www.highsource.de Mario Kretschmer

Software -Engineering

Bin => HEX

<= 2^2 2^1 2^0

1 1 1 0	0 1 0 1	0 1 1 1
8+4+2+0	0+4+0+1	0+4+2+1
Dez 14	5	7
Hexa E	5	7

16^2	16^1	16^0
E	5	7

256*14
=
3584 + 80 + 7

256 :2 = 128 Rest 0
128 :2 = 64 Rest 0
64 :2 = 32 Rest 0
32 :2= 16 Rest 0
16 :2= 8 Rest 0
8 :2= 4 Rest 0
4 :2= 2 Rest 0
2 :2= 1 Rest 0
1 :2= 0 Rest 1

Bei Umrechnung von Dezimalenzahlensystem wird immer die Basis durch die Dezimalzahl geteilt egal ob 2, 8 oder 16 und der Rest ergibt die Zahl.

Softwarekrise von 1968 =>

Was macht Softwareentwicklung problematisch

- Software ist immateriell => keine physikalischen Gesetze, nicht messbar
- Software unterliegt einem starken moralischen Verschleiß = vorausschauend, Anpassbarkeit
- Portierbarkeit auf unterschiedliche Hardware-/ Softwareplattform
- Akzeptieren der Software durch Benutzer
- Wenig Standards, Normen für Software nicht wie bei der Industrie (DIN- Normen)

