

1. Phasenmodell

Auftraggeber

(Teil der) Weltsicht des Auftraggebers

Was soll ins Modell?

Produktdefinition (Pflichtenheft)

Wie? - soll es realisiert werden
(Auftragnehmersicht, abstrakt, Sprachenunabhängig)

Softwarespezifikation (Systemarchitektur)
(gleiche Parameter, gleiche Struktur)

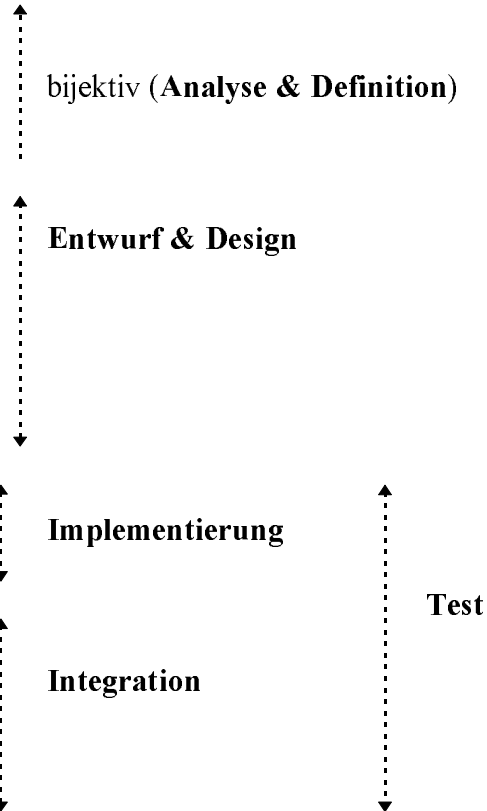
Syntax, Sprachkonstrukte

Einzelmodule

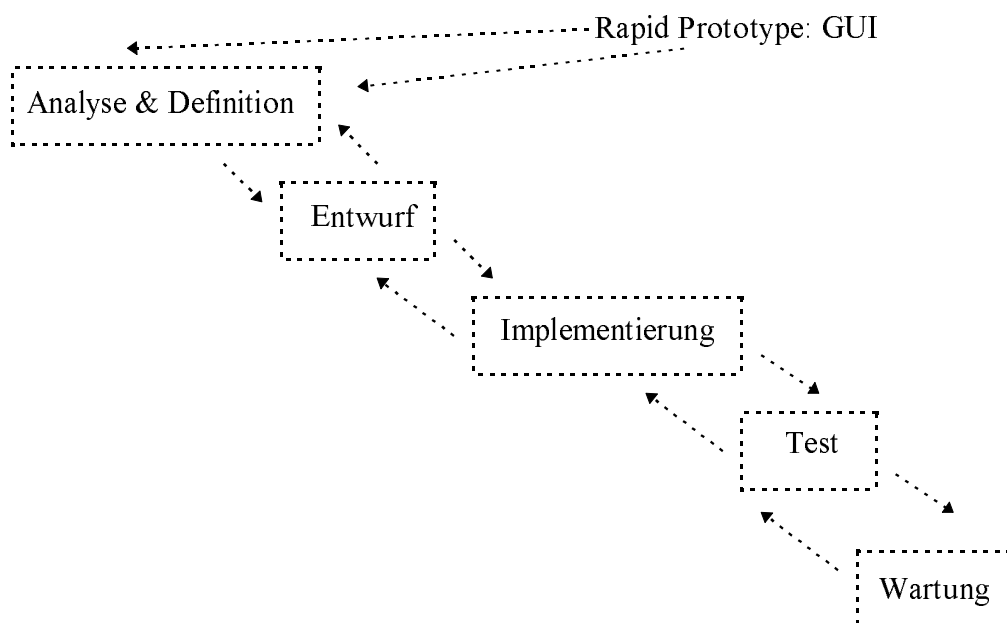
Reihenfolge

System

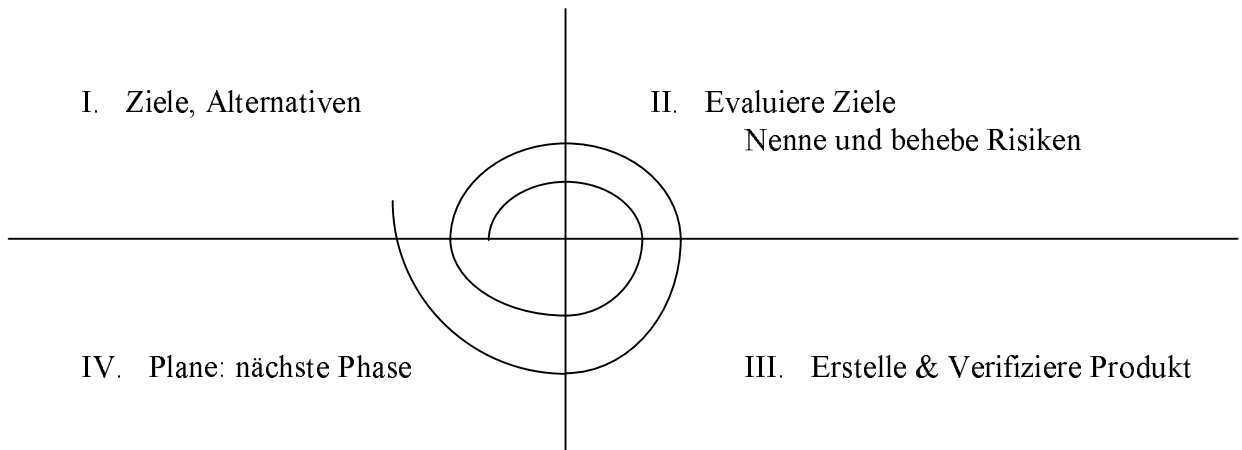
Auftraggeber



-> auch Wasserfallmodell genannt!



2. Spiralmodell (Barry Böhm, TRW, 1988)



Evolutionäre Softwareentwicklung:

- iteiertes Phasenmodell
