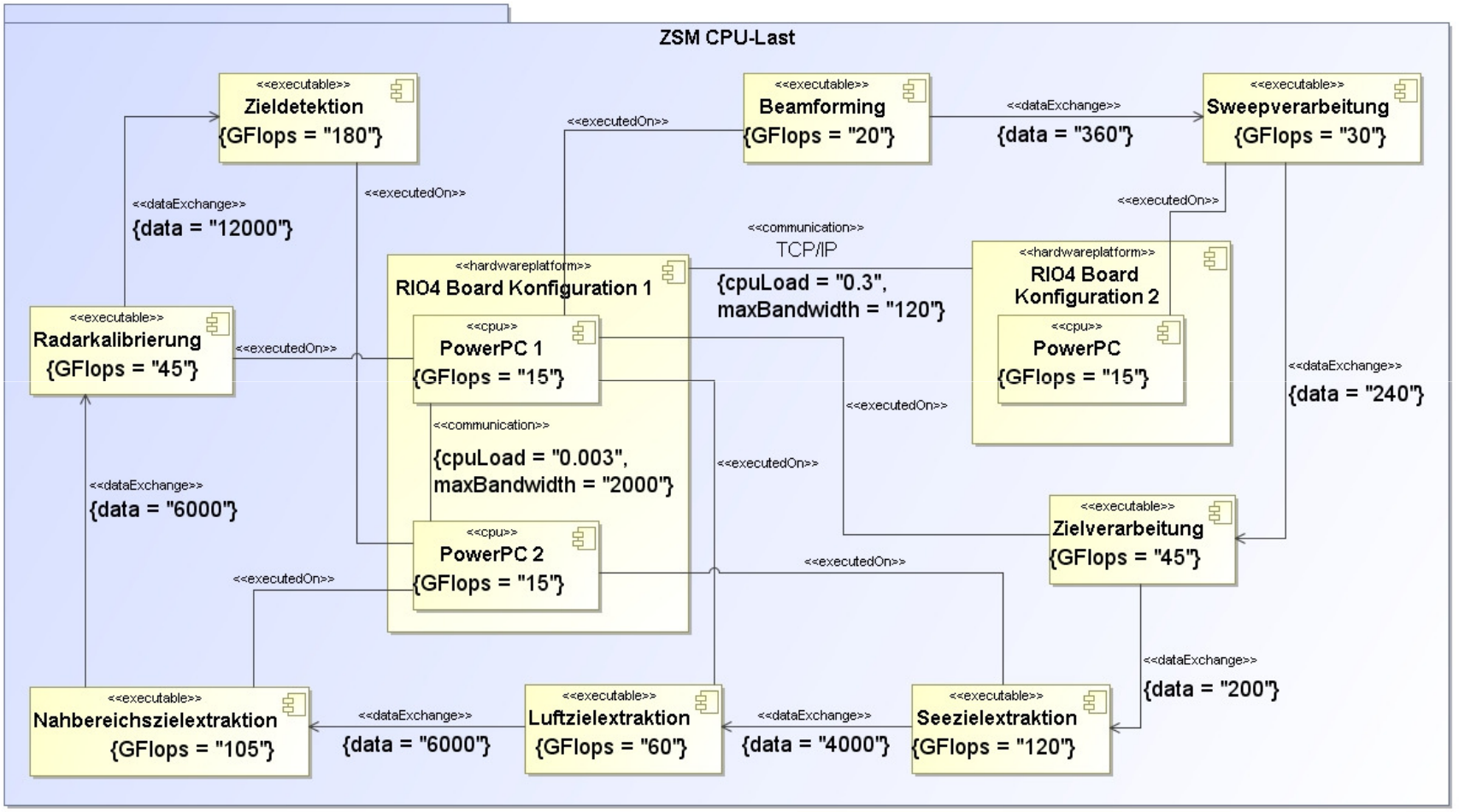
Zielspezifische Notation



- UML wird leichtgewichtig durch ein UML-Profil erweitert
- 6 Stereotypen
- 5 TaggedValues
- Anwendung auf Komponenten und Assoziationen