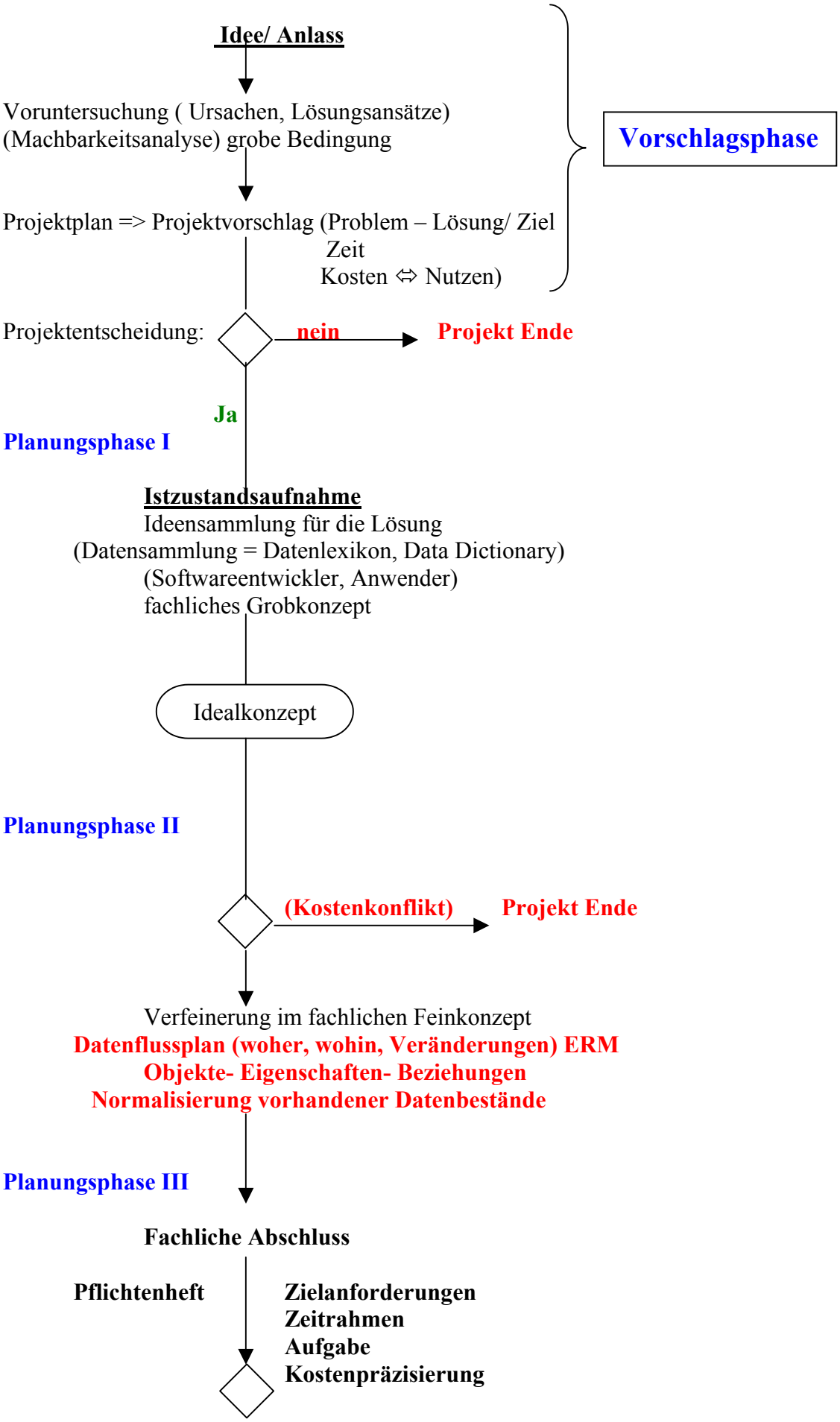
Software – Engineering

Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software – Systemen

Zielorientiert: Plan also der Weg zum Ziel, Kosten, Personalaufwand, Zeit
Systematisch: die Verschiedenen Entwickler müssen darauf achten das die Schnittstellen zu den anderen Entwicklergruppen passen

Software – Qualitätskriterien

- **Korrektheit** = entspricht Software den Anforderungen der Zielvorgabe (laut Pflichtenheft)
- **Zuverlässigkeit** = Software leistet über einen bestimmten Zeitraum fehlerfrei ihre Aufgaben
- **Robustheit** = -Absturzsicherheit,
  -Reaktion auf Benutzerfehler
  -unvorhergesehene Ereignisse wie (Stromausfall)
- **Vertrauenswürdigkeit** = Software darf im Fehlerfall keine „Katastrophen“ verursachen
- **Effizienz** = Auslastung der Ressourcen (Hardware, Zeit, Einsparung von Kosten)
- **Benutzerfreundlichkeit** = -klare Oberflächenstrukturen
  -unterschiedliche
  -Auswahloptionen (Menü, Tastatur, Symbolleisten, Befehle)
  -Hilfesysteme
- **Wartbarkeit** = -Einfache Fehlersuche durch modularen Aufbau
  -Austauschbarkeit der Module im Zusammenhang mit veränderten Anforderungen
  -Übersicht durch Dokumentation


Idee/ Anlass
Voruntersuchung (Ursachen, Lösungsansätze)
(Machbarkeitsanalyse) grobe Bedingung
Projektplan => Projektvorschlag (Problem – Lösung/ Ziel
Zeit
Kosten ⇔ Nutzen)
Vorschlagsphase
Projektentscheidung:
nein
Projekt Ende
Ja
Planungsphase I
Istzustandsaufnahme
Ideensammlung für die Lösung
(Datensammlung = Datenlexikon, Data Dictionary)
(Softwareentwickler, Anwender)
fachliches Grobkonzept
Idealkonzept
Planungsphase II
(Kostenkonflikt)
Projekt Ende
Verfeinerung im fachlichen Feinkonzept
Datenflussplan (woher, wohin, Veränderungen) ERM
Objekte- Eigenschaften- Beziehungen
Normalisierung vorhandener Datenbestände
Planungsphase III
Fachliche Abschluss
Pflichtenheft
Zielanforderungen
Zeitrahmen
Aufgabe
Kostenpräzisierung

→ **Ende**

IV- Grobkonzept

Programmablaufplan

Strukturierung des Programms

- **Ein- , Ausgabe –Masken**
- **Speicherung**
- **Sicherungserfordernisse**
- **Bereitstellung von und Zugriff auf Daten**
- **Schnittstellenprobleme**
- **Modularisierungsüberlegungen**
  - **Wiederverwendung von Modulen**
  - **Redundanzen Vermeidung**
- **Objektdefinitionen (Objektorientierte Programmierung)**
  - **Objekte**
    - **welche Eigenschaften**
    - **welche Methode (Programme) sollen zur Veränderung genutzt bzw. Programmiert werden**

