2. Der Software-Entwicklungs-zyklus

2.1 Klassische Phasenmodelle

2.1.1 Wasserfallmodell

2.1.2 Rapid Prototyping

2.2 Objektorientierte Phasenmodelle

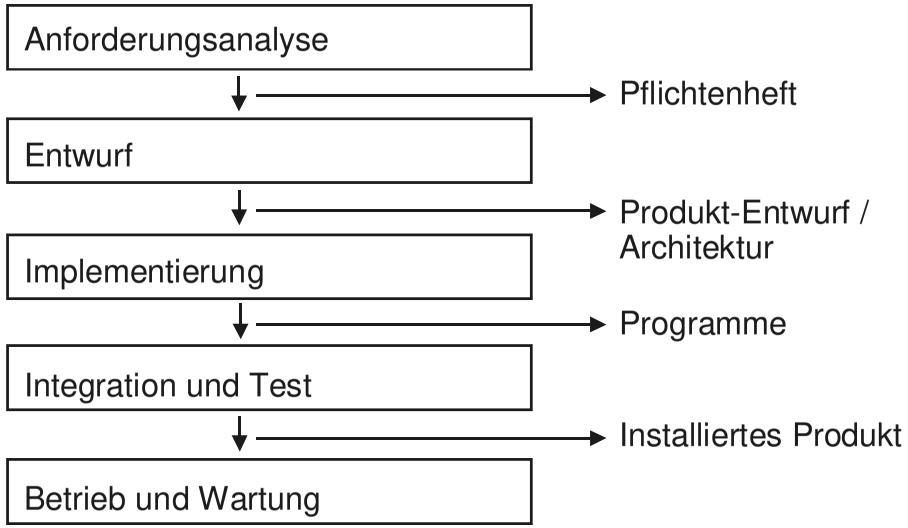
2.2.1 OOA / OOD / OOP

2.2.2 Iteratives Phasenmodell

2.2.3 Exkurs: Anwendungsfälle (OOA)

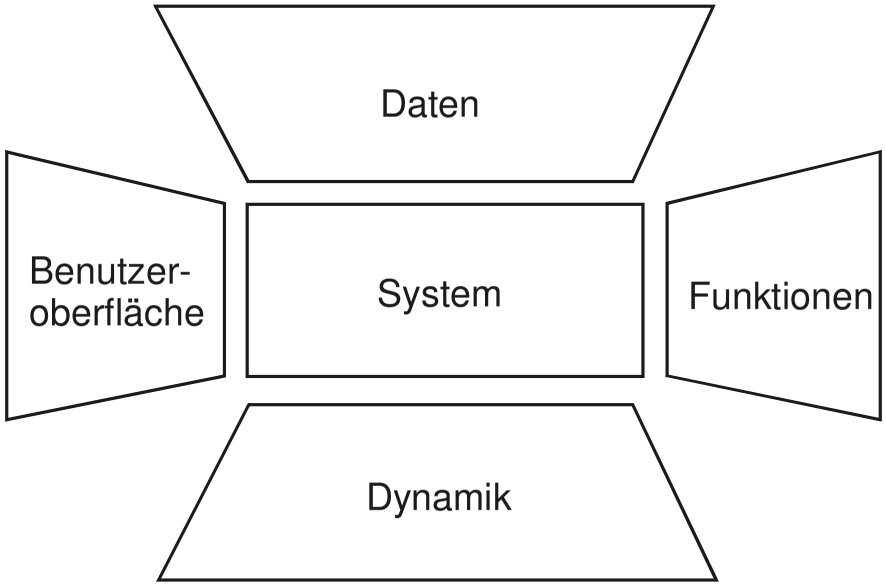
2.1 Klassische Phasenmodelle

2.1.1 Das Wasserfallmodell



Anforderungsanalyse

Was muss definiert werden?



Pflichtenheft und Spezifikationen

Modellierung

Wie wird etwas definiert bzw. notiert?

- Text ist unübersichtlich und wenig formal.
- Daher sind graphische Notationen entwickelt worden.

Diagramme haben Vorteile:

- Sie sind formaler und damit nicht so mehrdeutig / missverständlich.
- Sie sind kompakter und oft leichter kommunizierbar.
- Sie sind evtl. (semi-) automatisch in Programmcode umsetzbar.