- Abfolge von „Proto“-typen (z.B. zuerst GUI, ...)
- geht im Wartungsbereich weiter

Transformationelle Softwareentwicklung:

formale Spezifikation



mathematisch korrekt

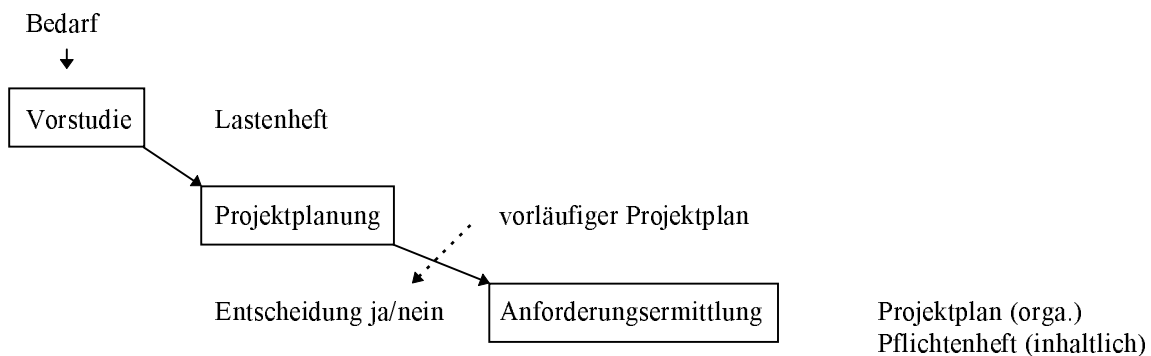
fertiges System

-> sehr aufwendig, bei großen Systemen unmöglich!

Qualitätskriterien im Betrieb (Wartungsphase):

- Effizienz
- Benutzerführung
- Ergonomie
- Dokumentation
- Altdaten übernehmen
- Zuverlässigkeit (Pflichtenheft erfüllt (korrekt); Fehlertoleranz (robust))
- Wartbarkeit

Analyse:



Vorstudie (Lastenheft)

Analytiker (Auftragnehmer)
+ Benutzer (Auftraggeber)

WAS soll das Produkt leisten?
WER ist betroffen?
WANN - sollte
- muß das Produkt fertig sein?