Feinkonzept

- **unmittelbare Vorbereitung der Programmierung**
- **Schematische Darstellung einzelner Verarbeitungsschritte mittels Case Tools (Nassi- Shneiderman Struktogramm**
- **Programmierung (Codierung)**

Phase V

Test

(Alpha- Phase) mit Testdaten in Testumgebung

- Modulweise
- In Gesamtheit

Phase VI

Implementierung in der Praxisumgebung

- Praxistest (Beta- Phase)
- Schulung
- organisatorische Umstellung

Abnahme/ Übergabe

Pflege & Wartung

Wasserfallmodel

- lineares Model
- Am Ende jeder Phase steht ein Teilprodukt was in der nächsten Phase weiterentwickelt wird (evaluiert) und hat eine Rückkopplung zur vorherigen Phase
- Streng Sequenziell
- Dieses Model wird meist als Metavorlage für andere Vorgehensmodelle gesehen

    - **Vorteile:** geringer Managementaufwand
    - **Nachteil:** Probleme werden häufig erst gegen Ende erkannt

V- Modell

- Weiterentwicklung des Wasserfallmodels
- Immer wieder Qualitätsprüfungen durch das Kreislaufprinzip
- Wird meist im öffentlichen Bereich eingesetzt
    - **Vorteile:** Gut geeignet für groß Projekte
      Kann angepasst und erweitert werden
    - **Nachteile:** Hoher Managementhaufewand
      Somit auch höhere Kosten
      Nicht für kleine Projekte geeignet

Prototyping (siehe DEKA Buch S.30)

Spiralmodel (siehe DEKRA Buch S.31)

Klassifizierung der Programmiersprachen

1. **Generation** Binärcode
    - ***(0 und 1)***
2. **Generation** Maschinennahe Programmierung
    - ***Assembler***
        - Prozessor abhängige Befehle auf Grund des Befehlssatzes über die der Prozessor verfügt.
        - Nicht portierbar
3. **Generation** Problemorientierte Sprachen
    - ***(C, Basic, Pascal, Fortran)***
4. **Generation** Deklarative Sprachen
    - An Umgangssprache angelehnte Befehle, die zur Auslösung von „Programmen“ führen
    - ***Sql***

2 Arten von Modulen (Prozedur)

Prozeduren sollten in mehrere Module (zur Wiederverwendbarkeit) aufgeteilt werden um Fehler schneller zu finden

- Modul (Fehler prüfen) = **Unterprogramme** gibt kein Ergebnis zurück leisten zwar etwas
- Modul (Rechenoperation) = **Funktion** gibt ein Ergebnis zurück

Werkzeuge zur Programmierung

- Editoren
- Compiler
- Debugger

Bezeichner (Namen)

- Zulässig Buchstabe, Ziffer, Unterstrich (keine Leerzeichen)
- Komplette Bezeichnungen (Namen) sind zu verwenden um das Objekt zu identifizieren
- Keine reservierten Wörter (IF, ELSE, WHILE.....) Programmiersprachenabhängig

Kommentare (Programmiersprachenabhängig)

- ' Basic
- // C++
- # Sql

Variable

- Bezeichner (Name) => Adresse im Hauptspeicher
- Datentyp => Hauptspeicherreservierung (Platz)

Deklaration einer Variablen

- **Explizit** (ausdrücklich) = jeder Variablen wird bei Deklaration ein Datentyp zugewiesen
- **Implizit** = die Deklaration erfolgt ohne Datentypenzuweisung. In diesem Fall wird als Datentyp „variant" angenommen und die erste Wertezuweisung entscheidet über den Datentyp. (könnte Fehlerquelle sein!!!)
- **Bei Eingabe von „option explizite" steht Implizit nicht mehr zur Verfügung**
  - Dim eingabe AS integer

Wertezuweisung

- Eingabe = 1686
- Eingabetext =`text`
- Dezimaleingabe = 12.68
- Datumseingabe = JJ/MM/TT

Gültigkeitsregeln von Variablen

Lokale Gültigkeit

- Nur in den Grenzen der Prozedur in der sie deklariert wurden. Bei Beendigung der Prozedur wird der Hauptspeicherbereich freigegeben, Inhalt der Variablen steht nicht mehr zur Verfügung.

