

Modul 326

|  |
| --- |
| Objektorientiert entwerfen und implementieren |
| Dokumentation |

**Überschrift**

***Untertitel***:

Dies ist ein normaler Text

* Auflistung 1
  + Auflistung 2
    - Auflistung 3

Das da hier ist Code. lol also nicht das da, aber in diesem Format

// Kommentar

**Pro** // das ist gut

**Kontra** // das ist nicht gut

¯\\_(ツ)\_/¯

Softwareprobleme Ende 1960er

Problem 1:

Niedrige Produktivität der Programmierer

Problem 2:

Unterhalt - Kunden möchten Programm optimieren / anpassen > Programmierer ist mit code überfordert

Problem 3:

Unterschiedliches Vorgehen - Verschiedene Sprachen für Analyse & Entwurf

Problem 4:

Wo ist oben? - Kein klarer Programmstart ; Wenn etwas angepasst wird - habe ich überall alles abgeändert?

Problem 5:

offen oder geschlossen?

Zusammengehöriger Code wird zu Modulen zusammengefasst.

Vorteil: Lokalisierte, bessere Wartbarkeit, dadurch besser wiederverwendbar

Nachteil: Bei Wiederverwendung passt es dann doch nicht ganz, also doch abändern und anpassen

Lösung: Objektorientierte Programmierung

Proz: Daten und Operationen; Daten sind statisch, Operationen transient

OO: Zusätzliche Abstraktionsebene, Objekte kapseln Daten und Verhalten, Klassen fassen gleiche Objekttypen zusammen, Erweiterbarkeit durch Vererbung

(Anstelle von vielen switch-cases für das selbe, nur noch eine Einzige - da, wo die Daten verarbeitet werden (Nur noch an einem Ort anstelle von vielen))

Grosser Vorteil OO: Die Analyse, Design, Entwurf und Programmierung an sich haben immer die selbe Sprachen - keine Strukturbrüche mehr, durchgängigkeit zwischen Analyse und Entwurf

UML - Unified Modelling Language - grafische Modelliersprache - heutiger Standard

OOA/OOD (Objectoriented Analysis / Objectoriented Design)

Ergebnis des OOA

Pflichtenheft - das Einstiegsdokument (Was der Kunde eigentlich will)

OOA-Modell - das Fachkonzept (Verständnis - was will der Kunde & nicht mit welcher Sprache das Programm geschrieben wird)

Prototyp der Benutzeroberfläche - Visualisierung des Programmes (Hey Kunde - ist es das, was du willst?)

PFLICHTENHEFT : Präsentation anschauen

Analyse-Modell:

Basis-Konzept - Was ist ein Objekt, Klasse, Operation & Attribut?

Statisches Konzept - Nicht veränderbare Aspekte, grundlegende Struktur (Assoziation, Vererbung, Paket)

Dynamisches Konzept - Verhalten des Systems im Zeitverlauf, Abläufe, Kommunikationsfluss (Anwendungsfall, Szenario, Botschaft, Zustandsautomat)

Use Case

Sequenz von Aktionen, welche das System in Interaktion mit Akteuren ausführt. Er wird durch ein bestimmtes Ereignis ausgelöst und ausgeführt, um ein Ziel zu erreichen.

Er ist als Black-Box zu verstehen; er beschreibt das extern wahrnehmbare Verhalten des Systems

BEISPIEL: Sie wollen ein Word-Dokument ausdrucken. Ein Ziel - und Sie müssen mit dem System interagieren (Word starten, Datei öffnen, 1-Seitig oder 2-Seitig, Drucker

auswählen, ausdrucken; fertig.

Use Case Beschreiben

Semiformal oder informal (umgangsprachlich) beschrieben,

Beschreibung als Folge von einzelnen Aktionen;

Aktionen für bessere Übersicht durchnummeriert;

Unterscheidung zwischen Standardfall (Zone wählen, Bargeld rein, Ticket raus), Erweiterungen (Zone wählen, Strecke wählen, Abfahrt und Gleis wird angezeigt), Alternative Abläufe (Zone wählen, Kartenzahlung wählen, Karte rein, PIN, Karte raus, Ticket raus);

Beschreibung stets unabhängig von der Benutzeroberfläche;

Use Case Template

Anwendungsfall < Name des Anwendungfalls > Ticket kaufen

Ziel < Zielsetzung beterfolgreicher Ausführung des Falles > Kunde kauft ein Ticket

Kategorie < primär (notwendig, häufig benötigt - Ticketverkauf), sekundär (notwendig, selten benötigt - Nachfüllen der Belegrolle druch Mitarbeiter), optional (nützlich, aber nicht notwendig - Aufladen des Handyguthabens wäre ein nice-to-have) >

Vorbedingung < erwarteter Zustand, bevor der Anwendungsfall beginnt > Eine Bestellung muss da sein, um diese bearbeiten zu können

Nachbedingung Erfolg < > Geld ist einkassiert, Ticket wurde ausgegeben

Nachbed. Fehlschlag < > Paper Jam - Der Drucker verklemmt, kriegt das Ticket nicht mehr raus - Geld muss wieder ausgegeben werden !

Akteure < Alle Akteure, die den Anwendungsfall ausführen >

Auslösendes Ereignis < > Der Kunde wählt eine Zone

Beschreibung < Hier wird der Standardfall beschrieben (Aus Sicht des Benutzers) >

Schritt 1: Benutzer wählt Zone

Schritt 2: Automat zeigt Preis an

Schritt 3: Benutzer gibt das Geld

…..

Erweiterungen < Erweiterung des Funktionsumfangs der ersten Aktion > Schritt 1a; Schritt 1b; Schritt 1c (Beispiel: Kunde will mit Kreditkarte bezahlen)

Alternativen < Alternative Ausführung der ersten Aktion > Schritt 1a; Schritt 3a; Schritt 3b;

Aus einem Use Case Template kann direkt ein Test Case gemacht werden

Akteur = keine Person, sondern eine Rolle (mehrere Personen wollen ein Getränk kaufen, nicht nur eine) - Symbol = Strichmännchen

Akteure stehen immer ausserhalb eines Systems