Architektur-Entwicklung

- Architectural Tactics für die Architekturentwicklung

- Attribute-Driven Design (ADD)
- ~~Domain-Specific Software Architecture (PuLSE-DSSA)~~
- Architecture-Based Design (ABD)
- Software-Kategorien
- ARC42

- Literatur

Idee fürs Design-Vorgehen

Architektur-
Entwicklung

- Tactics
- ADD
- PuLSE-DSSA
- ABD
- Kategorien
- arc42
- Literatur
- ...
- ...
- ...
- ...
- ...

Man lässt sich von den Qualitätsanforderungen leiten und wählt geeignete *Tactics* aus (oder Stile/Pattern, welche die gefragten *Tactics* unterstützen)

Attribute-Driven Design (ADD); Idee

- "Architektur-relevante" Anforderungen (Funktionalität und Qualität) werden als (einfache) Szenarien dargestellt
- Iteration über die Szenarien
  - Qualitätsanforderungen werden für die Auswahl geeigneter Tactics/Architekturstile benutzt
  - Funktionale Anforderungen werden benutzt, um die Komponententypen dieser Stile zu "instanziieren"

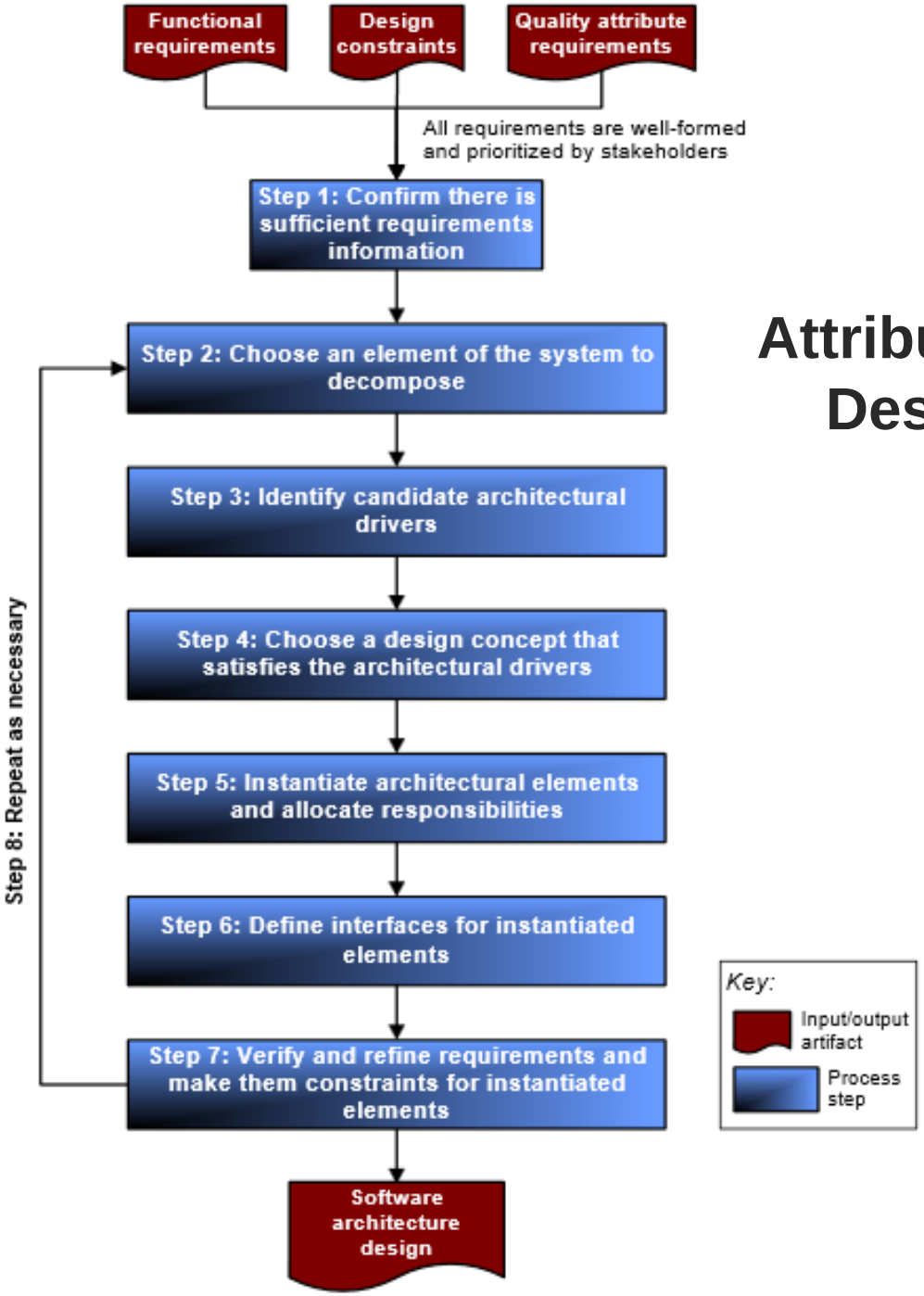
  In jedem Iterationsschritt wird die Architektur Top-Down verfeinert

- Genauer: nächste Folie

Architektur-
Entwicklung

- Tactics
- ADD
- PuLSE-DSSA
- ABD
- Kategorien
- arc42
- Literatur
- ...
- ...
- ...
- ...
- ...