Ziel: grobe Kostenschätzung
technische Machbarkeit (ja/nein)

Projektplanung

Vorgehensmodell

Meilensteine (komplett abgeschlossener Vorgang), administrative Kontrolle (druck AG)

Aufwandschätzung

#PM (Anzahl der Personenmonate) -> CoCoMo (Constructive Code Model)
(+Gemeinkosten)
-> Gesamtkosten (Personal)
(+Kosten für Hardware)

#PM -> wann, wieviele? => Erfahrung aus früheren Projekten ist zuverlässiger als formale Methoden

=> vorläufiger Projektplan

Durchführbarkeitsstudie - organisatorisch
- wirtschaftlich

Endgültige ja/nein-Entscheidung des AG

CoCoMo: < komische Formeln - siehe Skript !!! >

Anforderungsermittlung

- | | | |
|------|------------------|--|
| II. | operativ | Schnittstellen, Daten, Funktionalitäten, Ausnahmebehandlung |
| III. | Qualität | zuverlässig, wartbar, effizient, benutzerfreundlich => quantitativ!
(MTBF (Mean Time between failure), #Transaktionen/Std., GUI Masken <= 10
Zeilen) |
| IV. | technisch | Hardware Umfeld, Betriebssystem, Programmiersprache |
| V. | validierung | Testfälle, Abnahmetest |
| VI. | Realisierung | Vorgehensmodell, Dokumentation, Personal, Geld, Zeit,
Vorschriften (ISO, Din, AG-spez.) |
| VII. | Wiederverwendung | (alter Software -> AN ; spätere Erweiterungen -> AG) |

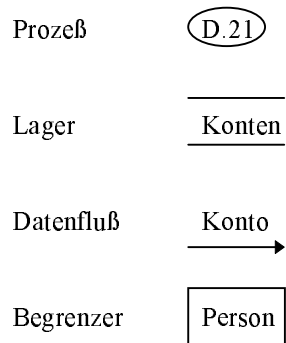
II operativ

Strukturierte Analyse

- Datenflussdiagramme (Prozesse, Lager, Flüsse)
- Datenmodell (ER, relationale DB)
- Zustandsautomaten (Ereignismodellierung)
- Datenlexikon

=> Balancierung zur MSA (Modernen Strukturierten Analyse - 1989)

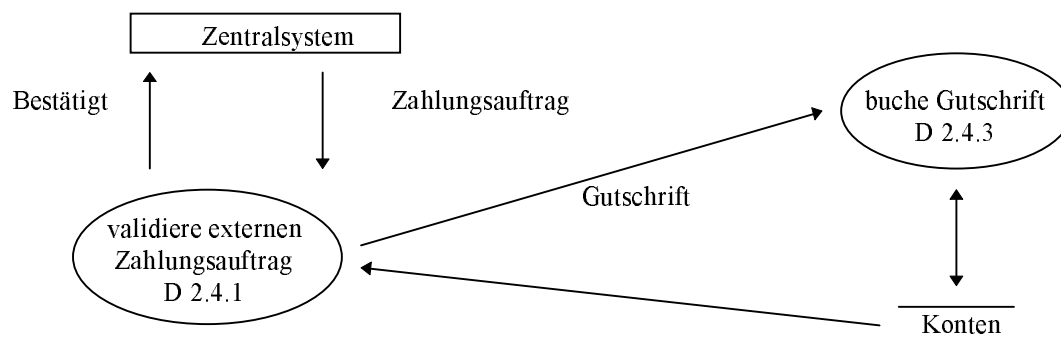
DFD (Daten Fluß Diagramm)



Prozeß als Begrenzer

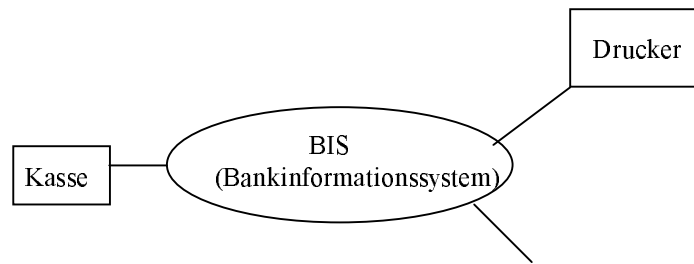
Prozeß DB

z.B.: Prozeß D 2.4 - Tagbuchungsbearbeitung

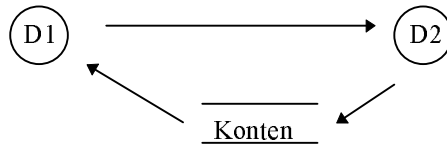


Kontextdiagramm

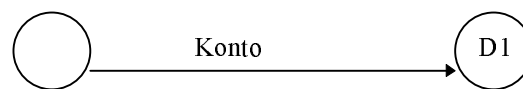
“das System“ + Begrenzer



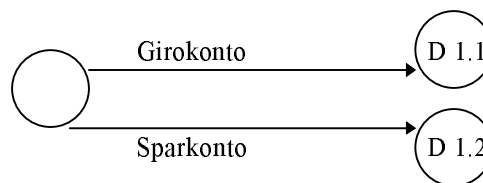
BIS:



Flüsse aufspalten (disjunkt!)



