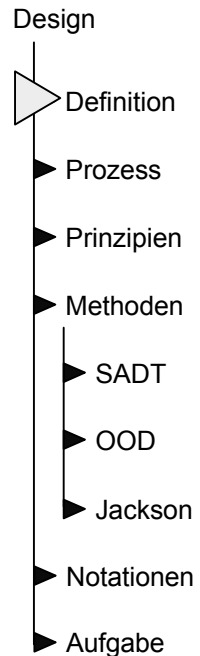


- Definition
- Der Prozess
- Prinzipien
- Strategien und Methoden
- Notationen

- Aufgabe



Design ist

- der Prozess zum Definieren
 - der Architektur,
 - der Komponenten,
 - der Schnittstellen und
 - anderer Charakteristika (Datenstrukturen, Algorithmen etc.)eines Systems oder einer Komponente sowie
- das Ergebnis dieses Prozesses.

Quelle: IEEE Std 610.12-1990.

IEEE Standard Glossary of Software Engineering Terminology



Partner-Diskussion: Design-Begriffe

Design

- ▶ Definition
- ▶ Prozess
- ▶ Prinzipien
- ▶ Methoden
 - ▶ SADT
 - ▶ OOD
 - ▶ Jackson
- ▶ Notationen
- ▶ Aufgabe

- Klären Sie mit einem Partner die folgenden Begriffe aus der vorherigen Definition:

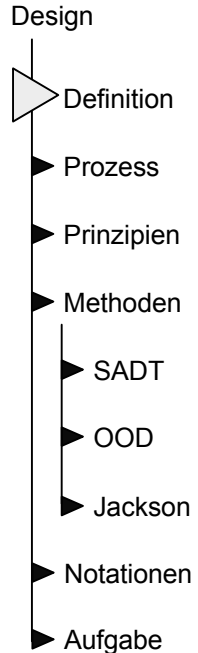
- Architektur
- Komponente
- Schnittstelle
- System

Notieren Sie schriftlich Ihre "Definitionen" (keine perfekte Formulierung notwendig) sowie mindestens ein Beispiel pro Begriff!

- Dauer: 5 Minuten



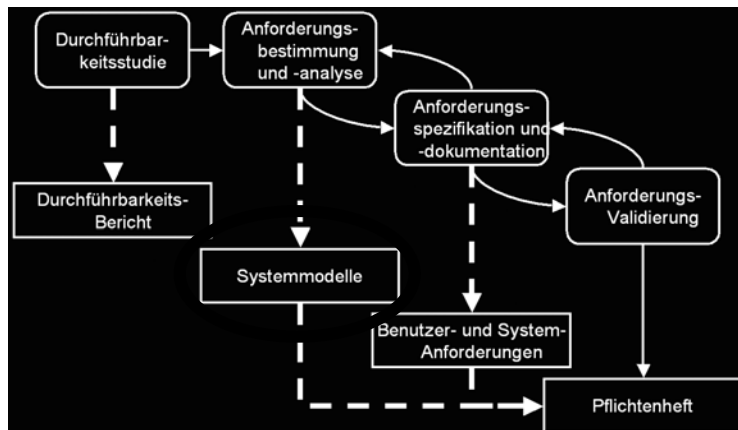
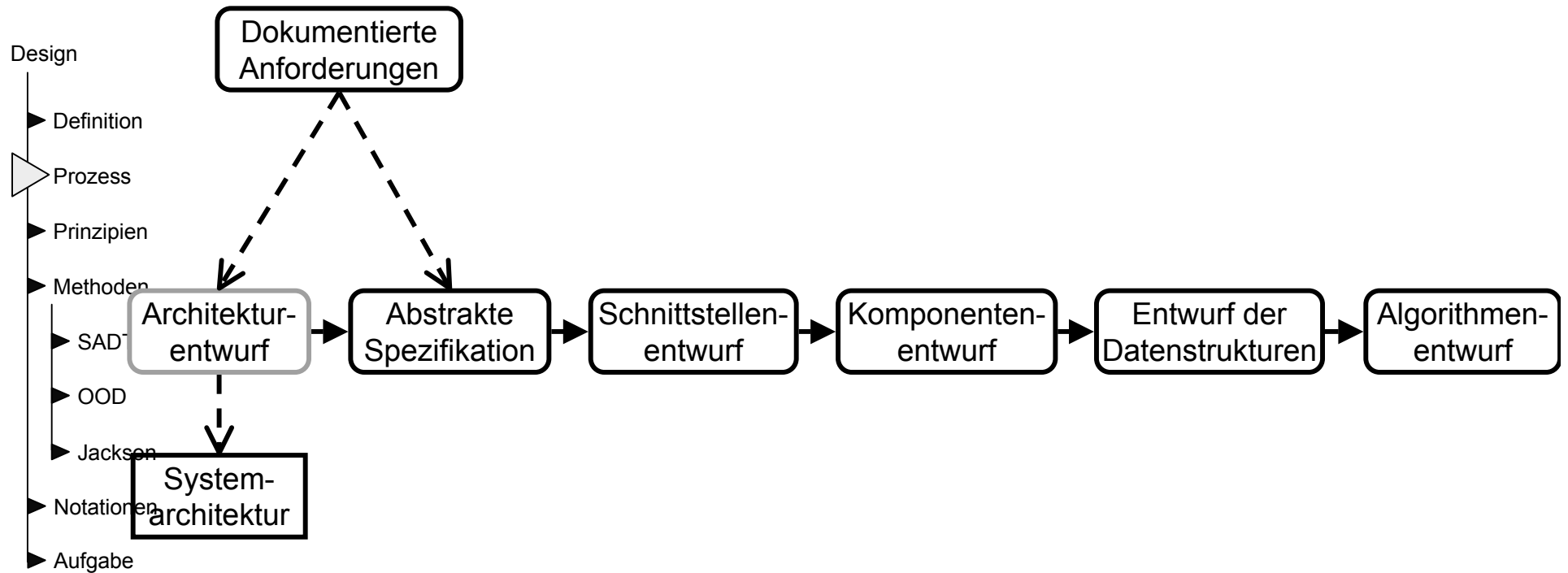
Vorgehen und Ergebnis