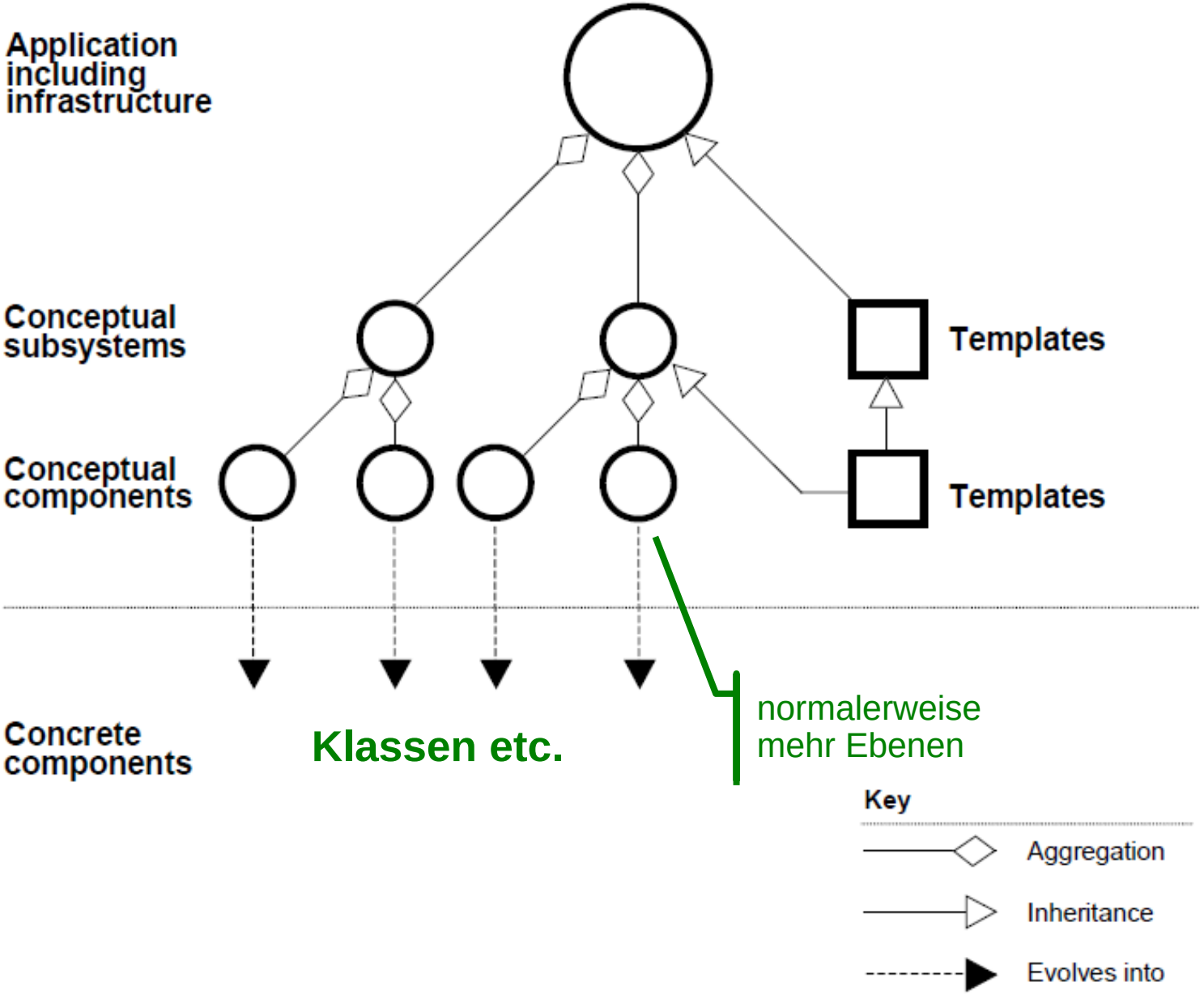
Attribute-Driven Design (ADD)



Architecture-Based Design (ABD)[BBC+2000]; Idee

- Input
  - Abstrakte und konkrete funktionale Requirements
  - Abstrakte und konkrete Qualitäts- und Business- Requirements
  - Constraints
- Vorgehen: rekursive Dekomposition/Verfeinerung
  1. Qualitäts- und Business- Requirements: Identifikation eines geeigneten Stils
  2. Funktionale Requirements: Zuweisen der bisherigen Funktionalität an die neuen Elements (oder deren Vater)
  3. Darstellung der verfeinerten Architektur aus drei Sichten: logical, concurrency, deployment view
     → führt eventuell zu neuen Aufgaben für die (neuen) Komponenten
  4. Templates einsetzen und bei Bedarf erweitern/ergänzen
  5. Validierung der Dekomposition mittels der konkreten Requirements
  6. Überprüfung der Constraints

Dekomposition eines Systems



Kategorien ("Software-Blutgruppen")

- Idee
  - Trennen von Zuständigkeiten
  - *Separation of concerns*
- Vorgehen
  1. Kategorien bilden
  2. Komponenten darin einsortieren

  Ziel:

  Komponenten einfach definieren, so dass sie jeweils genau einer Kategorie angehören
- Einsatz: Bewertung verschiedener Alternativen

Komplexitätsmaß für den *Vergleich mehrerer Architekturen*

Vorarbeit

- Komplexität einer *reinen Kategorie* := Anzahl der Vorgänger im Kategoriegraph
- Komplexität einer *unreinen Kategorie* := Summe der Komplexitäten der zusammengemischten Kategorien