Sichten auf ein Softwareprodukt

Notationen für Sichten auf ein Softwareprodukt

- **Benutzersicht**
  - "Gemalte" Bildschirmansichten
  - Use-Cases (UML)
- **Funktionale Sicht**
  - Funktionsbaum
  - Klassendiagramme (UML)
- **Datenorientierte Sicht**
  - Jackson-Diagramme
  - ER-Diagramme
  - Klassendiagramme (UML)
- **Dynamische Sicht**
  - Petri-Netze
  - Interaktionsdiagramme (UML)
  - Aktivitätsdiagramm (UML)
  - Zustandsübergangsdiagramme (UML)

Aufbau des Pflichtenhefts (1)

1. Zielbestimmung

1.1 Musskriterien
1.2 Wunschkriterien
1.3 Abgrenzungskriterien (was nicht erforderlich ist)

2. Produkt-Einsatz

2.1 Anwendungsbereiche
2.2 Zielgruppen
2.3 Betriebsbedingungen

3. Produkt-Umgebung

3.1 Software
3.2 Hardware
3.3 "Orgware"
3.4 Produkt-Schnittstellen

4. Produkt-Funktionen

Je Funktion ein Unterkapitel. Funktionen aus Benutzersicht beschreiben (WAS jedes Modul leistet und NICHT wie)

5. Produkt-Daten

Aufbau des Pflichtenhefts (2)

6. Produkt-Leistungen

7. Benutzeroberfläche
Bildschirmlayout, Drucklayout,
Tastaturbelegung, Dialogstruktur, Ton

8. Qualitäts-Zielbestimmung

9. Globale Testszenarien

10. Entwicklungs-Umgebung
10.1 Software
10.2 Hardware
10.3 “Orgware“
10.4 Entwicklungs-Schnittstellen

11. Ergänzungen/Sonstiges

Benutzerhandbuch

Aufgabe
Handhabung des Softwareprodukts beschreiben

Adressaten
Endbenutzer

Gibt es bei großen Softwaresystemen verschiedene Klassen von Endbenutzern (z. B. Anwender, Systemverwalter), so sollten getrennte Benutzerhandbücher geschrieben werden.

Stile
- User Guide
- Reference Card
- Tutorial
- Online-Hilfe

Entwurfsphase

1. **Spezifikation der Architektur des Gesamtsystems**
   - Festlegung der Hardware und Software
   - Definition von Schnittstellen zur „Außenwelt“ (Benutzerschnittstellen, Netzwerkschnittstellen usw.)
   - Aufteilung der Funktionalität auf Komponenten (Module)
   - Definition von Schnittstellen zwischen den Komponenten
2. **Spezifikation der einzelnen Komponenten (Module)**
   - Spezifikation der Algorithmen und Datenstrukturen für jedes einzelne Modul

Dabei Einsatz einer geeigneten Spezifikationssprache / **Notation** und einer geeigneten Entwurfs**methode**. In diesem Praktikum: UML (Unified Modeling Language).

Implementierungsphase

- Weitere Zerlegung des Gesamtsystems in Module
- Algorithmen und Datenstrukturen festlegen
- Aufteilung der Arbeit auf die einzelnen Programmierer
- Codieren

Integration und Test

- Abnahmetest der einzelnen Module mit möglichst umfassenden Testfällen
- Integrationstest (Zusammenfügen der Module zum Gesamtsystem und Gesamttest)
- Dokumentation der Testergebnisse

Betrieb und Wartung

Betriebsübergabe

- Installation beim Kunden
- Aufnahme des operativen Betriebs
- Schulung von Mitarbeitern beim Kunden

Ergebnis

- Installiertes Produkt
- Gesamtdokumentation
- Abnahmeprotokoll

Wartung

- Stabilisierung, Fehlerbehebung
- Leistungsverbesserung
- Kleinere Anpassungen, Erweiterungen, Ergänzungen

2.1.2 Rapid Prototyping

Ein gravierender Nachteil des Wasserfallmodells ist, dass man nach Abschluss einer Phase nicht mehr zu ihr zurück kehren kann. Oft zeigen sich aber Probleme erst in der nächsten Phase oder gar erst beim Testen oder im Betrieb der Software. Deshalb gibt es viele Varianten des Wasserfallmodells, die eine Rückkehr in eine frühere Phase erlauben. Ein Beispiel ist das Spiralmodell von Boehm.

Im Extremfall geht man möglichst schnell von den Anforderungen zu einem ersten Prototyp. Dieser besteht oft vorwiegend aus simulierten Benutzeroberflächen. Er hat vor allem das Ziel, dem Auftraggeber die Funktionen des endgültigen Systems vorzuführen. Denn dabei zeigt sich oft, dass das Pflichtenheft den Wünschen des Auftraggebers nicht wirklich entspricht, weil er sich die Funktionsweise der Software im voraus nicht gut vorstellen konnte.

Man nennt diese Software-Entwicklungsmethode **rapid prototyping**.