- Anforderungen werden analysiert, um eine Beschreibung der
 - internen Struktur und
 - Organisationeines Systems zu erstellen, die als Basis der Konstruktion (Implementierung) dient
- Das Ergebnis beschreibt die Architektur des Systems, das heißt
 - wie das System aus Komponenten aufgebaut ist und
 - die Schnittstellen/Beziehungen zwischen diesen Komponenten
- **Ziel:**
Das Ergebnis muss eindeutig (genug) sein, damit es bei der Implementierung keine Abstimmungsprobleme gibt!

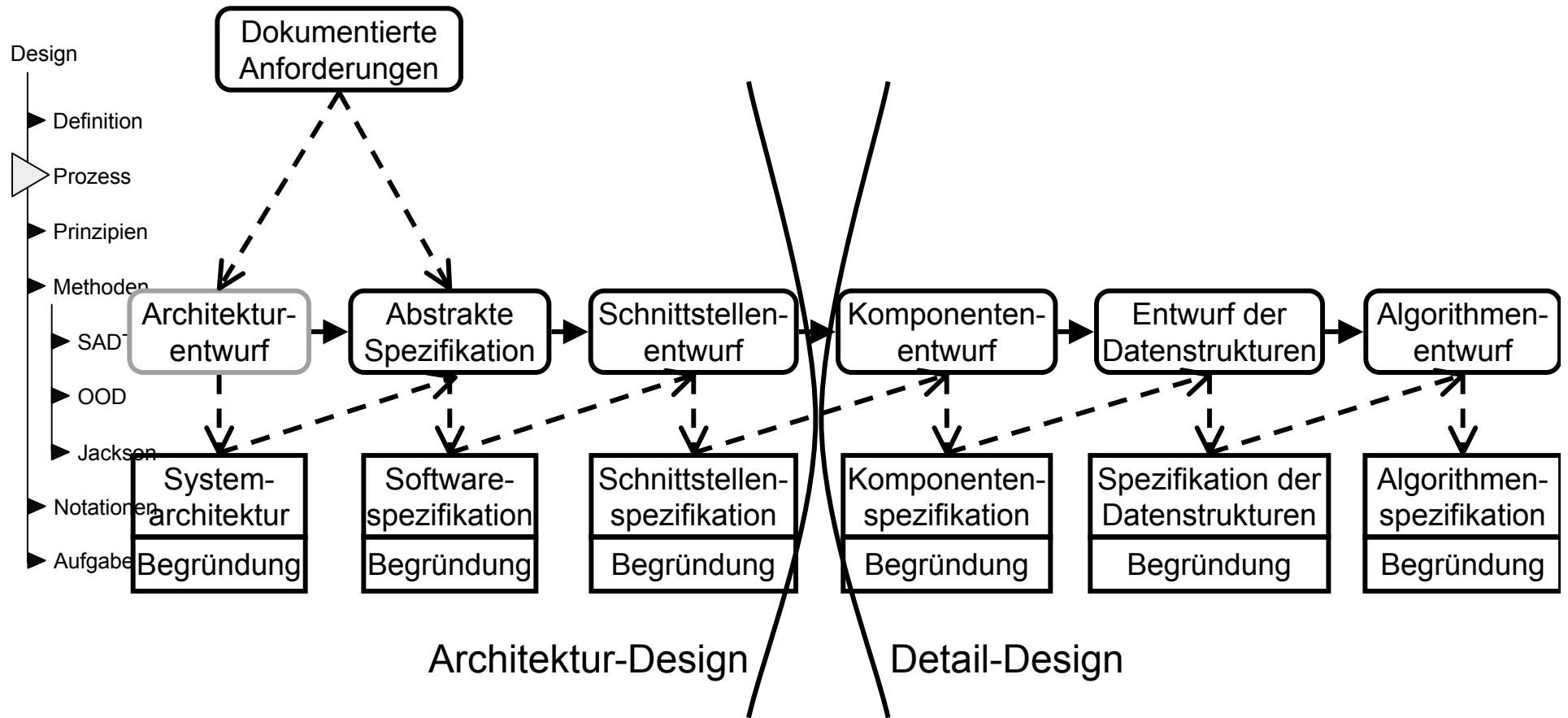


Der Entwurfsprozess



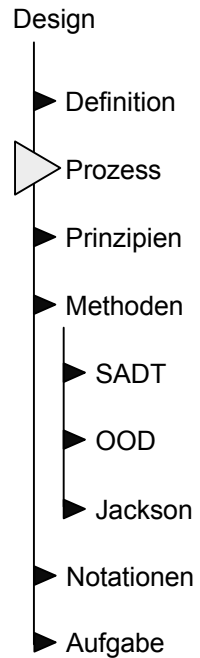


Der Entwurfsprozess





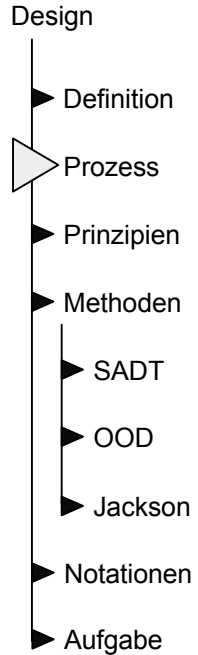
Design als kreativer Prozess



- Lösen schwieriger Probleme – Probleme, die keine eindeutige Lösung besitzen (daher müssen Begründungen dokumentiert werden!)