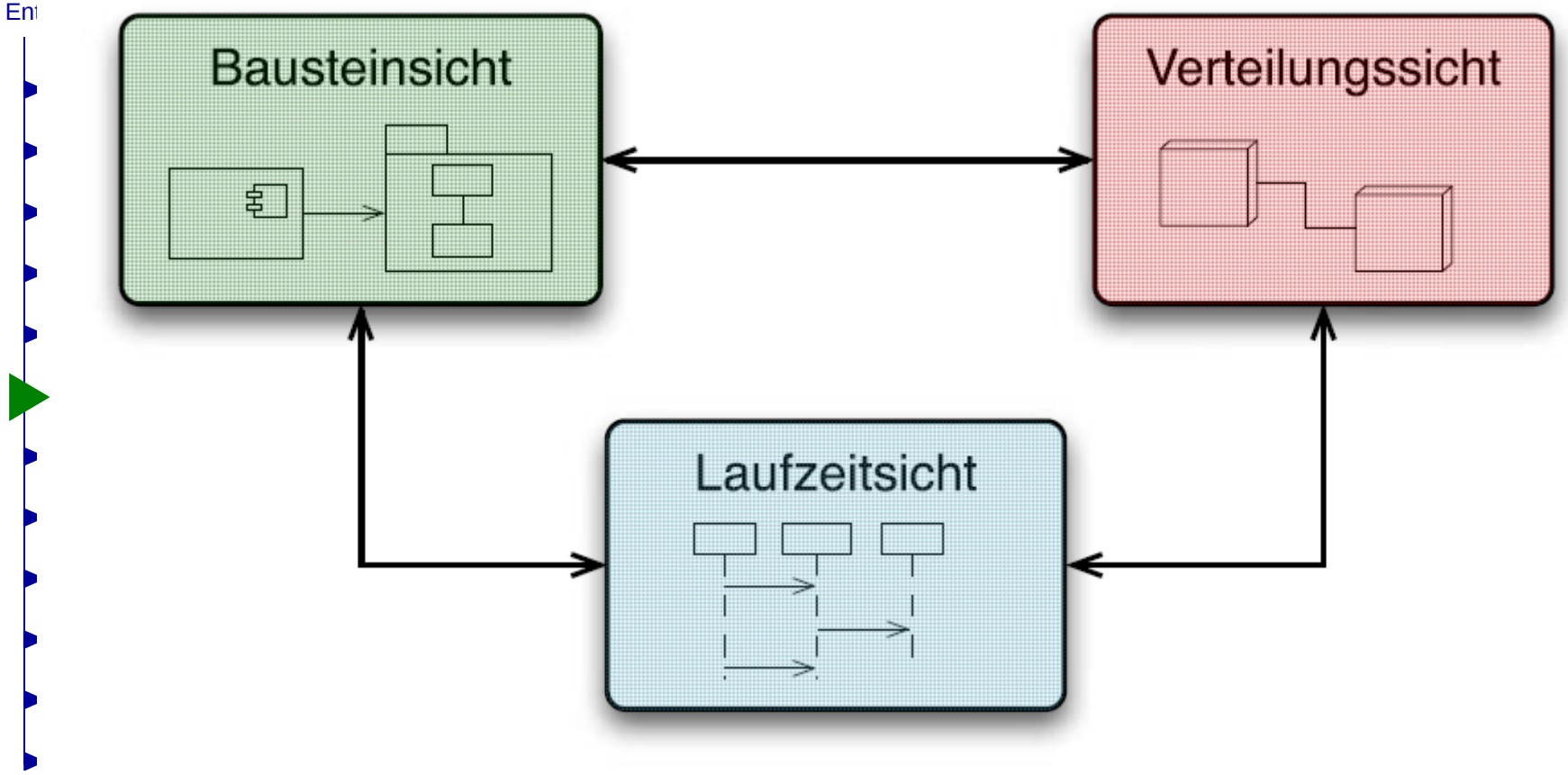
Komplexitätsbestimmung

- Komplexität einer *einfachen Komponente* := Komplexität ihrer Kategorie
- Mischende Komponenten sollen vermieden werden
- Komplexität einer *zusammengesetzten Komponente* (oder eines *Systems*) := gewichtetes (z.B. LoC) Mittel der Komplexitäten der Subkomponenten

Kommunikation zwischen Komponenten

- Hohe Kategorien sehen nur solche, die sie verfeinern
- Zwei Kategorien dürfen nur über gemeinsame Vorfahren kommunizieren
  - Eine Komponente K einer Kategorie a darf eine (Komponente H oder Schnittstelle S einer) Kategorie b genau dann importieren oder exportieren, wenn a eine Verfeinerung von b ist.
  - K und H sind Komponenten der Kategorien a und b; sie dürfen nur über eine Schnittstelle S der Kategorie c kommunizieren, wenn a und b c verfeinern.
- Eine Komponente, die eine andere aus einer nicht-höheren Kategorie direkt aufruft, erwirbt zusätzlich deren Kategorie(n)

Sichten nach Starke/Hruschka (1)

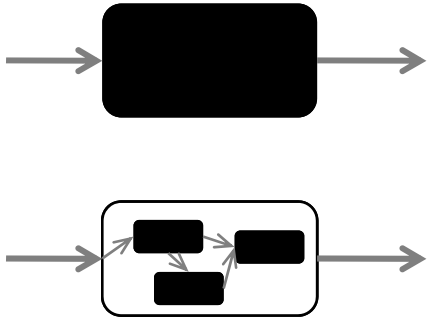


Sichten nach Starke/Hruschka (2)

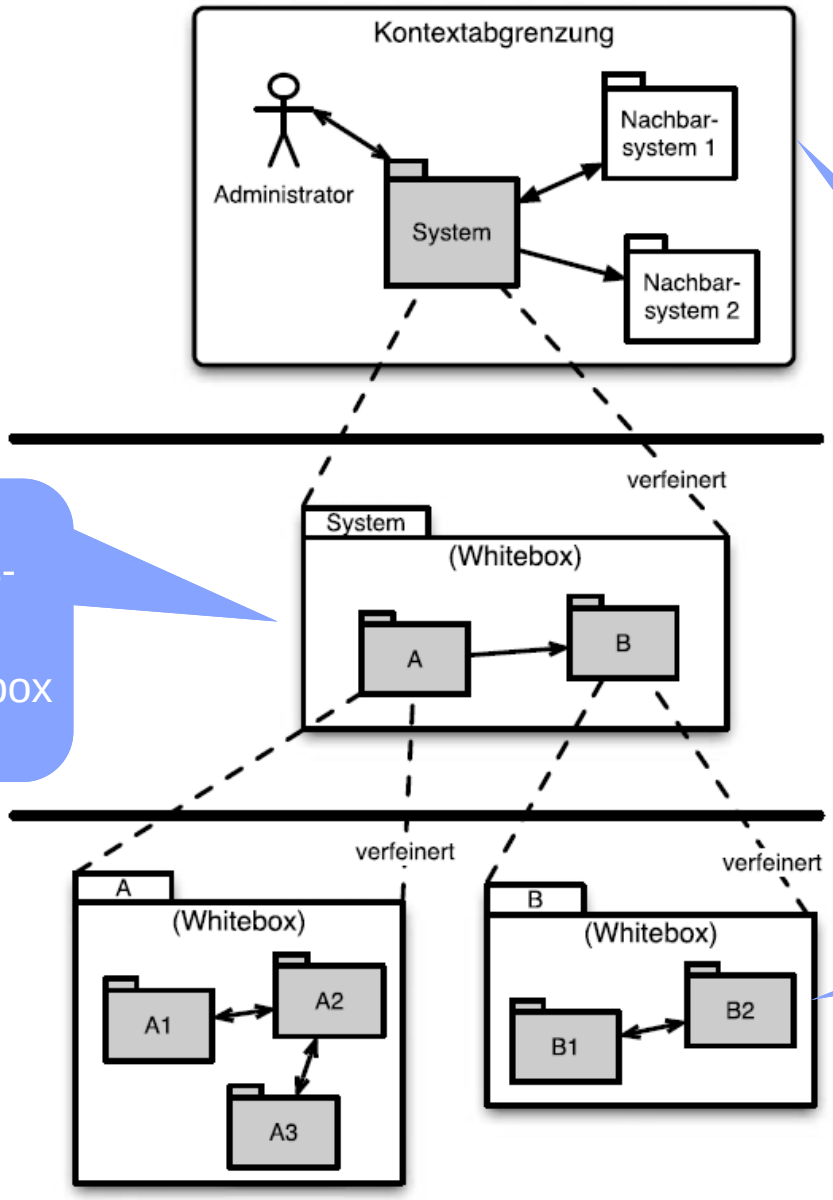
- Bausteinsicht
  - wie ist das System aus Softwarebausteinen aufgebaut (statische Struktur)
  - deren Beziehungen und Schnittstellen untereinander
- Laufzeitsicht
  - wie interagieren die Bausteine des Systems zur Laufzeit miteinander bzw. mit den Nachbarsystemen (dynamische Struktur)
  - wesentliche, beispielhafte Abläufe
- Verteilungssicht
  - welche Teile des Systems laufen auf welchen Rechnern
  - an welchen geographischen Standorten
  - in welcher Umgebung