Beispiel

Zielspezifisches Modell



2 RIO4-Boards, 3 CPUs und 8 Funktionalitäten

Beispiel

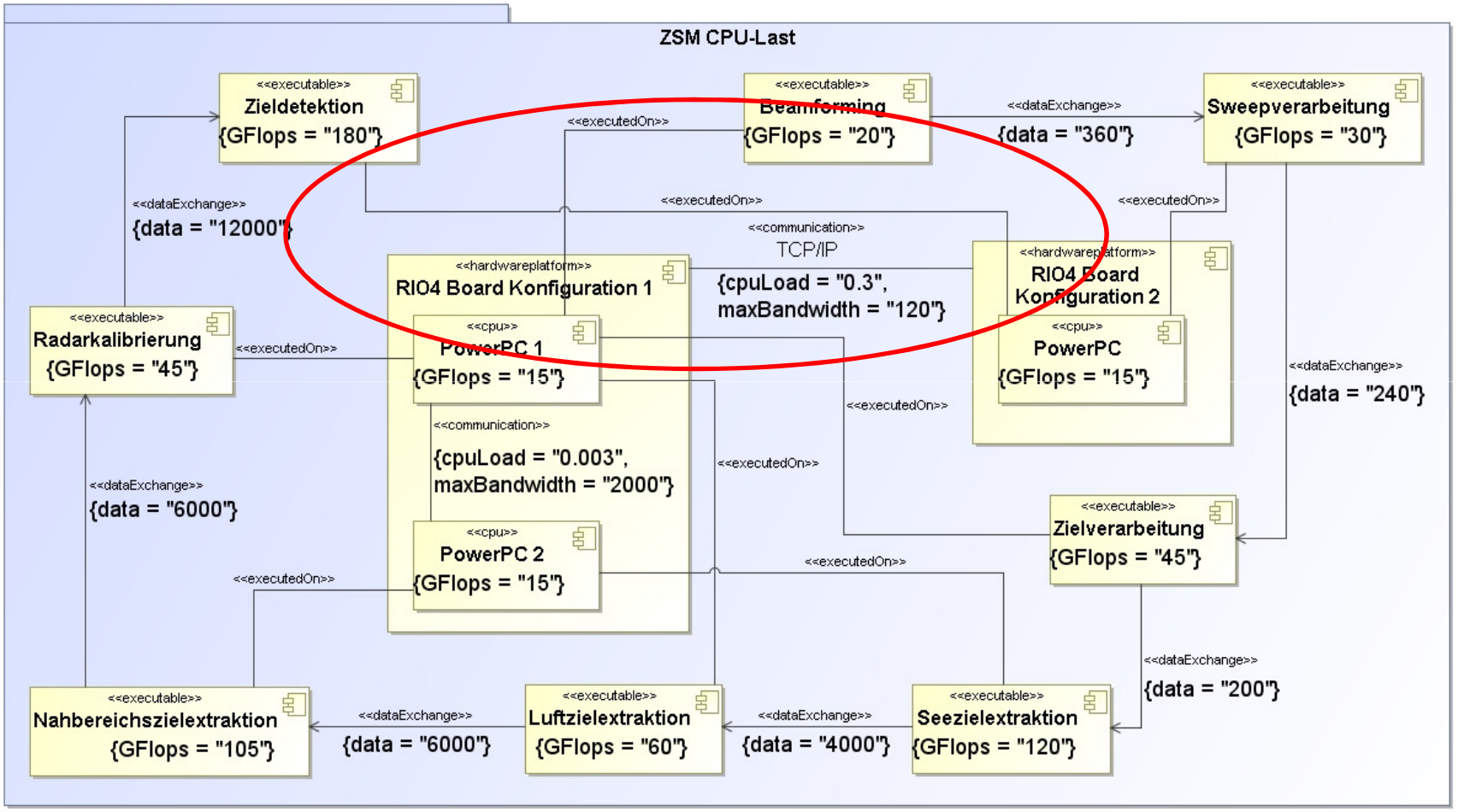
Auswertung

RIO4-Board Konfiguration 1, PowerPC 1			RIO4-Board Konfiguration 1, PowerPC 2			RIO4-Board Konfiguration 2, PowerPC		
Zeit [sec]	CPU Auslastung [%]	Aktion	Zeit [sec]	CPU Auslastung [%]	Aktion	Zeit [sec]	CPU Auslastung [%]	Aktion
5,000	100,00%	Beamforming	15,000	0,00%	idle	5,000	0,00%	idle
3,000	36,00%	DataExchange Beamforming an Sweepverarbeitung	0,100	0,60%	DataExchange Zielverarbeitung an Seezielextraktion	3,000	36,00%	DataExchange Beamforming an Sweepverarbeitung
2,000	0,00%	idle	8,000	100,00%	Seezielextraktion	2,000	100,00%	Sweepverarbeitung
2,000	36,00%	DataExchange Sweepverarbeitung an Zielverarbeitung	2,000	6,00%	DataExchange Seezielextraktion an Luftzielextraktion	2,000	36,00%	DataExchange Sweepverarbeitung an Zielverarbeitung
3,000	100,00%	Zielverarbeitung	4,000	0,00%	idle	58,100	0,00%	idle
0,100	0,60%	DataExchange Zielverarbeitung an Seezielextraktion	3,000	6,00%	DataExchange Luftzielextraktion an Nahbereichsextraktion			
8,000	0,00%	idle	7,000	100,00%	Nahbereichsextraktion			
2,000	6,00%	DataExchange Seezielextraktion an Luftzielextraktion	3,000	6,00%	DataExchange Nahbereichsextraktion an Radarkalibrierung			
4,000	100,00%	Luftzielextraktion	3,000	0,00%	idle			
3,000	6,00%	DataExchange Luftzielextraktion an Nahbereichsextraktion	3,000	6,00%	DataExchange Radarkalibrierung an Zieldetektion			
7,000	0,00%	idle	22,000	100,00%	Zieldetektion			
3,000	6,00%	DataExchange Nahbereichsextraktion an Radarkalibrierung						
3,000	100,00%	Radarkalibrierung						
3,000	6,00%	DataExchange Radarkalibrierung an Zieldetektion						
22,000	0,00%	idle						
70,100	**24,91%**		70,100	**53,72%**		70,100	**5,42%**	

