Software Engineering I

5. Entwurf

Prof. Dr. Eckhard Kruse

DHBW Mannheim

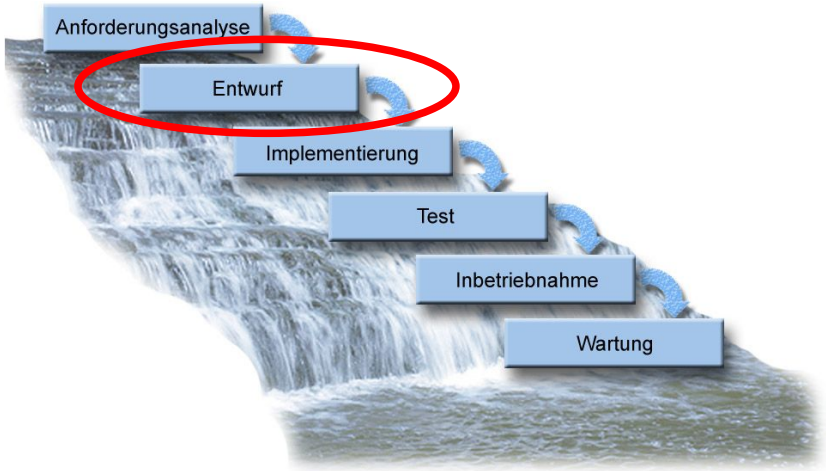
Entwurf

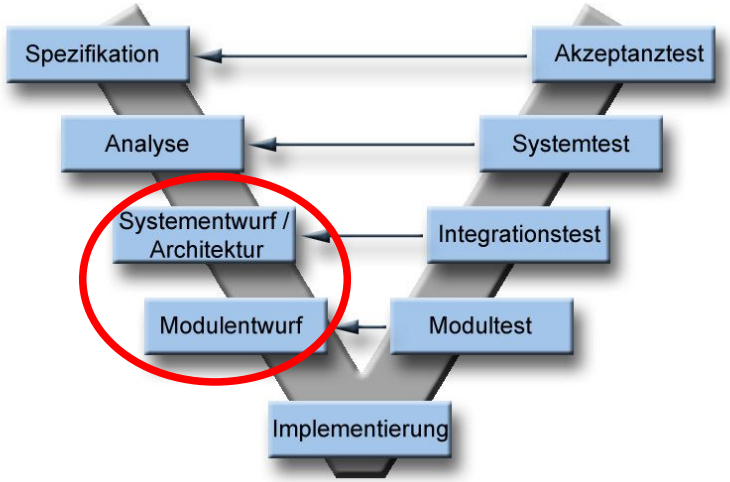
Definition

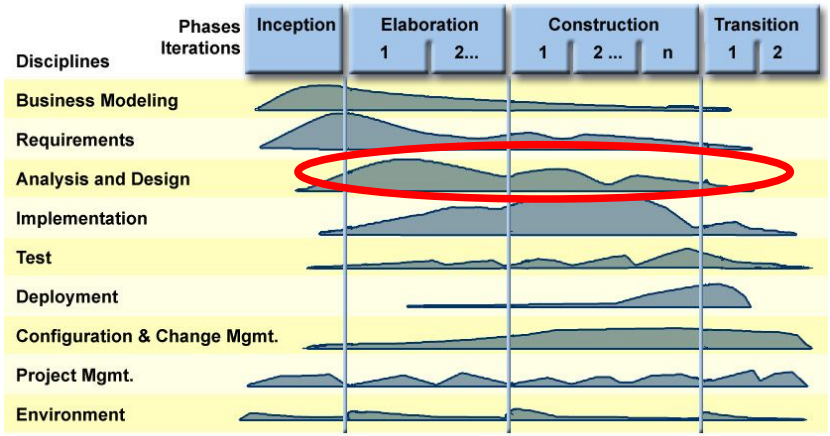
Der **Entwurf (engl. design)** dient dazu, auf Basis der Anforderungen schrittweise eine entsprechende technische Realisierung zu entwerfen. Hierzu wird das System schrittweise vom Groben ins Feine in Komponenten und Verantwortlichkeiten zerlegt.

