Objektorientiertes Projekt

- Bis Ende der 80er Jahre wurden strukturierte Methoden zur Softwareerstellung eingesetzt
- Vorteil von strukturierten Methoden waren eine genau definierte Vorgehensweise für die Verfeinerung bei der Analyse einer Aufgabe
- Gegen Ende der 80er Jahre erkannte man, dass mit wachsender Komplexität rein methodische Ansätze bei der Softwareentwicklung nicht mehr hinreichend zum Ziel führten.
- → Veröffentlichung mehrerer objektorientierter Methoden
- Bei vielen verschiedenen objektorientierten Methoden war es schwierig zwischen den verschiedenen Projekten Vergleiche anzustellen oder über diese zu kommunizieren
- Ab Mitte der 90er Jahre startete man den Versuch einer einheitlichen objektorientierten Vorgehensweise → Entwicklung von **UML**

UML (Unified Modeling Language) History

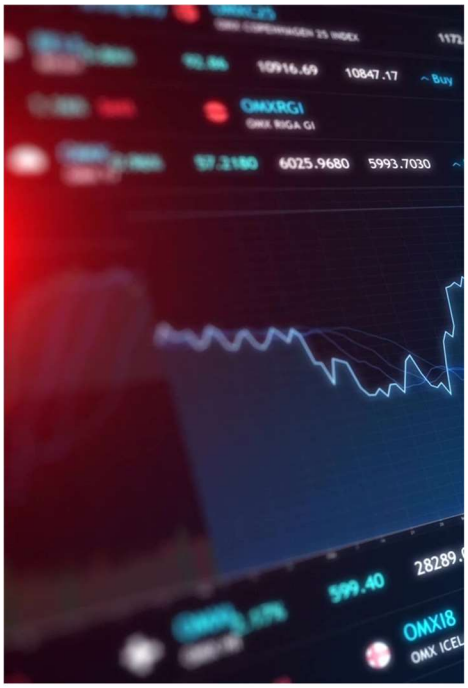
- UML bietet eine einheitliche Notation um objektorientierte Paradigmen darzustellen
- UML wurde von den drei Autoren James Rumbaugh, Grady Booch und Ivar Jacobsen entwickelt
- Erstes Ergebnis ihrer Arbeit im Oktober 1995 → UML Version 0.8
- Nach UML 0.9 wurden immer mehr Unternehmen in die Entwicklung von UML mit einbezogen

UML

- UML ist eine grafische Sprache zur Beschreibung von Softwaresystemen
- UML versucht dabei eine einheitliche Notation für alle Einsatzbereiche zu definieren
- UML dient heute als Standard in der Analyse und dem Design objektorientierter Anwendungen
- Damit kann man verschiedene Anwendungen (Datenbankanwendungen, Echtzeitanwendungen, ...) einheitlich darstellen
- UML besteht aus verschiedenen Diagrammen – dabei definieren die Diagramme verschiedene Betrachtungsweisen der Design - und Implementierungstätigkeiten im Projekt
- Grundsätzlich kann man die Diagramme in 2 Diagrammtypen unterteilen: Strukturdiagramme und Verhaltensdiagramme
- Diagramme besitzen spezifische Elemente deren Semantik genau festgelegt sind

UML Diagrammtypen

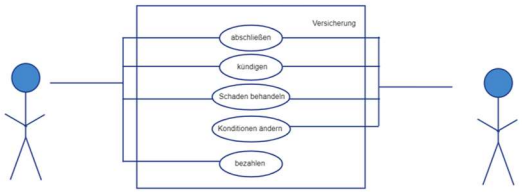
- Betrachtungsweise: Requirements (Anforderungen an das System)
  - Use-Case-Diagramm (Anwendungsfalldiagramm) Akteure, Szenarios
- Betrachtungsweise: Statische Sicht (logischer Aufbau des Systems)
  - Klassendiagramm
    Klassen, Beziehungen
  - Paketdiagramm
    Strukturierung der Darstellung
  - Kollaborationsd (Kommunikationsd)
    Zusammenwirken von Komponenten
- Betrachtungsweise: Dynamische Sicht (Interaktionen, Abläufe)
  - Aktivitätsdiagramm
    Ablaufmöglichkeiten
  - Sequenzdiagramm
    Objekte, Interaktionen
  - Zustandsdiagramm
    Internes Verhalten von Objekten
- Betrachtungsweise: Implementierung
  - Komponentendiagramm
    Innere Struktur von Objekten
  - Verteilungsdiagramm
    Einbettung von Objekten in eine Umgebung