CPU 2 auf HP 1 überschreitet mittlere CPU-Last von 50%

Beispiel

Zielspezifisches Modell - Iteration



Zieldetektion wird auf anderem PowerPC ausgeführt

Beispiel

Auswertung - Iteration

RIO4-Board Konfiguration 1, PowerPC 1			RIO4-Board Konfiguration 1, PowerPC 2			RIO4-Board Konfiguration 2, PowerPC		
Zeit [sec]	CPU Auslastung [%]	Aktion	Zeit [sec]	CPU Auslastung [%]	Aktion	Zeit [sec]	CPU Auslastung [%]	Aktion
5,000	100,00%	Beamforming	15,000	0,00%	idle	5,000	0,00%	idle
3,000	36,00%	DataExchange Beamforming an Sweepverarbeitung	0,100	0,60%	DataExchange Zielverarbeitung an Seezielextraktion	3,000	36,00%	DataExchange Beamforming an Sweepverarbeitung
2,000	0,00%	idle	8,000	100,00%	Seezielextraktion	2,000	100,00%	Sweepverarbeitung
2,000	36,00%	DataExchange Sweepverarbeitung an Zielverarbeitung	2,000	6,00%	DataExchange Seezielextraktion an Luftzielextraktion	2,000	36,00%	DataExchange Sweepverarbeitung an Zielverarbeitung
3,000	100,00%	Zielverarbeitung	4,000	0,00%	idle	33,100	0,00%	idle
0,100	0,60%	DataExchange Zielverarbeitung an Seezielextraktion	3,000	6,00%	DataExchange Luftzielextraktion an Nahbereichsextraktion	50,000	36,00%	DataExchange Radarkalibrierung an Zieldetektion
8,000	0,00%	idle	7,000	100,00%	Nahbereichsextraktion	22,000	100,00%	Zieldetektion
2,000	6,00%	DataExchange Seezielextraktion an Luftzielextraktion	3,000	6,00%	DataExchange Nahbereichsextraktion an Radarkalibrierung			
4,000	100,00%	Luftzielextraktion	75,000	0,00%	idle			
3,000	6,00%	DataExchange Luftzielextraktion an Nahbereichsextraktion						
7,000	0,00%	idle						
3,000	6,00%	DataExchange Nahbereichsextraktion an Radarkalibrierung						
3,000	100,00%	Radarkalibrierung						
50,000	36,00%	DataExchange Radarkalibrierung an Zieldetektion						
22,000	0,00%	idle						
117,100	**30,13%**		117,100	**13,22%**		117,100	**37,40%**	