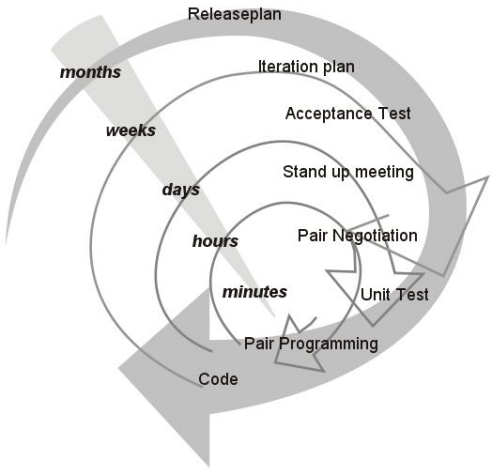
- Der Entwurf erfolgt auf verschiedenen Ebenen (je komplexer das System, desto mehr Ebenen).
- Typische Begriffe sind Systementwurf (=Gesamtsystem), Architektur (Gesamtsystem oder Teilsysteme / Komponenten), Design (auf Komponenten- / Modulebene). (Die Grenzen der Begriffe sind fließend.)
- Der Fokus liegt auf der Aufteilung in Komponenten/Module, Verantwortlichkeiten und Schnittstellen untereinander – nicht auf deren internen Realisierung.
- **Ergebnisse:** Entwurfsdokumente, Diagramme (z.B. UML)

Enwurf in den Vorgehensmodellen









Entwurf – ein weiter Begriff

Beispiel

Entwurf
eines Sortieralgorithmus

Entwurf
eines komplexen Softwaresystems für die industrielle Automatisierung

Was ist gleich?

Was ist verschieden?

Softwaresysteme sind hierarchisch aufgebaut:

- (Produktlinie)
- Produkt/System
- Subsystem
- Komponente
- Modul
- Klasse

Entwurf

Wo wäre hier der Entwurf eines Algorithmus einzuordnen?

Systementwurf

Definition

Beim **Systementwurf (engl. system design)** wird die Software-Architektur, d.h. der Bauplan der Software, entwickelt. Das Gesamtsystem wird auf einer groben Ebene in Schichten/Teilsysteme/Komponenten zerlegt, die verschiedene Aufgaben übernehmen (z.B. Datenhaltung, Benutzerschnittstelle, Geschäftslogik).

Wichtige Prinzipien:

- Dokumentation der Architekturentscheidungen: Was sind die Ziele und wie sollen sie erreicht werden (und warum nicht anders)
- "Separation of concerns": Aufteilung verschiedener Funktionen in verschiedene Softwarekomponenten.
- Definition von Schnittstellen (auf grober Ebene): Was/wie kommunizieren die unabhängigen Komponenten?
- Grundlegende technische Fragen: Betriebssystem, 3rd party-Komponenten, Frameworks
- Unterscheidung verschiedener Sichten, z.B. Komponenten, Deployment, Informationsmodell, Data+Control flow / collaboration

→ Ergebnis: Architekturdokument

Aufteilung in Komponenten

Eine Komponente...

- dient einem klar umrissenen Zweck: “separation of concerns”
- fasst eng zusammengehörige Funktionalität zusammen: “high cohesion”
- verbirgt interne Implementierungsdetails: “information hiding”
- ist nur lose, über wenige Schnittstellen mit anderen Teilsystemen gekoppelt: “low coupling"
- besteht aus öffentlicher Schnittstelle und Rumpf (ihre Realisierung)
- sollte sich als eigenständige Einheit entwickeln, übersetzen und testen lassen (ggf. mit hilfsweisen Dummy-Implementierungen für benötigte externe Komponenten)
- ist typischerweise in der Verantwortung einer Person/eines Teilteams.

Sichten

Definition

Beim Entwurf unterscheidet man verschiedene **Sichten (Views)** auf das System, die verschiedene Aspekte der Architektur in den Mittelpunkt stellen.

Wichtige Sichten (Notwendigkeit/Umfang hängt vom konkreten Projekt ab):

- **Logische Struktur:** Hierchische Aufteilung des Systems. Unterscheidung verschiedener Granulationsebenen.
  - z.B. Komponentendiagramm, Klassendiagramm, Objektdiagramm
- **Physikalische Sicht:** Abbildung der Software auf (ggf. logische) Hardwarekomponenten, z.B. Client, Server
  - z.B. Verteilungsdiagramm (Deployment Diagram)
  - insbesondere bei verteilten Systemen
- **Teilsysteme:** Zerlegung des Quellcodes in unabhängig bearbeitbare Teile mit definierten Schnittstellen
  - Paketdiagramm (Package Diagram)
- **Datenfluss:** Wie fließen Daten durch das System
  - z.B. Aktivitätsdiagramm, Kommunikationsdiagramm
- **Kontrollfluss**: Welche Operationen rufen sich gegenseitig in welcher Reihenfolge auf, wo gibt es Nebenläufigkeiten
  - z.B. Aktivitäts-, Sequenz- und Kommunikationsdiagramm