Zugehörige Konzepte

- Ziele
- Einschränkungen / Beschränkungen (*Constraints*)
- Alternativen
- Darstellungen
- Lösungen

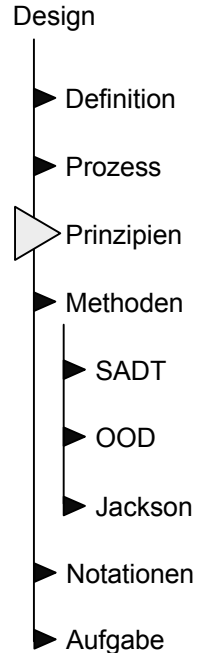


- Architektur-Design (Top-Level-Design)
 - Top-Level-Struktur
 - Komponenten, Beziehungen zwischen Komponenten
- Detail-Design
 - Genaue Beschreibung jeder Komponente für die Implementierung





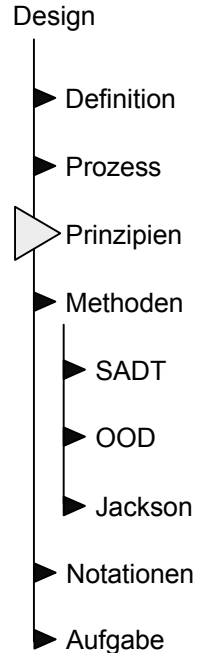
Design-Prinzipien (*enabling techniques*) 1/2



- Abstraktion – der Prozess, Information zu vergessen
 - Abstraktion durch Parametrisierung, durch Spezifikation
 - Prozedurale Abstraktion, Datenabstraktion, Kontrollabstraktion
- Coupling and Cohesion
 - Zusammenhang zwischen Komponenten
 - Zusammenhang (innerhalb) der Komponenten
- Dekomposition und Modularisierung
 - Zerlegen eines großen Systems in kleinere, beherrschbare Einheiten
 - Verteilen von Funktionalität auf Komponenten



Design-Prinzipien (*enabling techniques*) 2/2



- Kapselung, Information Hiding
 - Verbergen der Details einer Komponente
 - Unzugänglich machen der Details
- Trennen von Schnittstelle und Implementierung
 - Komponenten durch ihre Schnittstellen spezifizieren
- Vollständigkeit und Primitivität
 - Komponenten enthalten alles, was zu einer Abstraktion gehört, aber nicht mehr



Partner-Interview: Design-Prinzipien

Design

- ▶ Definition
- ▶ Prozess
- ▶ Prinzipien
- ▶ Methoden
 - ▶ SADT
 - ▶ OOD
 - ▶ Jackson
- ▶ Notationen
- ▶ Aufgabe

- Erklären Sie abwechselnd mit Ihrem Partner die folgenden Begriffe
 - Abstraktion
 - Coupling and Cohesion
 - Dekomposition und Modularisierung
 - Kapselung, Information Hiding
 - Trennen von Schnittstelle und Implementierung
 - Vollständigkeit und Primitivität

Geben Sie für jeden Begriff mindestens ein Beispiel an!

- Dauer: 6 Minuten



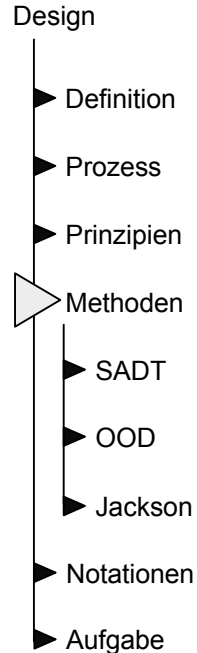
Design

- ▶ Definition
- ▶ Prozess
- ▶ Prinzipien
- ▶ Methoden
 - ▶ SADT
 - ▶ OOD
 - ▶ Jackson
- ▶ Notationen
- ▶ Aufgabe

Fragen?



Design-Strategien und -Methoden



- Allgemeine, methoden-übergreifende Strategien
 - Divide and Conquer
 - Stepwise Refinement
 - Top-Down <-> Bottom-Up
- Spezielle Vorgehensweisen / Methoden
 - Funktions-orientiertes (Strukturiertes) Design
 - Objekt-orientiertes Design
 - Datenstruktur-orientiertes Design
 - Formale Methoden
 - Transformationsmethoden
 - Beispiel: Model-Driven Architecture (MDA)