Use-Case-Diagramm

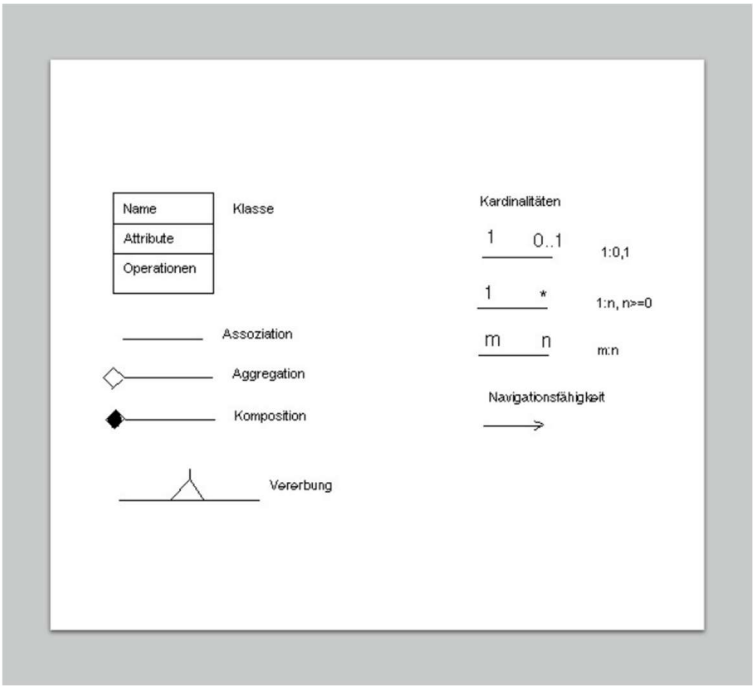
- Beschreibt das Zusammenwirken von Personen mit einem System
- Use-Case ist eine typische Handlung, die ein Benutzer mit dem System ausführt
- Use-Cases werden mit Ellipsen dargestellt
- Verbindung zwischen Use -Cases und Aktoren werden mit Linien dargestellt
- Aktoren werden nach Rollenverhalten unterschieden
- Bilden die Grundlage eines Systems

UML – Use-Case-Diagramm Beispiel

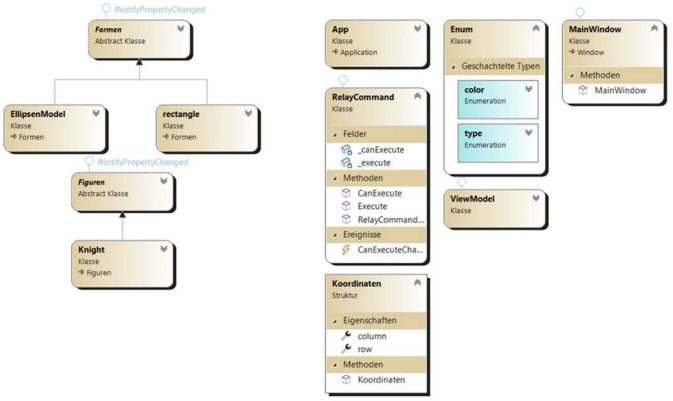


Klassendiagramm

- Beschreibt die Struktur der Objekte und ihre Beziehung zueinander
- Zwei wesentliche Elemente:
  - Objekt: Gegenstand mit präziser Bedeutung
  - Klasse: Beschreibt die Eigenschaften eines Objektes
- Die wichtigsten Elemente eines Klassendiagramms



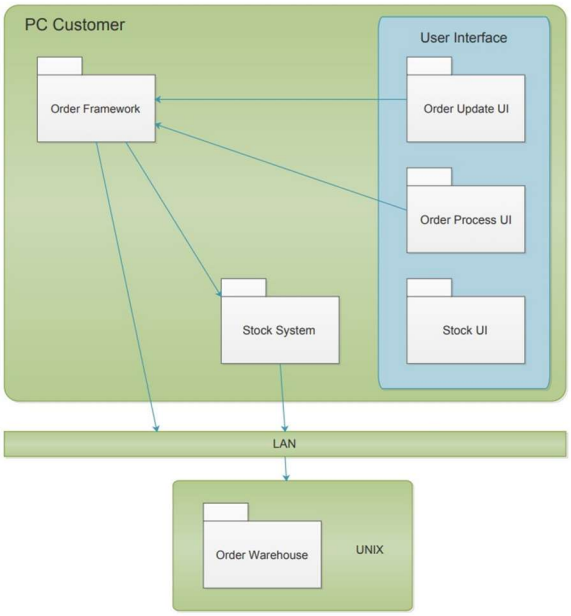
Klassendiagramm - HTK



Paketdiagramm

- Gibt einen guten Gesamtüberblick über das gesamte System
- Fasst Gruppen von Elementen bzw. auch Diagrammen zusammen
- Paketdiagramm besteht aus Paketen und Abhängigkeiten
- Abhängigkeit gibt an, dass ein Paket an einem Ende der Abhängigkeiten sich evt. ändern muss, sollte sich ein Paket an einer Pfeilspitze ändern
- Paketdiagramme werden oft zur Organisation von Klassen- und Anwendungsfalldiagrammen verwendet.