UML

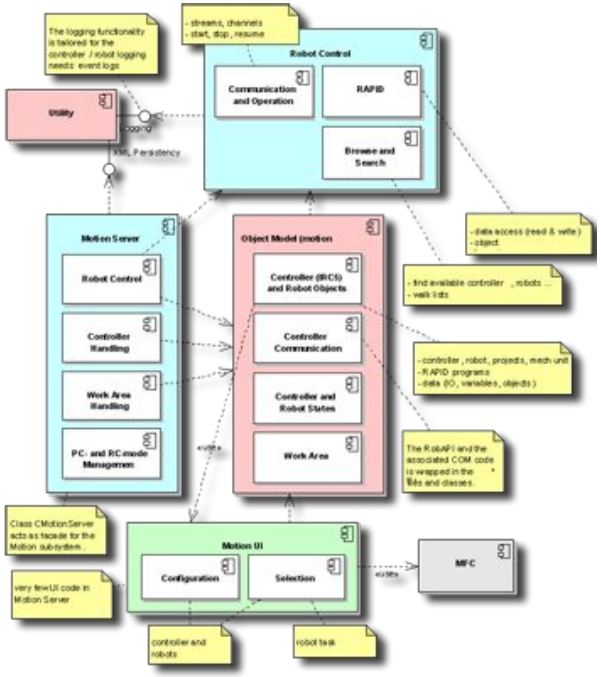
Definition

UML (Unified Modelling Language) ist eine Notation und Semantik zum Entwurf, zur Visualisierung und Dokumentation von Modellen für die (insbesondere objektorientierte) Softwareentwicklung.

UML ist ein Standard der OMG (http://www.omg.org/uml).

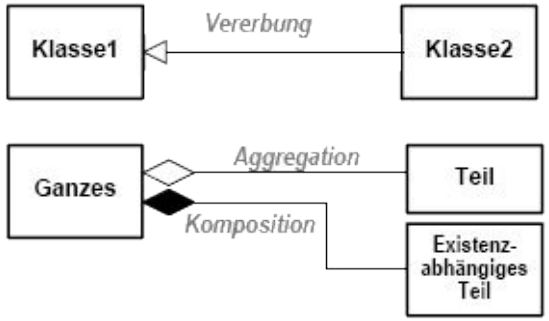
- Sehr weit verbreitet, UML-Wissen ist "Pflicht" für Softwareentwickler
- Einsatz in der Anforderungsanalyse: Use Cases, Modellierung von Geschäftsprozessen
- Einsatz im Entwurf: Diverse Sichten auf die Architektur
- Einsatz in der Implementierung (z.B. Codegenerierung aus den Modellen)
  - aber weniger häufig, u.a. Problem des *Roundtrip-Engineerings*
- Es gibt zahlreiche UML-Werkzeuge für die Erstellung von UML Diagrammen und die Generierung von Code (ggf. in die IDE integriert)
- Aktuelle Version: UML 2.5.1

Wichtig: UML **+ Annotationen + textuelle Beschreibung**

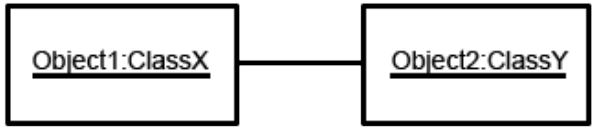


UML Strukturdiagramme (1/2)

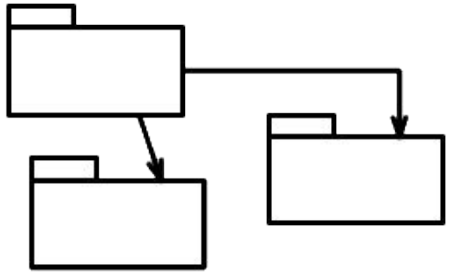
Klassendiagramm (Class diagram)
Statische Struktur von Klassen und Beziehungen (Assoziationen, Aggregation, Komposition, Spezialisierung/ Generalisierung)



Objektdiagramm (Object diagram)
Struktur von Instanzen+Verbindungen, Laufzeitszenarien