Bausteinsicht (1)

- Baustein - Blackbox
  - nur Schnittstellen und Funktionen
- Baustein - Whitebox
  - zeigt innere Struktur
- Bausteinsicht
  - hierarchische Verfeinerung des Systems
  - Wurzel: Kontextdiagramm
  - Tiefe
    - → abhängig von Erfordernissen



Bausteinsicht (2)



aus der Anforderungs-spezifikation

System als Blackbox

1. Verfeinerungs-ebene

System als Whitebox

2. Verfeinerungs-ebene

Bausteinsicht (3)

- Diagramm
  - Paketdiagramm
  - Komponentendiagramm
- Verfeinerung
  - Ebene 0: Kontextdiagramm
    - → Kommunikation mit externen Komponenten
  - Ebene 1: system-intern
    - → Zerlegung des Systems in Komponenten
  - Ebene 2: Zerlegung der Komponenten aus Ebene 1
  - …

Bausteinsicht (3)

Blackbox-Beschreibung eines Bausteins

Bezeichnung	Bedeutung
Name	Wie heißt diese Blackbox?
Zweck & Verantwort-lichkeit	Welche Verantwortung übernimmt der Baustein in der Gesamtlösung?
Schnittstelle(n)	Welche Schnittstellen hat der Baustein?
(optional) Abhängig-keiten	Wovon ist dieser Baustein abhängig?
Erfüllte Anforderungen	Verweise zu den Anforderungen, die dieser Baustein erfüllt.
Variabilität	Was kann sich an diesem Baustein ändern?
Code-Artefakte	Verweise auf Quellcode-Artefakte.
Weitere Informationen	Beispielsweise Autor, Versions- und Änderungs-information.
(optional) Offene Punkte	

Bausteinsicht (4)

- Tactics
- ADD
- PuLSE-D
- ABD
- Kategorie
- arc42
- Literatur
- ...
- ...
- ...
- ...
- ...

Whitebox-Beschreibung eines Bausteins

Bezeichnung	Bedeutung
Whitebox zu <<Name>>	Bezeichnung der zu detaillierenden Blackbox
Übersichtsdiagramm	Klassen-, Paket- oder Komponentendiagramm
Lokale Bausteine	Liste aller lokalen (Blackbox-)Bausteine
Lokale Beziehungen und Abhängigkeiten	Liste der lokalen Abhängigkeiten, d. h. Schnittstellen zwischen lokalen Blackboxes oder zu äußeren Bausteinen
Entwurfsentscheidungen	Welche Entscheidungen haben zu dieser Struktur geführt?
(optional) Verworfene Entwurfsalternativen	Welche Alternativen gab es?
(optional) Verweise	Verweise auf zusätzliche Informationen
(optional) Offene Punkte	

Diagramm!

eingesetzte Arch.-Muster

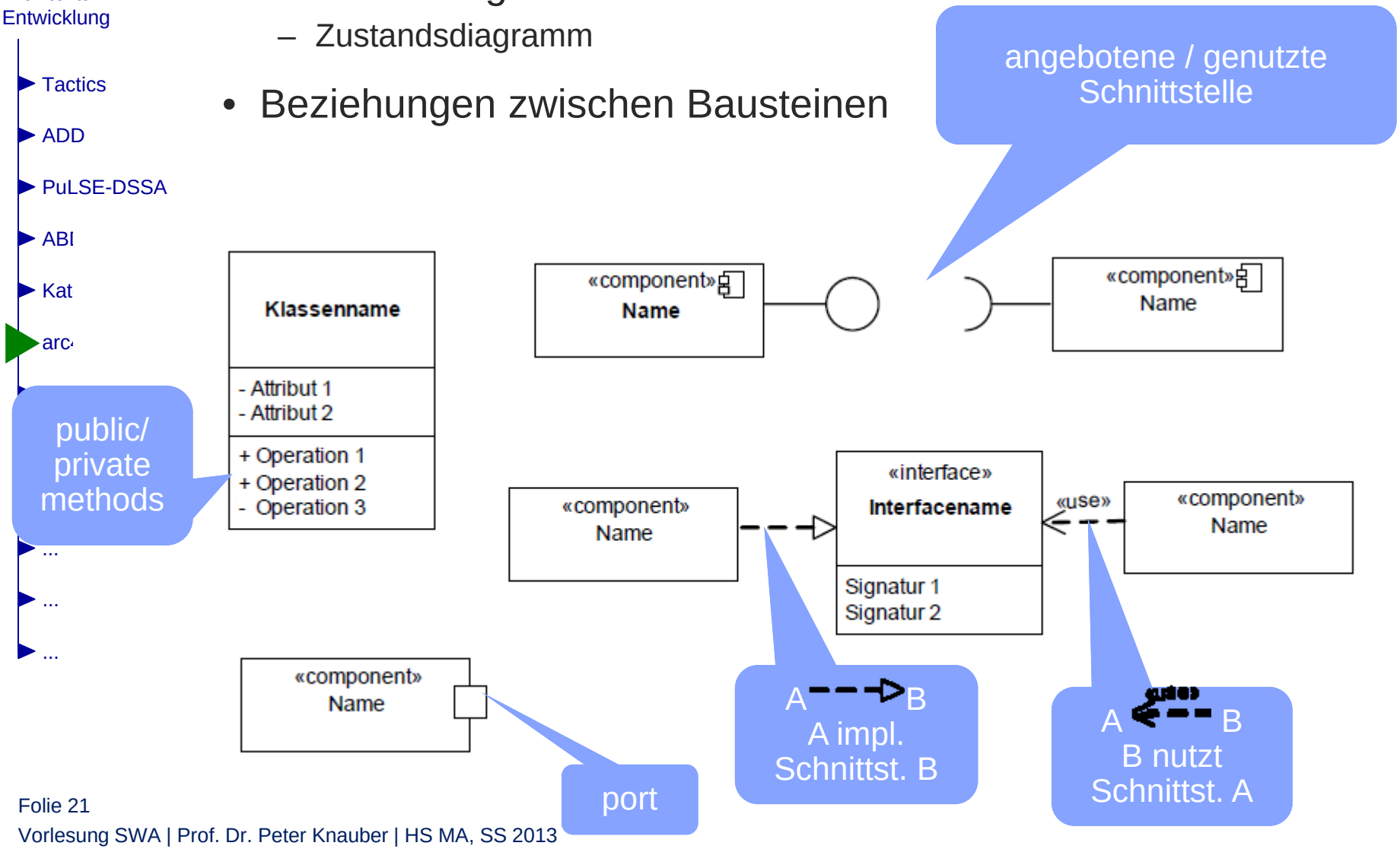
Bausteinsicht (5)