Anforderung von mittlerer CPU-Last unter 50% wird erfüllt

SOPHIST

- Nutzen und LessonsLearned
- Was wurde nicht gezeigt
- Ausblick

Zusammenfassung und Ausblick

Nutzen und gelernte Vorteile

Teil 1/3

SOPHIST

- Späte Trennung von HW und SW möglich
  - Frühe Abschätzung der benötigten Ressourcen möglich
  - Validierung von Designentscheidungen nach systematischer Evaluation möglicher Ressourcenkombinationen
  - Systemmodellierung bereitet den Weg für HW-SW-Codesign
- Reaktion auf sich ändernde Randbedingungen wird erleichtert
  - Auswirkungen des Einbindens bzw. Wegfalls von Ressourcen können simuliert werden
  - Es ist sofort ersichtlich, wo sich welche Änderungen ergeben

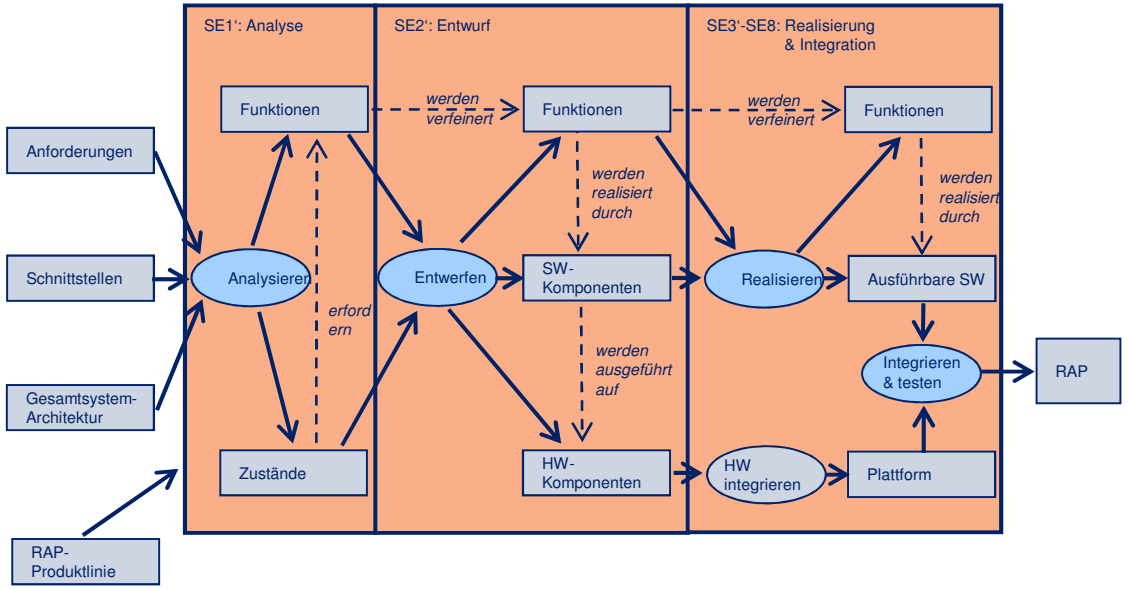
Nutzen und gelernte Vorteile

Teil 2/3

- Besprechungen innerhalb des Radarprozessor-Teams werden nicht mehr auf „PowerPoint-Ebene“ durchgeführt
- Kommunikation mit anderen Teams wird erleichtert
  - Beschreibung der extern zu entwickelnden Teile in UML
  - Definierte Schnittstellen und Funktionalität der Teile