2.2 Objektorientierte Phasenmodelle

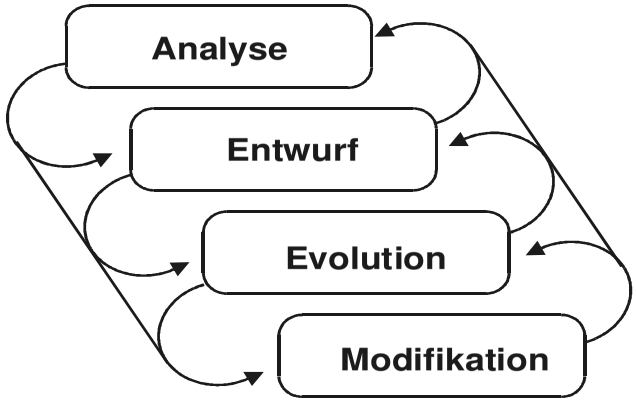
Bei der Objektorientierung werden (anders als bei klassischen Prozeduren und Funktionen) Daten und Bearbeitungsanweisungen zu **Objekten** zusammen gefasst (gekapselt). Das hat den Vorteil, dass erforderliche Änderungen lokal ausgeführt werden können und somit überschaubar sind.

Die Objektorientierung erleichtert die Zerlegung von großen Softwaresystemen in Komponenten und die Wartung der einzelnen Module. Sie verbessert auch die Wiederverwendbarkeit der Module.

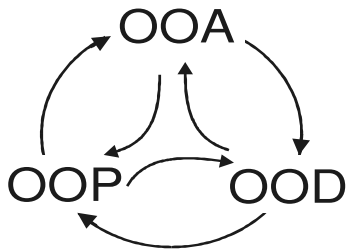
Objektorientierte Phasenmodelle und Entwurfsmethoden „denken“ von Anfang an in Objekten. Objekte aus der Domäne der Anwendung werden auf DV-technische Objekte abgebildet.