Beschreibung Schnittstelle

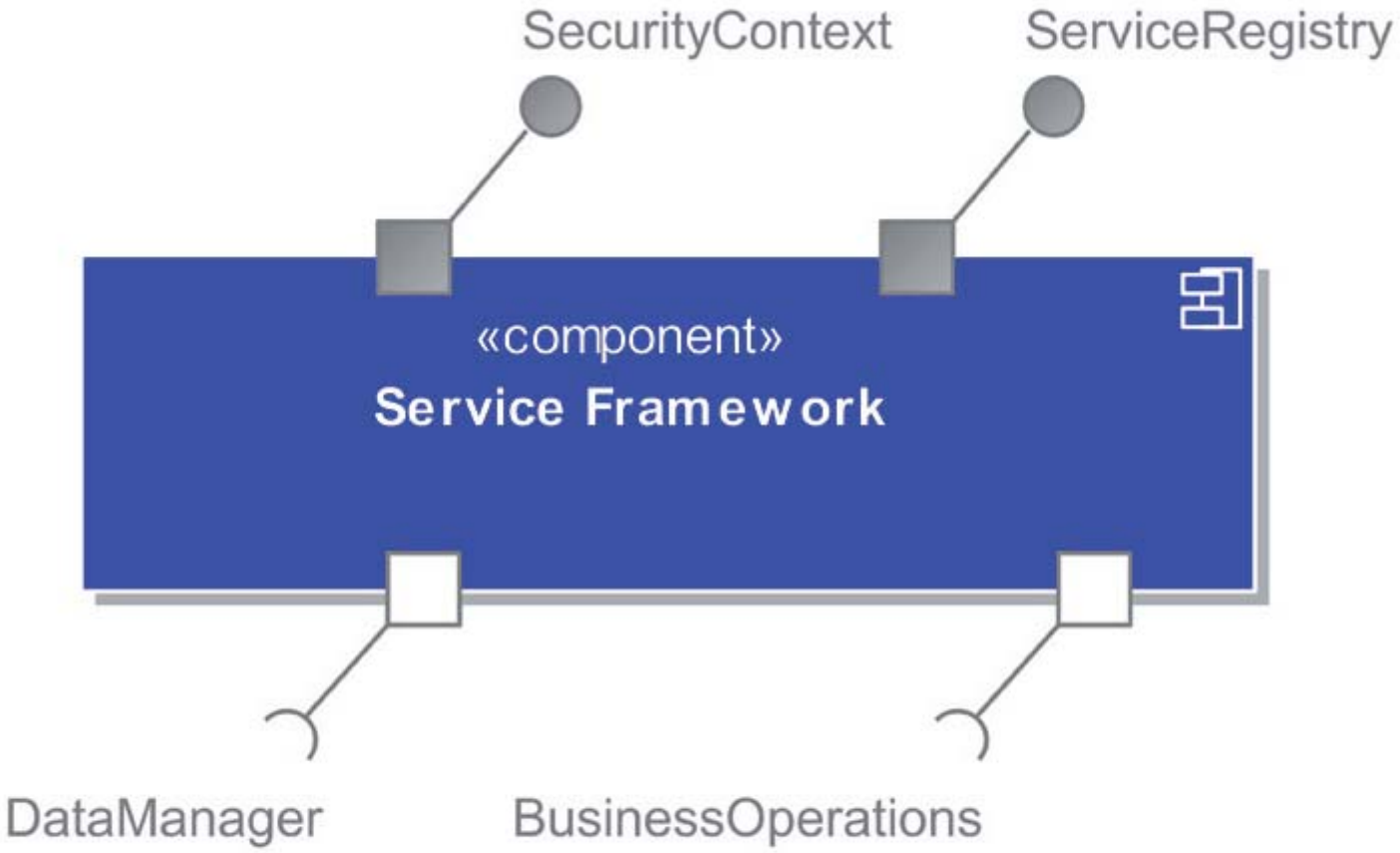
Bezeichnung	Bedeutung
Identifikation	Genaue Bezeichnung (inklusive Version) der Schnittstelle
Sender/Auslöser	Welcher Baustein löst die Übertragung der Ressource aus?
Empfänger	Welcher Baustein empfängt die Ressource?
Ressource	Welche Ressourcen überträgt diese Schnittstelle?
Kanal/Medium	Über welches Medium oder welche Kanäle findet die Übertragung der Ressource physisch statt?
Ablauf	Welche einzelnen Schritte sind zur Übertragung der Ressource nötig?
Fehlerszenarien	Welche Fehler können hier auftreten, und wie werden sie behandelt?
Variabilität	Konfigurationsparameter der Schnittstelle
Qualitätsmerkmale	Welche Quality-of-Service-Anforderungen erfüllt die Schnittstelle?
Benutzungshinweise	Hinweise oder Beispiele zur Benutzung dieser Schnittstelle

Bausteinsicht (6)

- Beschreibung eines Bausteins
  - Zustandsdiagramm
- Beziehungen zwischen Bausteinen