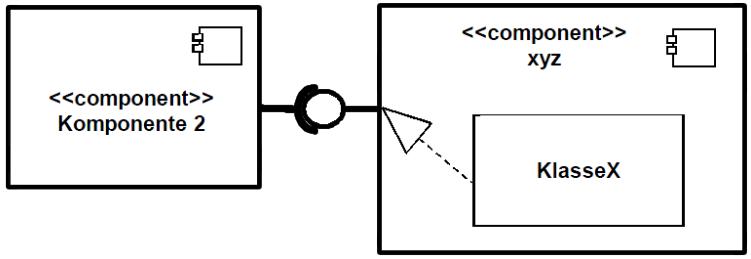


Paketdiagramm (Package diagram)
Zerlegung des Gesamtsystems in Pakete/Teilsysteme (typisch: Source code packages)

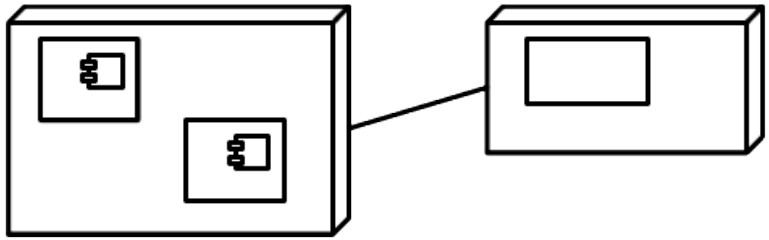


UML Strukturdiagramme (2/2)

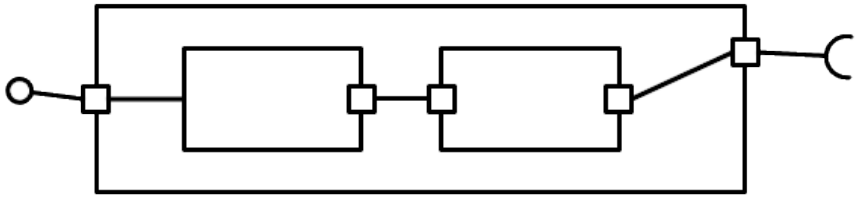
Komponentendiagramm (Component diagram)
Komponenten und Schnittstellen (Anbieter/Nutzer)



Verteilungsdiagramm (Deployment diagram)
Verteilung von Systembausteinen
auf die Hardware/Infrastruktur

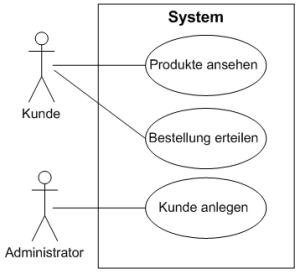


Kompositionsstrukturdiagramm (Composite structure diagram)
Laufzeitsicht auf Systembausteine, ihre
Zusammenarbeit und Schnittstellen,
z.B. interne Klassenstruktur

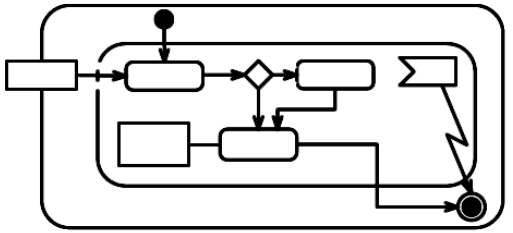


UML Verhaltensdiagramme (1/2)

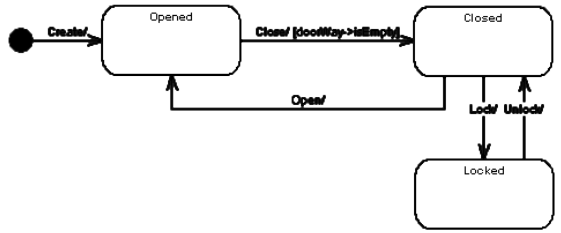
Anwendungsfalldiagramm (Use Case diagram)
Übersicht alle Vorgänge/Prozesse aus Sicht der Anwender (Akteure) und anderer externer Systeme



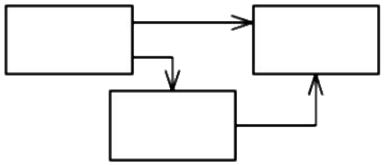
Aktivitätsdiagramm (Activity diagram)
Abläufe innerhalb von Systembausteinen, z.B. Algorithmen, Prozessbeschreibungen (ähnlich wie Flußdiagramme)



Zustandsdiagramm (State diagram)
Zustände eines Systembausteins, Zustandsübergänge und zugehörige Aktivitäten.