Globale Gültigkeit

- Gültigkeit über Prozedurgrenzen hinaus. Inhalt kann in unterschiedlichen Prozeduren benutzt werden

Operatoren

- Arithmetische Operatoren
  +
  -
  *
  / Fehlerprüfen für division durch 0
  % oder MOD (Modulo) Wenn Zahl % 2 Rest 0, dann gerade Zahl sonst ungerade Zahl

Inkrement

- Das erhöhen einer Variablen um 1 Bsp. x=x+1 ; x++ oder (++x erst erhöhen)

Dekrement

- Das vermindern einer Variablen um 1

Vergleichsoperatoren

= (entweder bedeutet es Gleich oder ist steht für eine Zuweisung
= = (bei manchen Progr. Sprachen ist das Gleich)
!= (ungleich)
<> (ungleich)

Logische Operatoren

NOT =>
AND =>
OR =>
XOR => exclusives ODER schließt wahr für beide Bedingungen aus ! siehe Buch S.68
IMP =>Implikation
EQV =>Äquivalenz

Beispiel:

Dim a,b,c,d AS Integer

a= 5
b= ++a + 6 => b=12; a=6
C= 7 => c=7
D= a++ + ++b => a=6; b=13 => d=19
=> a=7
also ist a=7; b=13; c=7; d=19

(a<b or b<c) and not c<d

(7 < 13 or 13< 7) and not 7 < 19
w f w
f
w f
w

Beispiel 2

a= 0 => a= 0
b= a- - => a= 0; b= 0
c= ++a – b++ => a= 0; b= 0; c= 0
d= a * - -b => a= 0 ;b= 0

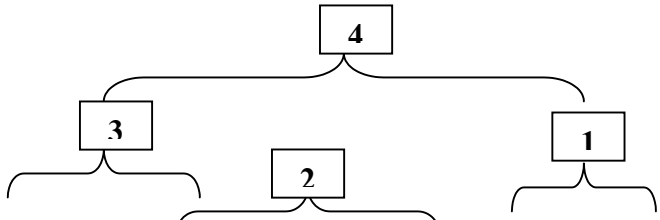
a < 0 and b> a or c> a and not c < ++ a

0 < 0 and 0> 0 or 0> 0 and not 0 < 1

Schritte: **1,2,3** **Weil NOT vor AND und OR**

Beispiel 3

a= 10 => a=10
a= a-- => a=10 - 1
b= ++a – 5 => a=10; b=5
c= b * a-- => c=50; b=5; a=10 -1
d= ++c + --b => **c=51; b=4; d=55; a=9**