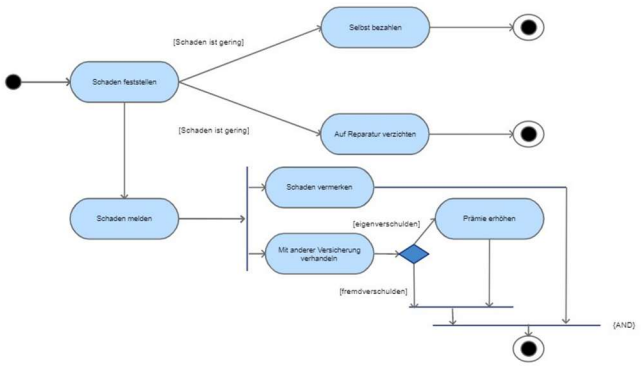
Paketdiagramm Beispiel



Aktivitätsdiagramm

- Aktivitätsdiagramme werden verwendet, um den Ablauf verschiedener Aktivitäten un Aktionen in ihrem Zusammenspiel innerhalb eines System zu veranschaulichen
- In Geschäftsmodellierungsprozessen als auch in der Softwareentwicklung weit verbreitet.
- Eine Aktivität ist ein einzelner Schritt innerhalb eines Programmablaufs
- Eine Aktivität wird durch ein Rechteck mit abgerundeten Ecken dargestellt
- Durch einen Übergang geht es von einer Aktivität in die nächste bzw. in den Abschluss der Interaktion

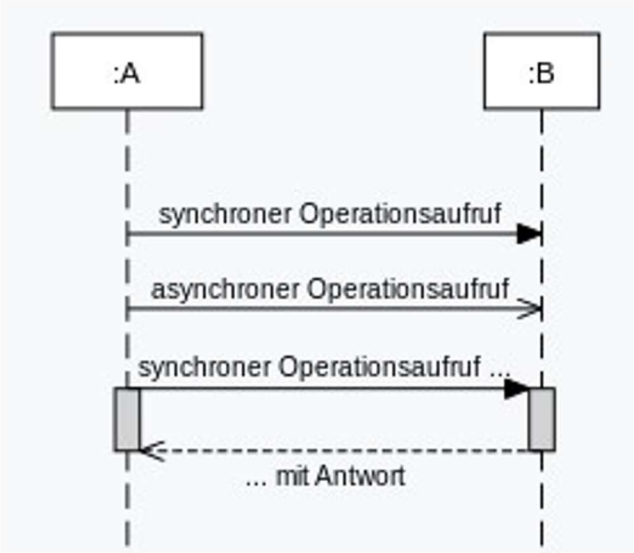
Aktivitätendiagramm



Sequenzdiagramm

- Zeigt die zeitliche, sequentielle Abfolge zwischen Interaktionen und Objekten
- Wird in der Softwareentwicklung zur Darstellung der Architektur verwendet
- Wird durch Szenarios erstellt – dabei konzentriert man sich auf die wichtigsten Fälle

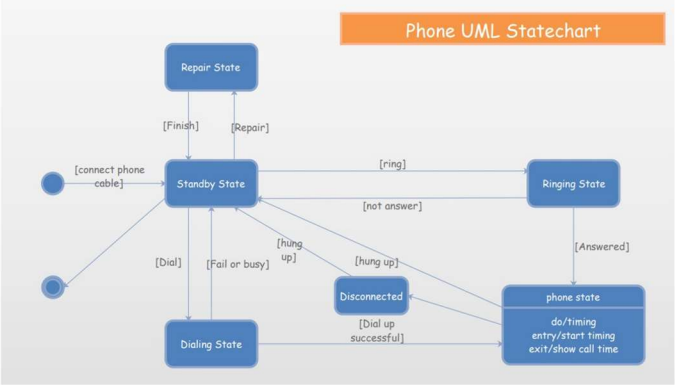
Sequenzdiagramm



Zustandsdiagramm (State-Transition-Diagram)

- Visualisieren unterschiedliche Zustände, die Objekte in ihrem Leben annehmen können
- Ein Zustandsdiagramm beschreibt eine sogenannte Statemachine
- Es gibt endlich viele Zustände, einen Anfangs - und mindestens einen Endzustand

Zustandsdiagramm



Komponentendiagramm

Eine Komponente stellt ein Stück Programmcode dar

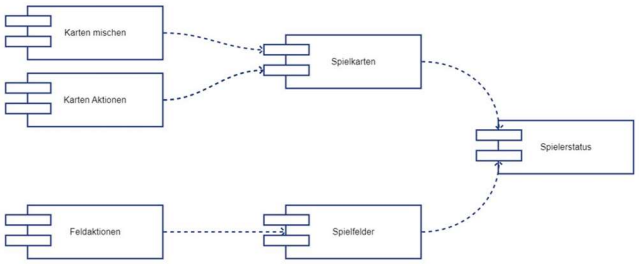
Kann ein Quellcode, Binärcode oder Teil eines ausführbaren Programms sein

Verbindungen zwischen den Abhängigkeiten stellen die Abhängigkeit voneinander dar

Die Darstellung eines Komponentendiagramms dient zum Beispiel zur Darstellung der Kompilierungsreihenfolge, oder auch um Implementierungsdetails zu überprüfen

Komponentendiagramm

Komponentendiagramm



Verteilungsdiagramm

- Zeigen wie Komponenten konfiguriert sind und welche Abhängigkeiten bestehen
- Es gibt Knoten (z.B. unterschiedliche Arten von Servern) und Artefakte (z.B. Client-/Datenbankschema)
- Verschiedene Artefakte können zu einem Knoten gehören

Verteilungsdiagramm