...



- Namen aus dem Datenlexikon!
- Prozeßname = Verbindung + Substantiv

Prozeßspezifikation

nicht: Flußdiagramm
Nassi-Shn.

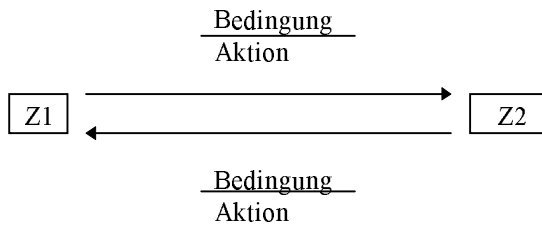
Besser: strukturierte Sprache

gut: Vorbedingungen
Nachbedingungen

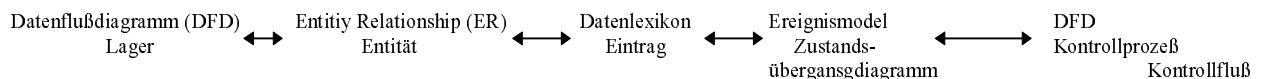
CASE-Tools

Moderne Strukturierte Analyse (MSA)

- Datenflußmodell
- ER (DB)
- Zustands- / Ereignismode



MSA: Balancierung



< Abb. 11.19 und weitere ... >

Lastenheft (Produktskizze):

- I. Problembeschreibung, Projektziele
- II. Funktionsumfang, Außenverhalten
- III. Benutzerprofil
- IV. Akzeptanzkriterien
- V. Entwicklungs-, Einsatz- und Wartungsumgebung, Schnittstellen, Nebenbedingungen
- VI. Lösungsstrategien
- VII. Informationsquellen (z.B. Ansprechpartner, Manuals), Glossar

Pflichtenheft (Produktdefinition):

- korrekt
- vollständig
- konsistent
- eindeutig
- funktional („was“ und nicht „wie“)
- verifizierbar (Spezifikation der Kundenwünschen; müssen vom System erfüllt werden)
- verfolgbar (ableitbar und systemspezifisch)
- leicht änderbar (indiziert, Querverweise)

Gliederungsvorschlag der Produktdefinition (Pflichtenheft):

- I. Einleitung: Ziele, Überblick, Zusammenfassung
- II. Operative Anforderungen: Beschreibung der Funktionalität und Daten des Softwaresystems
- III. Qualitätsanforderungen (z.B. Effizienz und Benutzerfreundlichkeit)
- IV. Technische Anforderungen
- V. Validierungsanforderungen

VI. Realisierungsanforderungen

VII. Hinweise für den Entwurf: Lösungsvorschläge, vorhersehbare Änderungen und Erweiterungen

VIII. Informationsquellen, Index, Glossar

Das Pflichtenheft ist die rechtliche Grundlage für die Projektausführung.

Funktionspunkte:

Lastenheft -> Funktionspunkte -> MannMonate -> Kosten

<u>5 Kategorien:</u>	einfach	mittel	komplex
Eingabedaten	(1-5 Felder) 3	(6-20 Felder) 4	(> 10 Felder) 6
Abfragen	3	4	6
Ausgaben	4	5	7
Datenbestände	7	10	15
Referenzdaten (=Festdaten)	5	7	10

R o h - F u n k t i o n s p u n k t e

Einflussfaktoren:

- Verflechtung mit anderen Anwendungssystemen (0-5)
- dezentrale Daten / dezentrale Verarbeitung (0-5)
- Transaktionsrate (0-5)
- Wiederverwendbarkeit in anderen Systemen (0-5)
- Datenbestandskonvertierungen (0-5)
- Anpassbarkeit (0-5)
- Verarbeitungslogik
 - Rechenoperationen (0-10)
 - Kontrollverfahren (0-5)
 - Ausnahmeregelungen (0-10)
 - Logik (DBMS) (0-5)