a++ < b or b++ > c and d < a-- or not c < --a

7 < 5 or 4+1 > 51 and 55 < 8-1 or not 51 < 8

f or f and f or not f

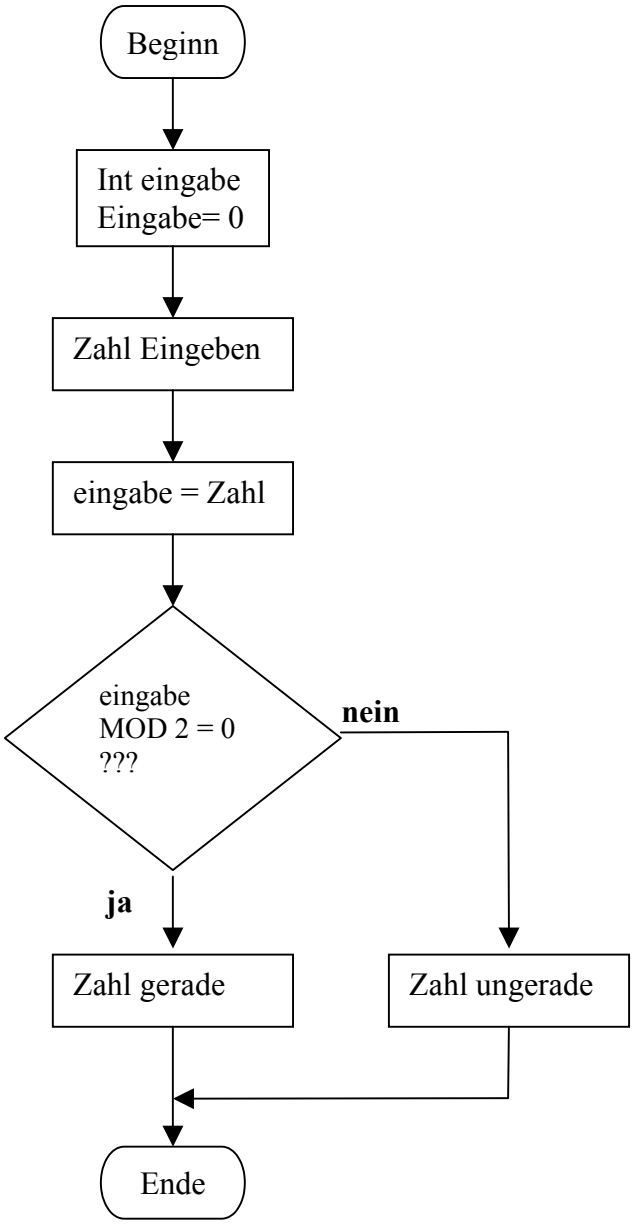
f or f or w

f or w

Ergebnis ist wahr

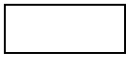
Beispiel Programmablaufplan: EVA Prinzip (Eingabe- Verarbeitung- Ausgabe)

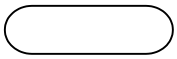
Prüfe ob Eingabe eine gerade Zahl ist



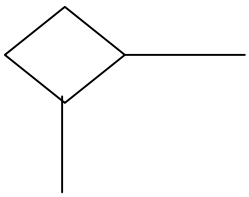
Hilfsmittel zur Darstellung der Programm ablaufe

Programmablaufplan
Logische Abfolge von Programmablaüfen (ALGORYTHMUS)
Symbole zur Darstellung bestimmter Ablaufe

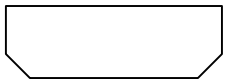
Anweisung 
Eingabe, Ausgabe, Berechnung, Wertezuweisungen, Variablendeklaration

Beginn und Ende 

Bedingungen (Besitzen einen bestimmten Wahrheitswert entweder WAHR o. FALSCH)
Ja und nein sind als und an den Wegen zu kenzeichnen



Wiederholungen (Schleifen) **Anfang =>** 

Ende => 

Schleifen

Schleifenkopf (wird eine so genannte Laufbedingung eingetragen) KOPFGESTEUERT
Schleife wird solange durchlaufen, solange die Laufbedingung WAHR ist

Schleifenfuß

Fußgesteuerte Schleife läuft solang bis Bedingung WAHR wird!
Die in der Schleife stehenden Anweisungen werden mindestenzs einmal ausgeführt, bevor die Laufbedingung geprüft wird.
Solange bis eine Bedingung erfüllt ist

Verbinder 

Konektor (wenn eine Neueseite gebraucht wird) 

Variablen sind als erstes zu definieren da noch daten im Hauptspeicher enthalten sein könnten

Programme:

Ist eine Zahl gerade oder ungerade!

Quelltext:

```
Private Sub StartButton_Click()
   Dim Zahl As Double
         Zahl = 0
   Zahl = EinBox.Text

  If Zahl Mod 2 = 0 Then
  AusBox.Text = "Toll du hast eine GERADE Zahl eingegeben"
  Else: AusBox.Text = "Schade du hast eine UNGERADE Zahl eingegeben"
  End If
EinBox.SetFocus
Einbox.SelStart = 0
EinboxSelLenght = 5
  End Sub
```

MWST aus einem Betrag ermitteln!