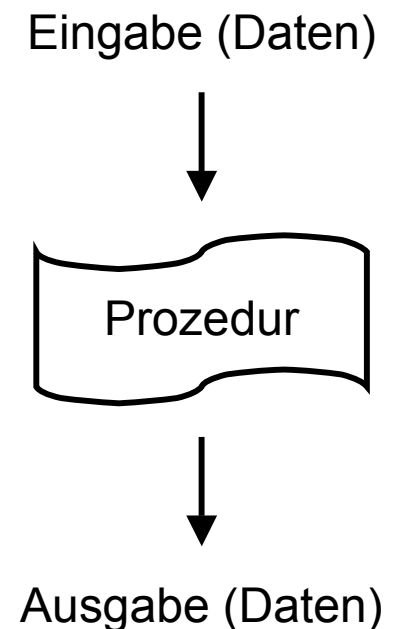


Strukturiertes Design

Design

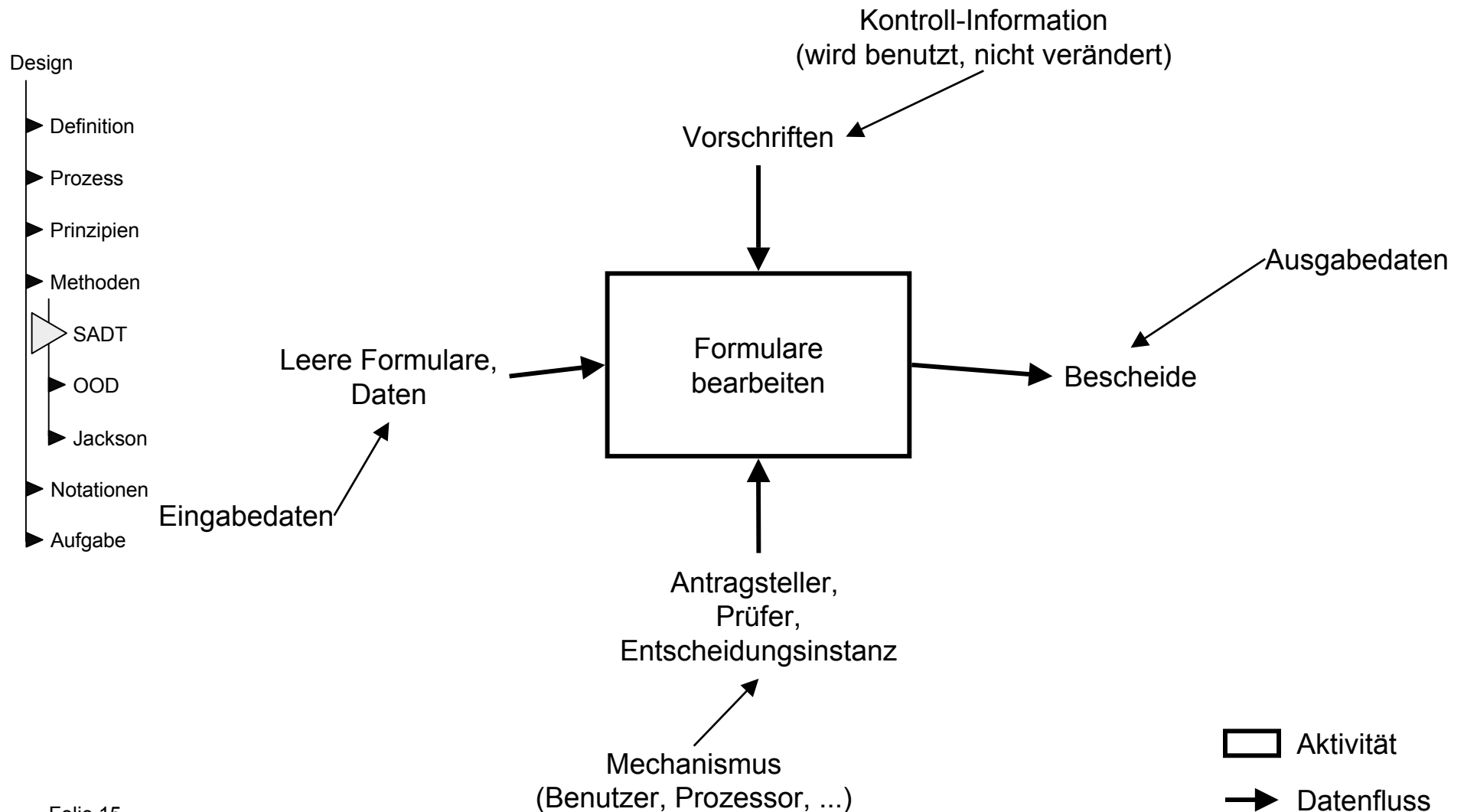
- ▶ Definition
- ▶ Prozess
- ▶ Prinzipien
- ▶ Methoden
 - ▶ SADT
 - ▶ OOD
 - ▶ Jackson
- ▶ Notationen
- ▶ Aufgabe

- Meist anschließend an eine Strukturierte Analyse
- Prinzip
 - Identifikation der Haupt-Funktionalität
 - Top-Down-Verfeinerung
- Ergebnisse
 - Datenfluss-Diagramme
 - Prozessbeschreibungen
- Eine mögliche Notation für Strukturierte Analyse und Design:
SADT = Structured Analysis and Design Technique