0 - 60 Punkte

$$FP = \text{Roh-FP} * (70 + EF) / 100$$

(70 + EF) / 100 liegt zwischen 0,7 und 1,3

$$MM (\text{MannMonate}) = f (FP)$$

$$MM = (FP / 25) ^ 1,3 \quad \text{bei IBM !}$$

oder/und:

$$FP = \text{Rohfunktionspunkte} * \text{Einflussfaktoren}$$

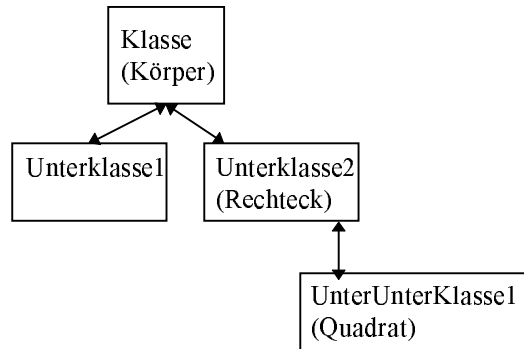
$$\text{Roh-FP} = \sum \text{Anzahl} * \text{Faktor}$$

$$MM = (FP / \text{Fähigkeit}) ^ (1,0 + \text{Bürokratie})$$

OOA (Objekt orientierte Analyse)

- Klasse (Attribute, Operationen/Methoden)
- Objekt
- Typ
- Variable

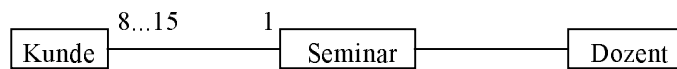
- Vererbung



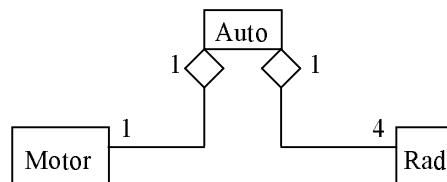
- Polymorphie

Beziehungstypen

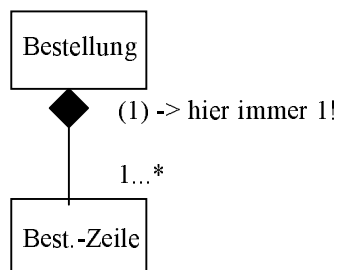
- Assoziation:



- Aggregation:

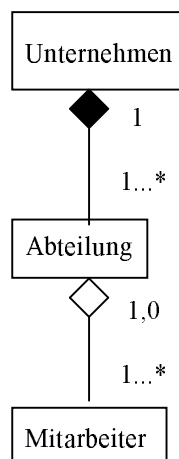


- Komposition:
(Existenz-Abhängigkeit)



-> Teile können nicht für sich alleine stehen -> wird das „Ganze“ gelöscht, müssen die Teile ebenfalls gelöscht werden.

Weiteres Beispiel:



(bei Menschen ist eine Komposition unwahrscheinlich, weil Teile nicht existenzabhängig vom Mensch sind)

Voraussetzungen für „gute“ SW-Architektur:

- fest definierte Schnittstellen
- niedrige äußere Kohärenz (Kopplung/Abhängigkeiten zwischen Schnittstellen)
- Architektur-Komponente verfügt über hohe innere Kohärenz (logisch zusammenhängendes Teilproblem)
- Komponente < 10 Personentage (Entwurf und Test)
- Außenverhalten der Komponente bleibt gleich, unabhängig von der inneren Realisierung (Wartbarkeit, Fehlerlokalisierung)

Bestandteile eines Moduls:

- (Export)-Schnittstelle - nach außen sichtbar
- Modulrumpf (Kapselung, Geheimnisprinzip)

Beispiel Modulaufbau:

Modulspezifikation <Modulname>

Schnittstellenspezifikation

Exporte

Konstanten
Datentypen
Operationen

Semantik (Vor-/Nachbedingung; Beschreibung der Operationen (Implementierung Methoden))

Rumpfspezifikation (Datenstrukturen, Algorithmen)

Spezifikationsende Modul <Modulname>

Datentypen werden in der Regel verborgen exportiert (Struktur im Rumpf verborgen)

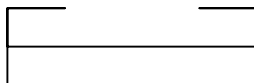
OO-Sprachen: private, public, protected

< wieder einige wirre Beispiele - siehe Skript >

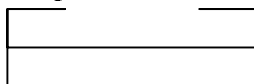
Abstrakter Datentyp (ADT) - Abstraktes Datenobjekt (ADO)

ADT

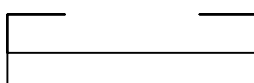
Belegstapel ADT



Belegliste ADT



Beleg ADT



-> zu den ADT's gehören ADO's (Instanz, Ausprägung, Exemplar)