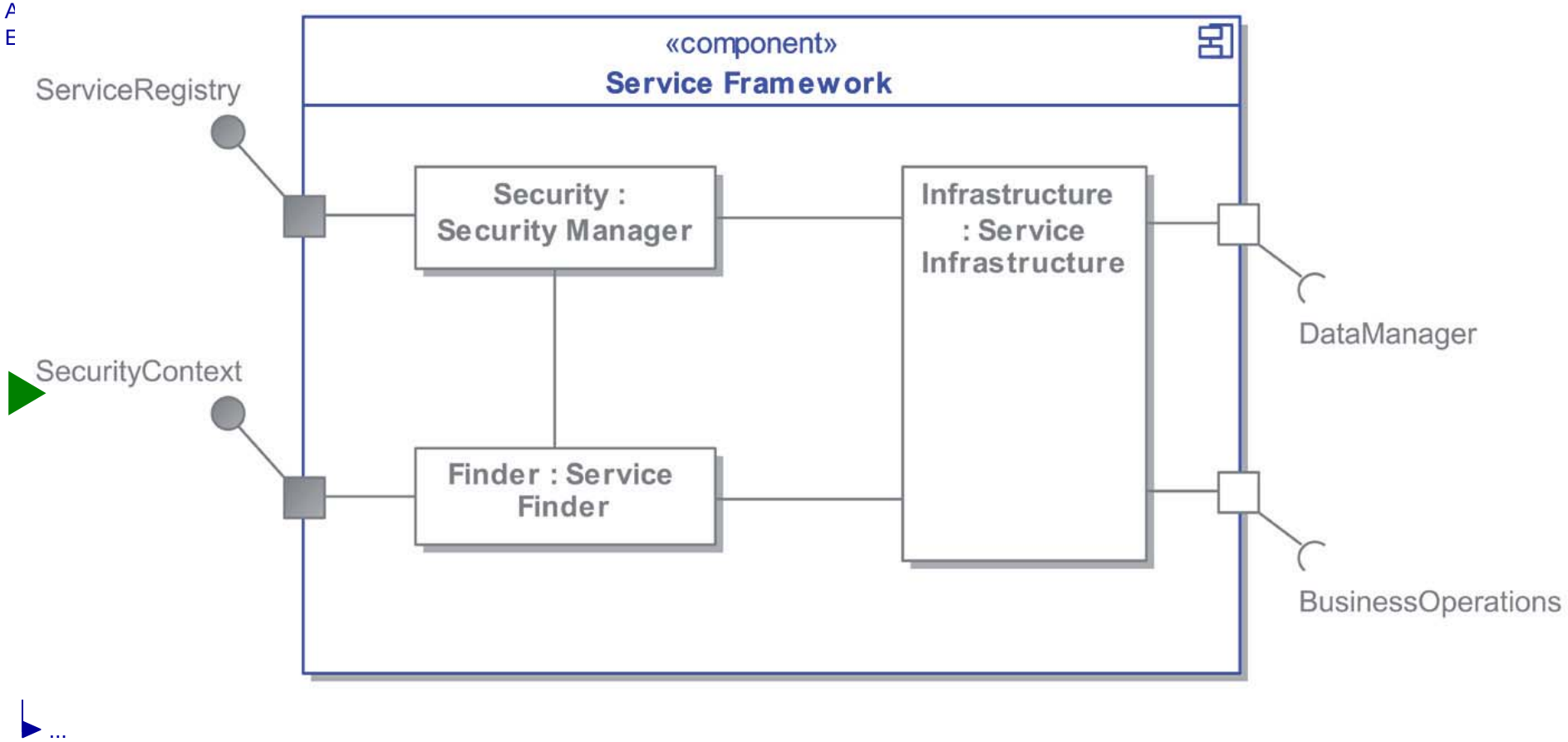


Beispiel Bausteinsicht: Service-Framework – Blackbox

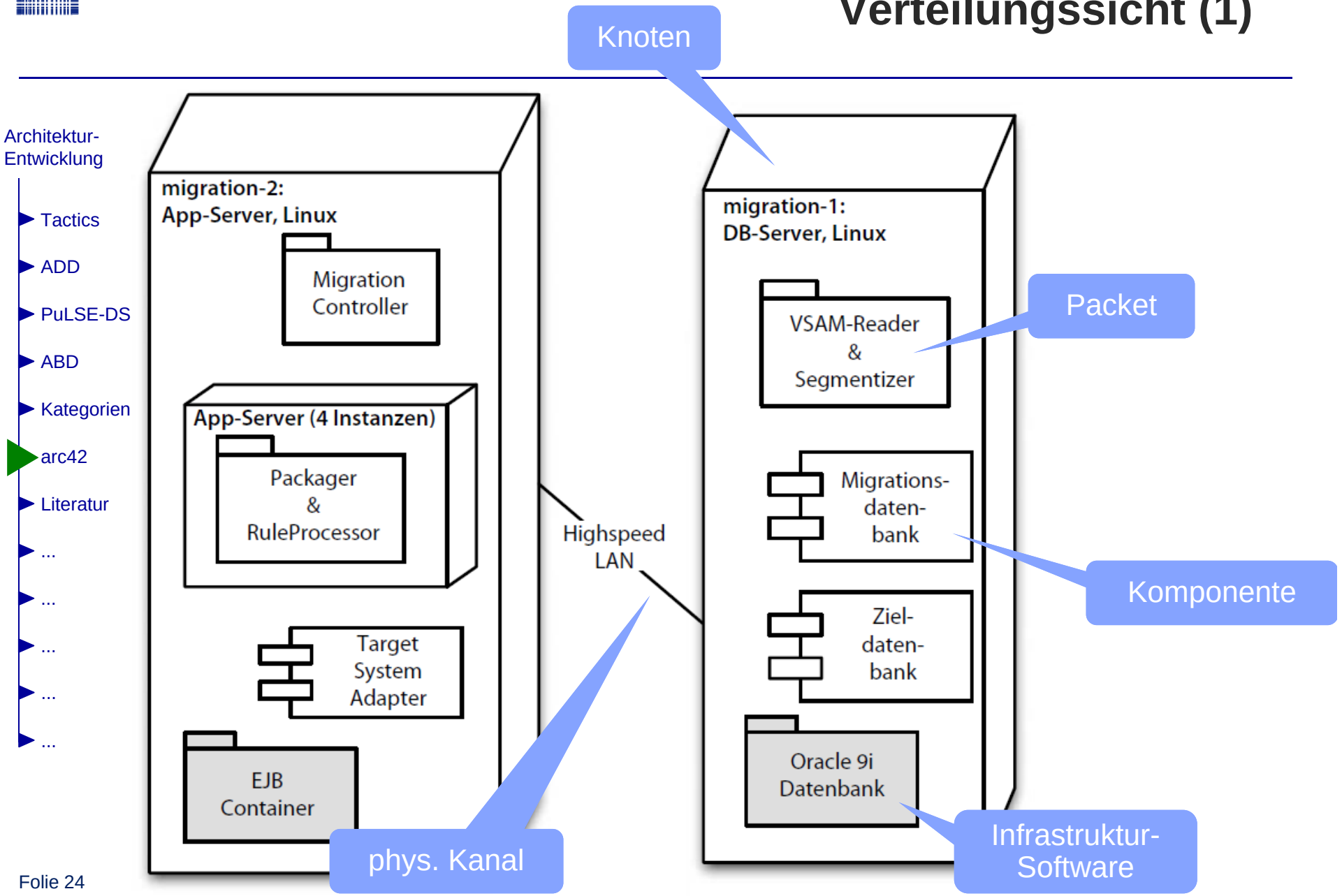


Darius Silingas: Using Simple Architecture Models to Coordinate Scrum Teams,
in ObjektSpektrum Online Architekturen/2010
http://www.sigs-datacom.de/fachzeitschriften/objektspektrum/online-themenspecials/artikelansicht.html?show=2876

Beispiel Bausteinsicht: Service-Framework – Whitebox



Verteilungssicht (1)



Verteilungssicht (2)

- Diagramm
  - Deployment-Diagramm
- Verfeinerung
  - Ebene 0: technisches Kontextdiagramm
    - → Kommunikation mit externen Systemen (Deployment-Diagramm mit System = ein Knoten)
  - Ebene 1: System-intern
    - → Hardware-Komponenten mit zugehörigen Komponenten
    - → Alternativen möglich Bsp.
      - variable Anzahl von Servern
      - DB und Server zusammen/verteilt

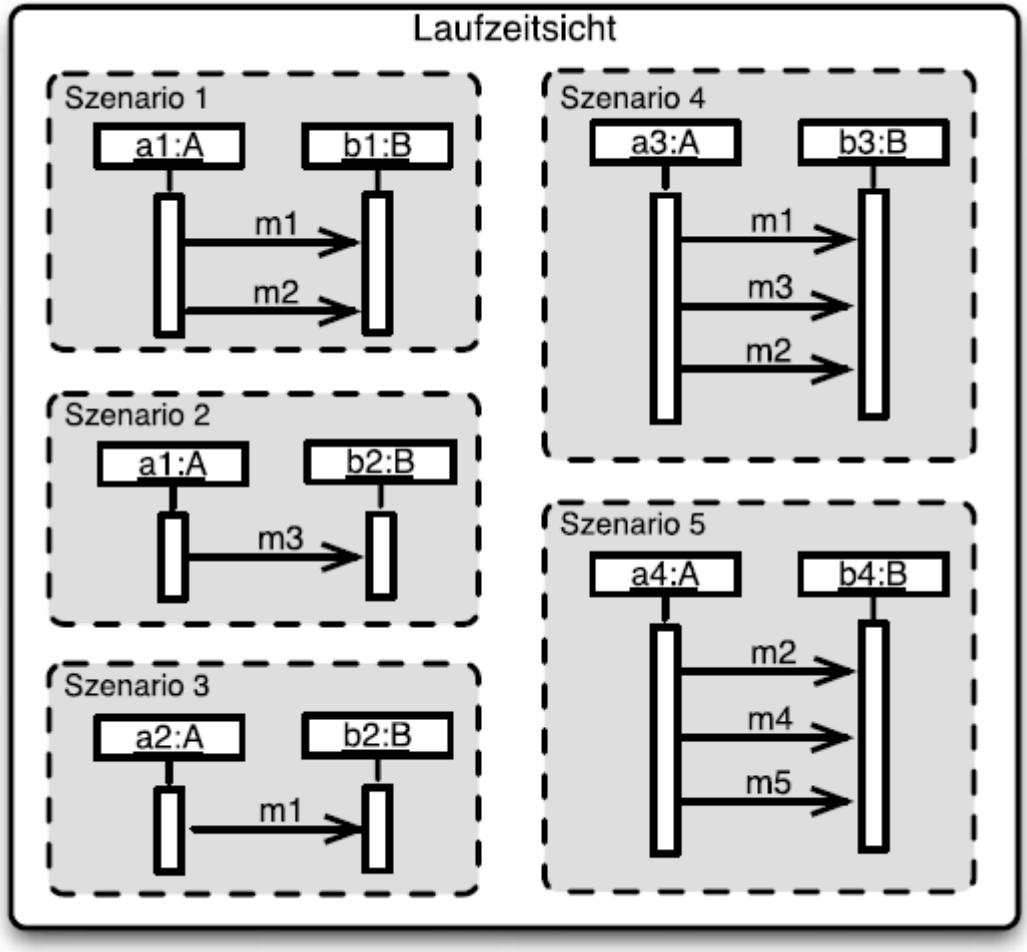
„Komponenten“ sind immer Software-Komponenten

Verteilungssicht (3)

Beschreibung Knoten

Element	Beschreibung
Name	Name des Knotens oder des Knoten-Typs
Beschreibung/Motivation	Kurze Beschreibung oder Kommentar, warum dieser Prozessor, dieser Chip oder diese Speichereinheit ausgewählt wurde; warum Sie als Architekt davon ausgehen, dass dieser Knoten die Leistung erbringen kann, die ihm softwaremäßig auferlegt wird.
Leistungsmerkmale	Technische Leistungs- oder Ausstattungsmerkmale. Hierzu zählen beispielsweise CPU, Haupt- oder Plattenspeicher.
Zugeordnete Verteilungsartefakte	Der entscheidende Teil der jeweiligen Verteilungsvariante: Welche Verteilungs- oder Installationsartefakte werden auf diesem Knoten installiert und ausgeführt?
Sonstige Verwaltungsinformationen	Preis- und Beschaffungsinformationen, Hinweis auf Standort, Hersteller oder sonstige organisatorische Infos.
Offene Punkte	Offene Fragen, Probleme oder Risiken im Zusammenhang mit diesem Knoten.