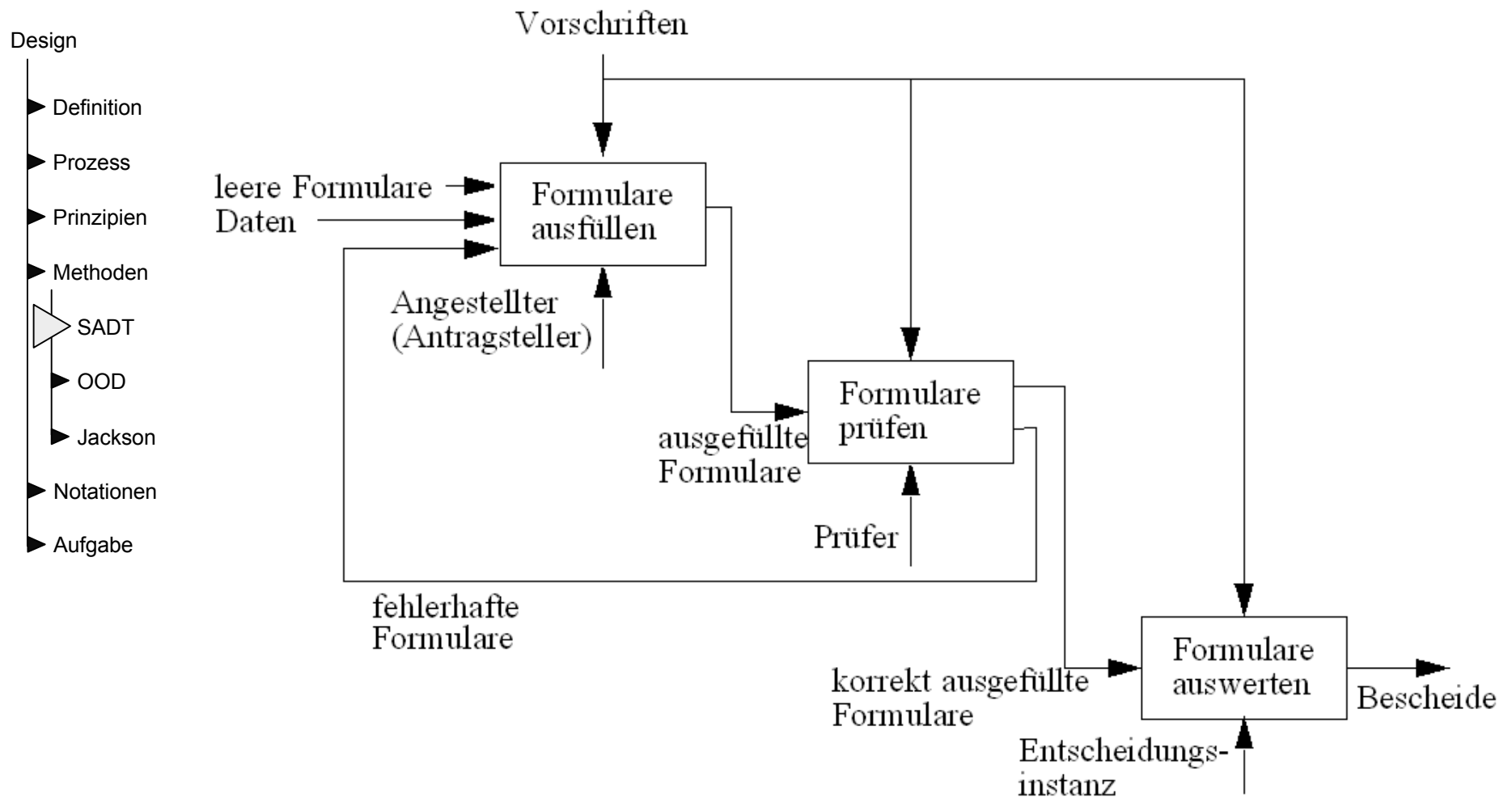


SADT-Diagramm: Beispiel "Formulare bearbeiten", A-0

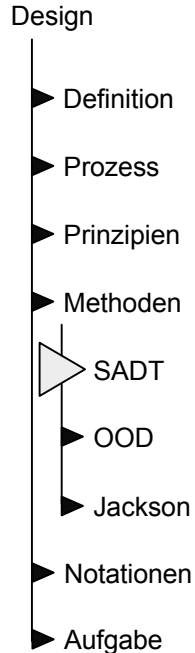




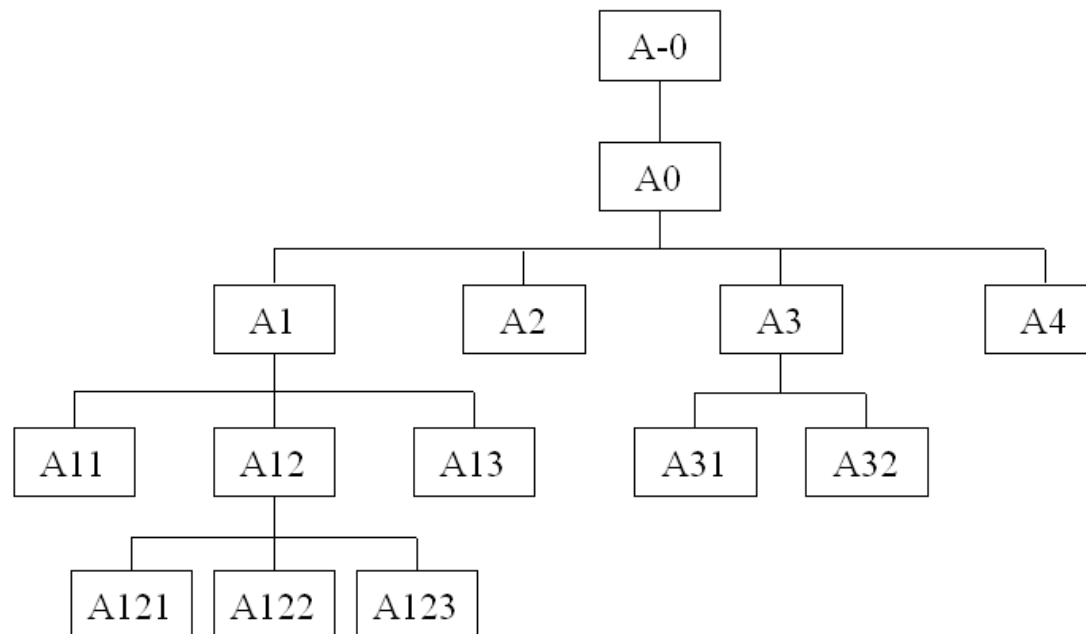
SADT-Diagramm: Beispiel "Formulare bearbeiten", A0



SADT-Konzepte



- Systembeschreibung jeweils von einem Standpunkt aus; unter Umständen mehrere Standpunkte nacheinander, Beispiel: Benutzer und Verwaltung einer Bibliothek
- Beschreibung beginnt auf der höchsten Abstraktionsebene; Verfeinerungen führen zu hierarchischer Zerlegung des Systems
- Zerlegung in Teilsysteme derart, dass jedes Teilsystem unabhängig von den anderen Teilen der gleichen Ebene verfeinert werden kann
- Keine Darstellungsmöglichkeiten für den Kontrollfluss (Bedingungen etc.)