-> Datenkapselung

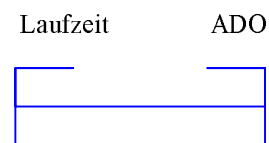
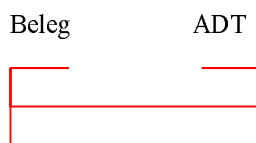
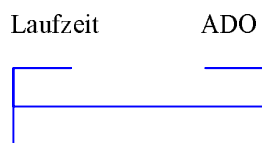
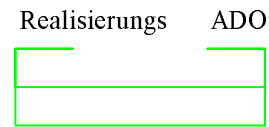
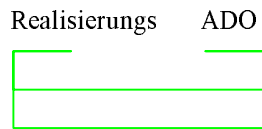
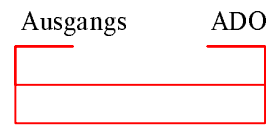
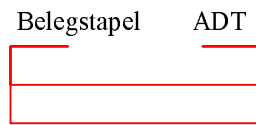
- Architektur ADO
(Teil des SW-Architekturmodells;
Lebensdauer bis Programmende)

- Laufzeit ADO
(werden dynamisch (new/delete) erzeugt)

- Realisierungs ADO
(implizit zur effizienten Realisierung;
nicht modelliert, ADT aber schon)

Beispiel:

Zugriff von außen nur über Eingangs-/Ausgangs ADO evtl. Belegstapel (allg.) - der Rest ist nach außen nicht sichtbar!



Instanzen (Eingang)

Modell „Klasse“

Instanzen (Ausgang)

Übergang Analyse -> Entwurf

Zuerst ADO's / ADT's aus ER-Diagramm in Module fassen. Danach funktionale Zerlegung (Prozesse) entweder in diese ADT's (Operationen) integrieren oder als Funktionsmodul (FM) realisieren.

Modularisierung bzgl. ADO/ADT vermeidet Redundanzen.

- Vorteile guter Modularisierung:
- Geheimnisprinzip fördert Wartbarkeit und Wiederverwendbarkeit
 - senkt Komplexität (im Gesamtbild sind nur die Softwareschnittstellen, nicht der Rumpf sichtbar)
 - Verkapselung = Sicherheit und Zuverlässigkeit (kein unkontrollierter Zugriff)
 - Exportschnittstelle eindeutig -> Rumpf kann unabhängig verändert werden.

Glossar:

Destruktor	Besondere Methode einer Klasse, die beim Löschen der Instanz automatisch aufgerufen wird
Konstruktor	Besondere Methode einer Klasse, die beim Erzeugen der Instanz automatisch aufgerufen wird
Polymorphismus	Der Tatsache, daß eine abgeleitete Klasse nichts anderes ist als die um Eigenschaften erweiterte Basisklasse, trägt Polymorphismus in der Art Rechnung, daß ein Zeiger vom Typ der
Basisklasse	auch auf eine Instanz der abgeleiteten Klasse zeigen kann, nicht jedoch umgekehrt.
Queue	Auf deutsch auch „Schlange“ oder „Warteschlange“ - ein abstrakter Datentyp (ADT) mit den Operationen enqueue (Element anhängen) und dequeue (Element entfernen). Queues sind FIFO-Strukturen.
Rein virtuelle Funktion	Eine leere Funktion, die in einer abgeleiteten Klasse durch eine neue Funktion ersetzt werden muß. Klassen mit mit rein virtuellen Funktionen sind abstrakte Klassen.
Stack	„Stapel“ - ein abstrakter Datentyp (ADT) mit den Operationen Push (Element auf den Stack legen) und Pop (Element vom Stack holen). Stacks sind LIFO-Strukturen.
Überladen ihrer	Als Überladen bezeichnet man das Definieren mehrerer gleichnamiger Funktionen, die sich in Parameterliste und/oder Rückgabewert unterscheiden. Ein Unterschied allein im Rückgabewert reicht nicht aus.
Virtuelle Funktion	Eine Funktion, die bei Bedarf in einer abgeleiteten Klasse durch eine neue Funktion gleichen Namens und mit gleichen Parametern ersetzt werden kann. Virtuelle Funktionen sind nötig für die dynamische Typüberprüfung

Literaturverzeichnis:

André Wilms: C++ Programmierung, Addison Wesley, 2. Auflage 1998