2.2.1 OOA / OOD / OOP

Die Modelle von Booch und Coad/Nicola



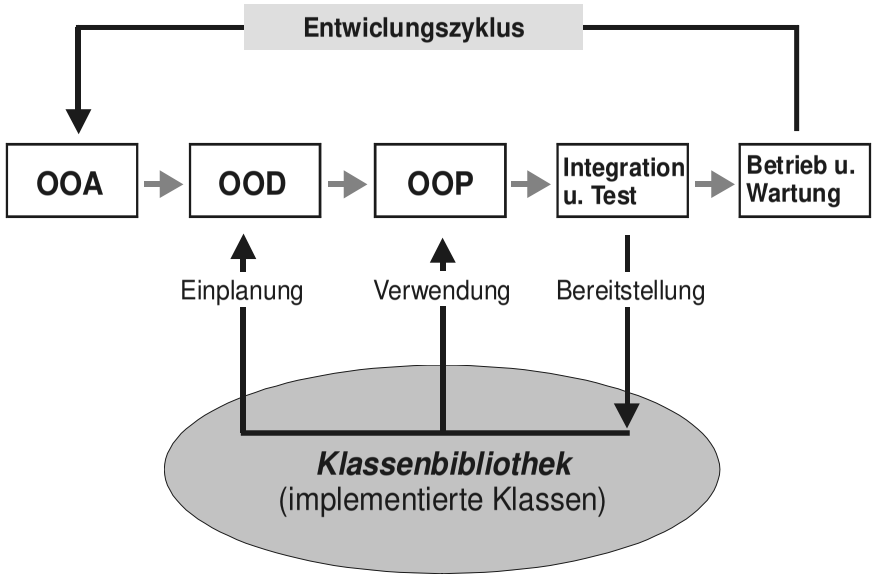
Evolutionäres Modell nach G. Booch



“Baseballmodell” nach Coad/Nicola

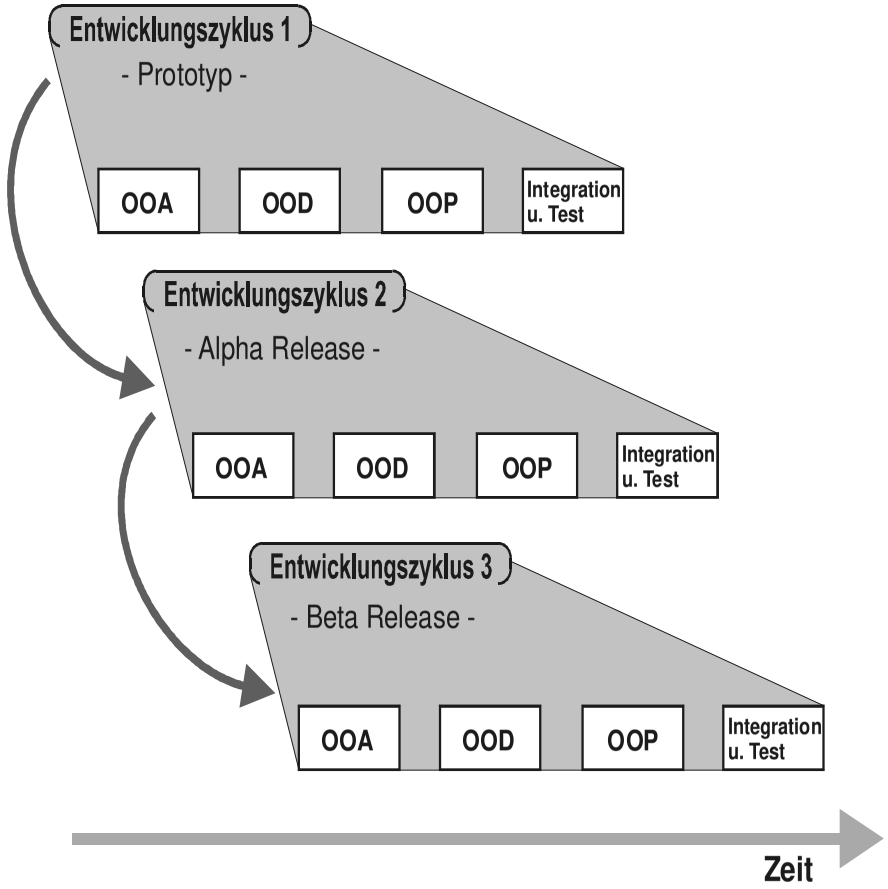
Wiederverwendbare Module im Entwurfsprozess

Objektorientierung verbessert die Wiederverwendbarkeit erheblich. Deshalb werden bereits existierende, wieder verwendbare Module bereits beim Entwurfsprozess berücksichtigt. Dies geschieht insbesondere durch Einbeziehung von *Klassenbibliotheken* und *Frameworks*.



2.2.2 Iteratives Phasenmodell

Die Entwicklungsphasen von Release i und Release i+1 überlappen sich.



2.2.3 Exkurs: Anwendungsfälle

Was sind Anwendungsfälle (Use Cases)?

- Beschreibungen von typischen Abläufen innerhalb des zu entwickelnden Systems.
- Sie umfassen in der Regel mehrere Szenarios, die durch ein gemeinsames Benutzerziel verbunden sind.

Was läßt sich aus Anwendungsfällen ermitteln?

- Anwendungsfälle liefern einen Überblick über die notwendige Funktionalität.
- Mit ihrer Hilfe lassen sich einzelne Komponenten sowie Interaktionen zwischen Komponenten identifizieren.
- Anhand der abgedeckten Anwendungsfälle lässt sich der Implementierungsfortschritt dokumentieren.

Struktur eines Anwendungsfalls

Anwendungsfall: Bezeichnung des Anwendungsfalls

Ziel: Verfolgtes Benutzerziel

Vorbedingungen: Was muss gewährleistet sein

Ablauf (die einzelnen Schritte)
1. ...
2. ...

Alternativen (mögliche Alternativen zu obigen Schritten)
2a. ...
2b. ...

Anwendungsfälle: Ein Beispiel (1)

Ein Beispiel aus Sicht des CluedoClients

Szenario 1: Erfolgreiche Anmeldung eines Spielers

Ein Spieler möchte sich bei einem CluedoServer anmelden. Er wählt einen entsprechenden Server aus und gibt einen Benutzernamen und ein Passwort an. Das System nimmt die Anmeldung vor und meldet, dass sie erfolgreich verlaufen ist.

Szenario 2: Misslungene Anmeldung eines Spielers

Ein Spieler möchte sich bei einem CluedoServer anmelden. Er wählt einen entsprechenden Server aus und gibt einen Benutzernamen und ein Passwort an. Das System nimmt die Anmeldung vor und meldet, daß sie aufgrund eines ungültigen Passworts nicht erfolgreich verlaufen ist.

Anwendungsfälle: Ein Beispiel (2)

Zusammenfassung obiger Szenarios zu einem Anwendungsfall

Anwendungsfall: Anmelden eines neuen Spielers

Ziel: Anmeldung des Spielers bei einem CluedoServer

Vorbedingungen: Der Spieler hat einen CluedoClient gestartet

Ablauf

1. Der Spieler gibt einen Benutzernamen und ein Passwort ein. Des weiteren wählt er einen Cluedo-Server aus.
2. Das System nimmt Kontakt mit dem Server auf und meldet den Spieler an. Der Spieler erhält eine Bestätigung, dass die Anmeldung erfolgreich war.

Alternativen

2a. Der Server hat die Anmeldung aufgrund eines ungültigen Passworts abgelehnt. Der Spieler wird zur erneuten Eingabe aufgefordert.

2b. Der Server nimmt keine weiteren Spieler mehr an. Das System gibt dieses an den Spieler weiter.