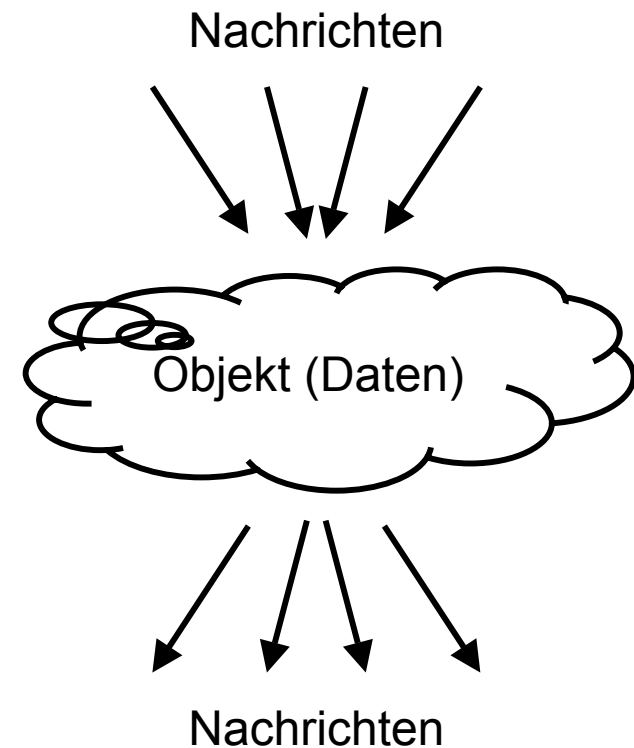


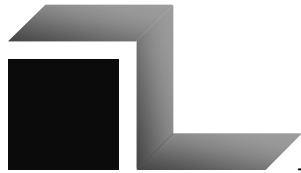
Objektorientiertes Design

Design

- ▶ Definition
- ▶ Prozess
- ▶ Prinzipien
- ▶ Methoden
- ▶ SADT
- ▶ OOD
- ▶ Jackson
- ▶ Notationen
- ▶ Aufgabe

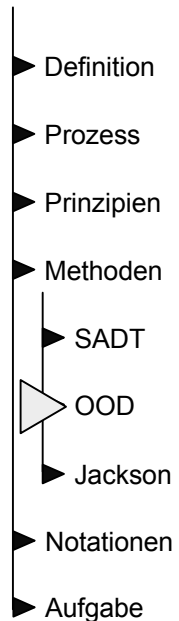
- **Prinzip**
Jedes Objekt/Ding (jede "Art" von Objekten) in der Realität wird durch ein Objekt (eine Klasse von Objekten) im Modell repräsentiert
- **Konzepte**
 - Klassen, Objekte
 - Vererbung, Polymorphismus
 - Komponenten
- **Ursprung: Daten-Abstraktion**
- **"Standard"-Notation:**
UML (Unified Modeling Language)





Objektorientierter Entwurf: Heuristische Vorgehensweise [Booch]

Design



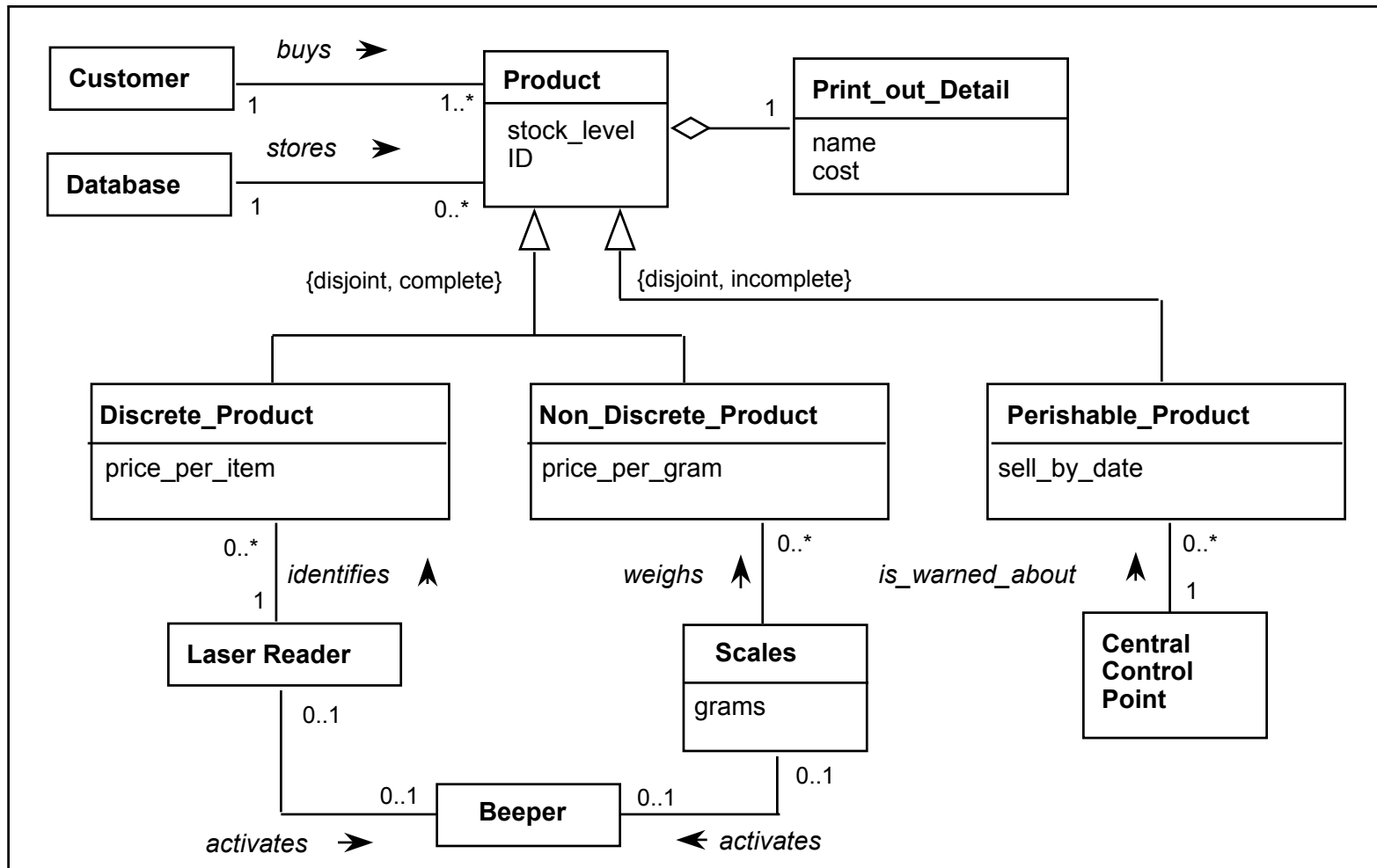
1. Identifiziere Klassen und Attribute anhand der *Substantive* aus der Anforderungsbeschreibung
2. Identifiziere Methoden (für die Klassen) anhand der *Verben* aus der Anforderungsbeschreibung
3. Definiere, welche Objekte welche andere sehen (müssen): Beziehungen (inklusive Vererbung) zwischen Klassen
3. Definiere die Schnittstelle (Interfaces) der Klassen: Festlegen von Parameterlisten, Ergebnistypen für die Methoden
4. Implementiere die Klassen, Attribute und Methoden: Wahl einer geeigneten Repräsentation in der Programmiersprache