Laufzeitsicht (1)



Darstellung ausgewählter Szenarien

- typische Abläufe
- komplexe Abläufe
- unklare Abläufe

Laufzeitsicht (2)

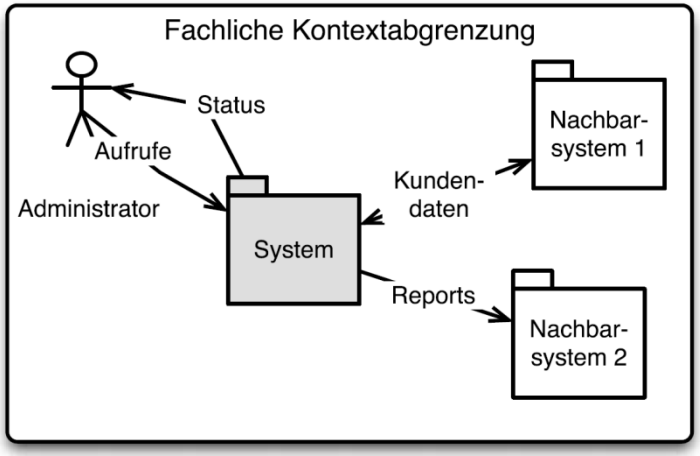
- Diagramm
  - Sequenzdiagramm
  - Alternativ: Kommunikationsdiagramm
- Verfeinerung
  - Ebene 0: Kontextdiagramm
    - → Kommunikation mit externen Systemen
  - Ebene 1: System-intern
    - → 1. Darstellung
      - Kommunikation zwischen Hardware-Komponenten (notwendig: Verteilungssicht)
    - → 2. Darstellung
      - Kommunikation zwischen Komponenten
  - Ebene 2 …
  - …

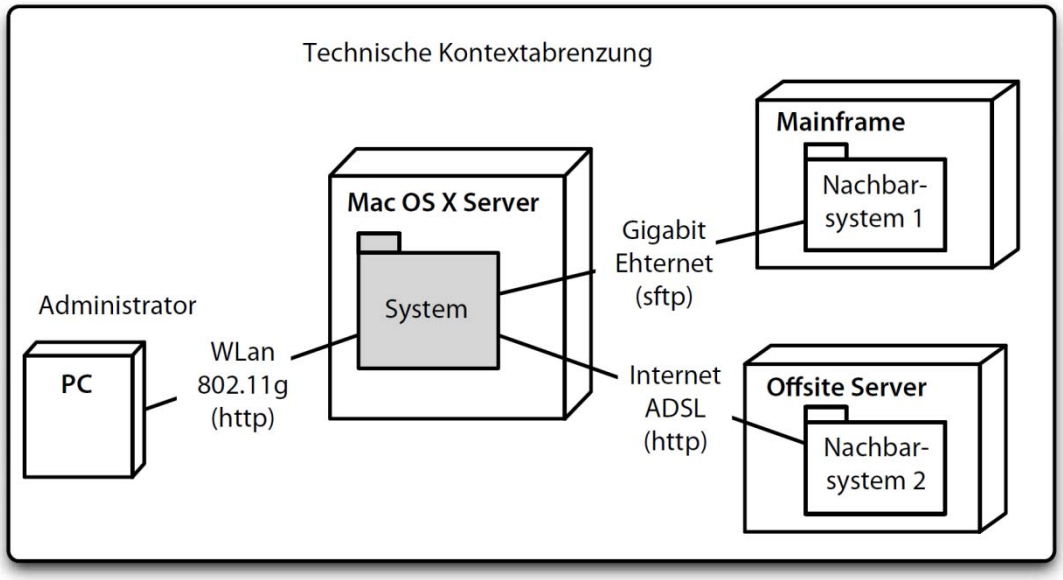
Ebenen zwischen Bausteinsicht und Laufzeitsicht nicht unbedingt identisch

Architektur-Dokumentation: Start

Architektur-
Entwicklung

- Tactics





- ...
- ...
- ...
- ...

Fachliche Schnittstelle	Technischer Kanal
Kundendaten	Gigabit Ethernet (sftp)
Reports	Internet ADSL (http)
Status	Wlan 802.11.g (http)
Aufrufe	Wlan 802.11.g (http)

Architektur-Dokumentation: arc42

1 Einführung und Ziele 1.1 Übersicht fachlicher Anforderungen 1.2 Wesentliche Architekturziele 1.3 Projektbeteiligte (Stakeholder)	Kurze Zusammenfassung der wesentlichen Anforderungen an das System. Wichtig ist die Darstellung der nichtfunktionalen Anforderungen (Qualitätsmerkmale), die i.d.R. die wesentlichen Architekturziele bilden.
2 Randbedingungen 2.1 Technische Randbedingungen 2.2 Organisatorische Randbedingungen 2.3 Konventionen	Faktoren, die den Systemarchitekten beim Entwurf einschränken.
3 Kontextabgrenzung 3.1 Fachlicher Kontext 3.2 Technischer oder Verteilungskontext	Die Top-Level Darstellung des Systems in seinem fachlichen oder technischen Umfeld.
4 Bausteinsicht	Die statische Struktur, die Zerlegung des Systems in Implementierungsbestandteile, hier Bausteine genannt. Beschrieben in verschiedenen Verfeinerungsebenen.
5 Laufzeitsicht	Zeigt die Zusammenarbeit von Bausteinen zur Laufzeit, illustriert die Dynamik des Systems.
6 Verteilungssicht	Beschreibt die möglichen Ausführungsumgebungen des Systems, seine technische Infrastruktur.
7 Typische Muster, Strukturen und Abläufe	Wiederholt benutzte oder auftretende Strukturen, die sich in die Top-down-Zerlegungen von Kapitel 4 und 6 nicht systematisch einfügen lassen.
8 Übergreifende Architekturaspekte und technische Konzepte	Lösungsansätze für übergreifende Themen, wie Persistenz, Transaktionen oder GUI.
9 Entwurfsentscheidungen	Zentrale architekturrelevante Entscheidungen mit Begründung

Quelle: arc42 template, *http://www.arc42.de/download*

Literatur

- Bachmann, Felix; Bass, Len; Chastek, Gary; Donohoe, Patrick; & Peruzzi, Fabio. *The Architecture Based Design Method* (CMU/SEI-2000-TR-001). Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/library/abstracts/reports/00tr001.cfm
- Jean-Marc DeBaud, Oliver Flege, Peter Knauber: PuLSE-DSSA – A Method for the Development of Software Reference Architectures; Proceedings of ISAW-3, 1998
- Rob Wojcik, Felix Bachmann, Len Bass, Paul C. Clements, Paulo Merson, Robert Nord, William G. Wood: Attribute-Driven Design (ADD), Version 2.0; (CMU/SEI-2006-TR-023). Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/library/abstracts/reports/06tr023.cfm
- Stefan Zörner: Software-Architekturen dokumentieren und kommunizieren; Hanser, 2012 *http://www.arc42.de*