Was haben wir nicht gezeigt

Es gibt noch viel mehr…



- Software-Architektur, Feindesign und Code-Generierung
- Begriffsmodell
- Übergänge zwischen Abteilungen
- Automatische Überprüfung der Modelle

Ausblick

... und woran wollen wir weiter arbeiten?

SOPHIST

Auf der methodischen Ebene

- Modellierung und Verfolgbarkeit zu Testfällen
- Automatisierter Austausch mit Nachbarabteilungen und Übergang zu RM-Tools
- Zweisprachigkeit unterstützen
- Optimale Nutzung der vorhandenen Daten für Validierung
- Systematische Evaluation durch dynamische Abläufe

Operativ im nächsten Projekt

- Domänen-Entwicklung versus Produktlinien-Ansatz untersuchen
- Erweiterung der systematischen Evaluation um weitere Einflussfaktoren

Noch Fragen?

SOPHIST

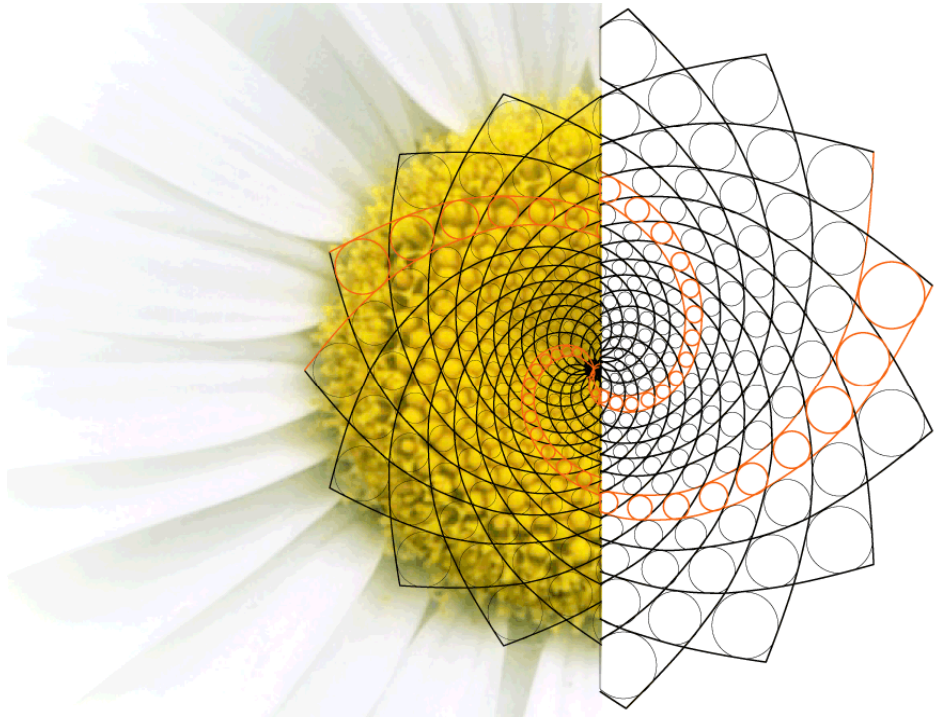
Infos gefällig?

SOPHIST

Die wichtigste Zeit eines Vortrags ist nach dem Vortrag

- Artikel zum Thema
- pdf des Vortrags
- Login in den Downloadbereich
- Newsletter zu OO und RE

Einfach Visitenkarte
abgeben oder Mail an
heureka@sophist.de
Schicken.



Wir erkennen die Struktur Ihrer Projektanforderungen!

Infos unter www.SOPHIST.de