Klassendiagramm: Beispiel "Produkte in einem Lager"

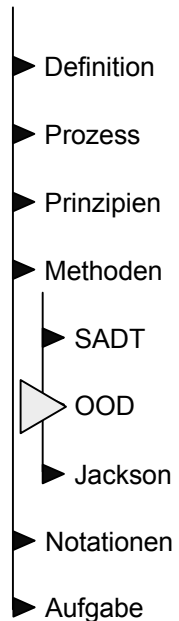
Design

- Definition
- Prozess
- Prinzipien
- Methoden
- SADT
- OOD
- Jackson
- Notationen
- Aufgabe





Design



Anleitung

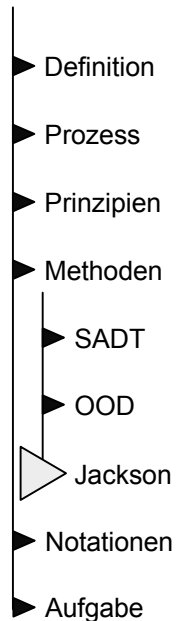
Aufgabe

- Auf der Vorlesungs-Homepage finden Sie zwei Implementierungen von Parkscheinautomaten:
Wählen Sie einen davon aus, *bringen Sie diesen zur nächsten Stunde mit*
- Dokumentieren Sie Szenarien / Use Cases, die von dem gewählten Automaten unterstützt werden
- Entwickeln Sie nach dem Muster der SIB / withdraw ein Design mit den notwendigen Abläufen und Klassen