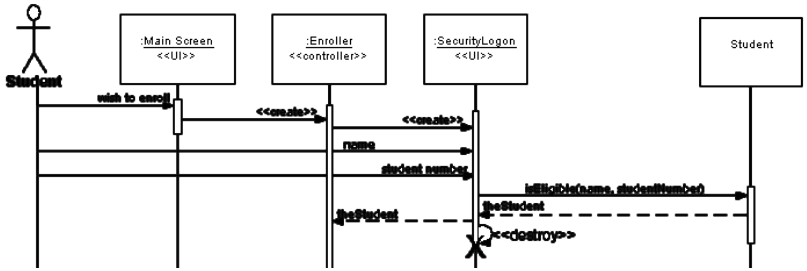


Kommunikationsdiagramm (Communication diagram)
Zusammenarbeit zwischen Instanzen von Systembestandteilen (früher: Kollaborationsdiagramm)

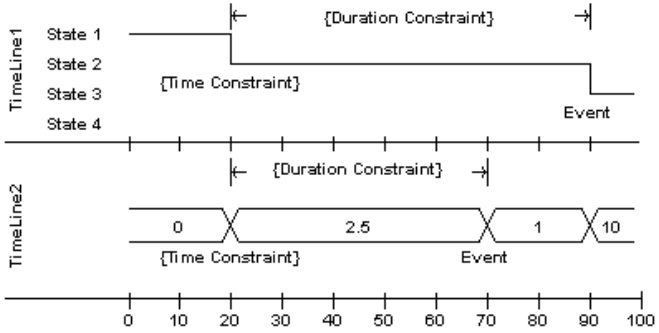


UML Verhaltensdiagramme (2/2)

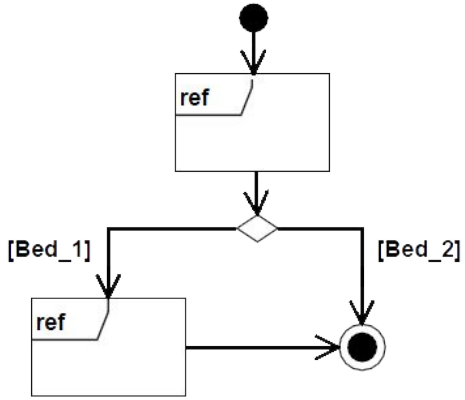
Sequenzdiagramm (Sequence diagram)
Nachrichtenaustausch zwischen Instanzen von Systembausteinen, meist konkrete Szenarien



Zeitdiagramm (timing diagram)
Zeitablauf, Zustände/Werte über der Zeit, Zeitabhängigkeiten zwischen Ereignissen



Interaktionsübersichtsdiagramm (Interaction Overview diagram)
Aktivitätsdiagramm auf hoher Abstraktionsebene: Zusammenspiel verschiedener Interaktionen + Referenzen auf Detaildiagramme



Architekturdokument Beispiele

Beispiel

- Arc42 (gutes Material, gute Grundlage für das Teamprojekt)
  - **`http://www.arc42.de`**
  - Dokumententemplates
  - Beispiel: https://www.dokchess.de/
- Software Architecture Document Template from CMU SEI
  - **`https://wiki.sei.cmu.edu/confluence/display/SAD/Software+Architecture+Documentation+Template`**
- Industry example, architecture documentation
- eStore Example
- Template
- RUP
- Design ready checklist

Lieferant: Entwurf

Software-Engineering-Projekt

P.13 Lieferant: Entwurf

Entwerfen Sie schrittweise die Architektur der Softwarelösung:

a) Welche Ziele verfolgt die Architektur? Inwieweit dient die Architektur der Umsetzung der funktionalen und nicht-funktionalen Anforderungen?

b) Wie sieht die Systemarchitektur aus: Was sind wesentliche Elemente/Komponenten? Welche Sichten sollten beschrieben werden?

c) Welches sind die grundlegenden zu verwendenden Technologien (z.B. Betriebssystem, Datenbanken, Programmiersprache). Sind evtl. Technologieevaluationen und/oder Prototypen erforderlich, um Entscheidungen fundiert treffen zu können?

d) Beschreiben Sie die Architektur und wesentliche Entscheidungen in einem Dokument. Charakterisieren Sie typische Anwendungsszenarien des Systems.

Lieferant: Entwurf Review

Software-Engineering-Projekt

P.14 Lieferant: Entwurf Review

Ihre Entwurfsphase ist abgeschlossen und das Dokument zur Systemarchitektur liegt vor. Im Rahmen der Qualitätssicherung in Ihrem Unternehmen soll ein erfahrener Architekt aus einer anderen Abteilung ein Assessment vornehmen.

a) Bereiten Sie eine kurze Präsentation vor, mit der Sie den Assessor von der Qualität Ihrer Softwarearchitektur überzeugen können.

b) Aktualisieren Sie ggf. Ihre Projektplanung und Ihr Risikomanagement.