Datenstruktur-orientiertes Design

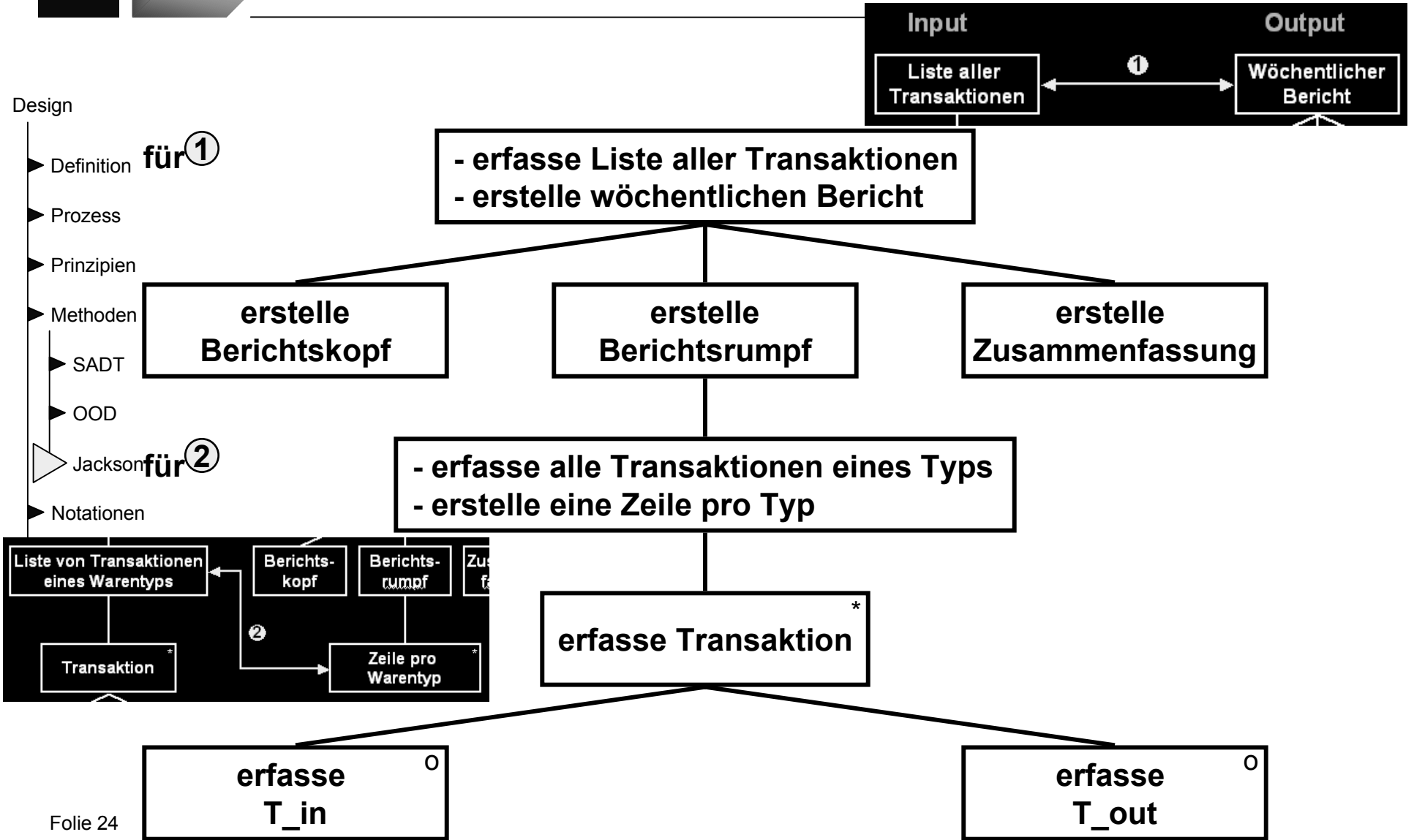
Design



- Prinzip
 - Ausgangsbasis: Struktur von Ein- und Ausgabedaten
 - Kontrollstrukturen werden so definiert, dass sie Eingabedaten in Ausgabedaten transformieren können
- Beispiel-Notation:
Jackson Structure Diagrams (Michael A. Jackson)
- Beispiel "Bestandsführung":
Es soll ein wöchentlicher Bericht über Bestandsänderungen in einem Lager generiert werden, sortiert nach Warentypen
 - 1. Schritt: Datenstruktur der Ein- und Ausgabedaten festlegen
 - 2. Schritt: passende Transformationen entwerfen



Jackson Struktur-Diagramm: Beispiel "Bestandsführung", Programmstruktur

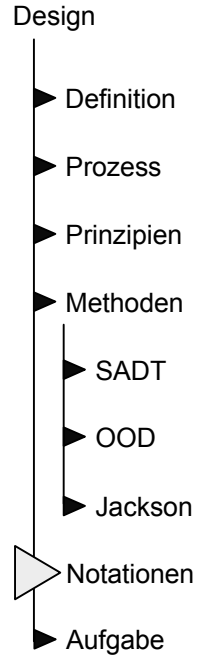




Design

- ▶ Definition
- ▶ Prozess
- ▶ Prinzipien
- ▶ Methoden
 - ▶ SADT
 - ▶ OOD
 - ▶ Jackson
- ▶ Notationen
- ▶ Aufgabe

Fragen?

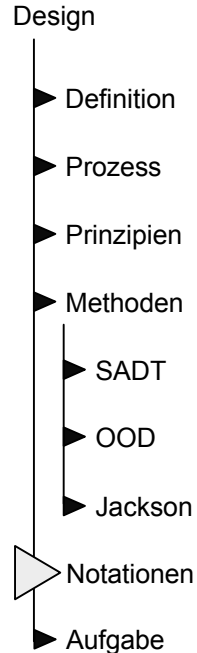


Aufgaben

- Beschreibung der Struktur
 - Beschreibung der Komponenten
 - Beschreibung der Beziehungen zwischen Komponenten
 - Statische Sicht
 - Überwiegend graphische Darstellung
- Beschreibung des (dynamischen) Verhaltens
 - Überwiegend für das Detail-Design genutzt
 - Dynamische Sicht
 - Graphische oder textuelle Darstellung



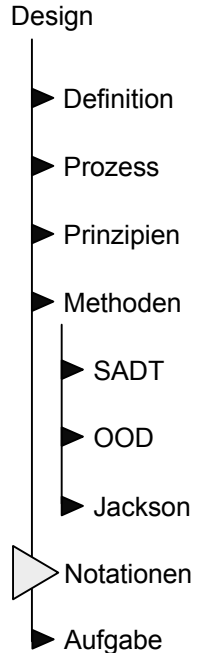
Design-Notationen zur Strukturbeschreibung



- Architektur-Beschreibungssprachen (ADL)
 - Komponenten und Konnektoren
- Klassen- und Objektdiagramme
 - Klassen und ihre Beziehungen
- Entity-Relationship-Diagramme (ER)
 - Konzeptionelle Datenmodelle (für Datenbanken)
- Schnittstellen-Beschreibungssprachen (IDL)
 - Name und Typen von Operationen von Komponentenschnittstellen
- Jackson-Structure-Diagrams
 - Datenstrukturen mittels Sequenz, Alternative, Iteration
- Structure Charts
 - Aufrufstruktur von Unterprogrammen/Modulen



Design-Notationen zur Verhaltensbeschreibung



- **Aktivitätsdiagramme, Flussdiagramme**
 - Kontrollfluss zwischen Aktivitäten
- **Kollaborationsdiagramme, Sequenzdiagramme**
 - Interaktion zwischen Objekten
 - Fokus auf ausgetauschten Nachrichten / auf deren zeitlichen Abfolge
- **Datenflussdiagramme**
 - Datenfluss zwischen Prozessen
- **Entscheidungstabellen und –Diagramme**
 - Darstellung komplexer Aktionen unter komplexen Bedingungen
- **Formale Spezifikationssprachen**
 - Mathematisch basiert, oft durch Vor- und Nachbedingungen von Aktivitäten
- **Pseudo-Code**
 - Detaillierte Verhaltensbeschreibung
- **Zustandsübergangstabellen und –Diagramme**
 - Zustandsänderungen durch Aktivitäten