Quelltext:

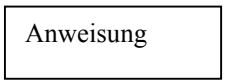
```
Private Sub StartButton_Click()
   Dim Zahl As Double
      Zahl = 0
   Zahl = Betrag.Text
   Dim MWSTa As Double
      MWSTa = 0
   MWSTa = MWST.Text
   Dim Summe As Double
      Summe = 0
  Summe = Betrag * MWST
  AusBox.Text = Summe
   Betrag.SetFocus
   Betrag.SelStart = 0
   Betrag.SelLength = 5
  End Sub
```

Taschenrechner!

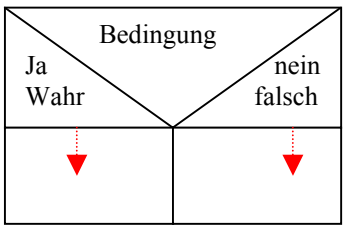
Quelltext:

```
Private Sub Form_Activate()
    Summand1.SetFocus
    Summand1.SelStart = 0
    Summand1.SelLength = 5

    Summand2.SelStart = 0
    Summand2.SelLength = 5
End Sub



Private Sub Gleich_Click()

Dim Sum1 As Double
    Sum1 = 0
    Sum1 = Summand1.Text
Dim Sum2 As Double
    Sum2 = 0
    Sum2 = Summand2.Text
Dim Summ As Double
    Summ = 0
    Summe.Text = Summ

    'Rechnung
    Summ = Sum1 + Sum2
    Summe.Text = Summ

    'Curser Einstellungen
    Summand1.SetFocus
    Summand1.SelStart = 0
    Summand1.SelLength = 5

End Sub
```

Nassi Schneidermann Programmablaufplan!

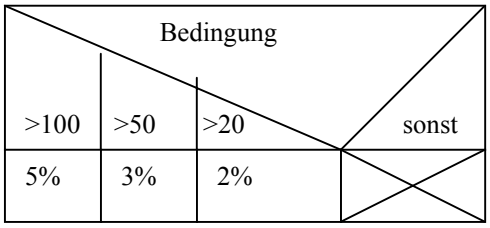


Anweisung (Rechnung, usw)

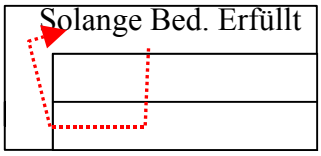


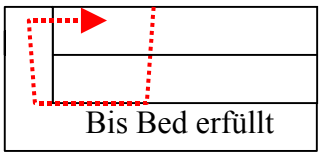
Entscheidung JA / NEIN

Mehrfachverzweigung



Schleife





Einführung in die Programmierung

Objektorientierete Programmierung

Mann hat ein Urobjekt welches nach eigenen wünschen verändert wird, es sollte jedoch darauf geachtes werden die Grundstruktur nicht zu verändern da es sonnst nicht mehr passt (Bsp. LEGO Steine)

Atribute/ Eigenschaften
Sind Werte die bei der Erstellung geändert werden
z.B. Farbe, Größe usw

Methoden (Bsp.Auto)
Beschleunigen
Bremsen

Ereignisorientiert
Gaspedal
Bremse

Sobald das **Ereignis** Gaspedal treten eintritt wird die **Methode** Beschleunigen ausgeführt!

Klassen
Haben Atribute, Eigenschaften, Methoden ist Sourcecode

Klausur

Bereich : Modelle zur Darstellung von Softwareentwicklung

Wasserfallmodel:

- Lienares Modell
- Stufenartig
- Am ende jeder Phase Teilprodukt welches in der nächsten Phase weiterentwickelt wird
- Streng Sequenziell
- Jede Phase besitzt rückkopplung zur vorherigen Phase

Vorteile:

- Geringer Managementaufwand
- Leicht Verständlich

Nachteile:

- Spätere Änderung nur mit hohem Aufwand möglich
- Unflexibel
- Kein Risikomanagement: Probleme werden häufig erst gegen Ende erkannt

V-Modell:

- Weiterentwicklung desWasserfallmodells
- Phasen die weiter ins Detail gehen (Treppenartig versetzt)
- Phase und Testphase bilden Hirarchiche Ebene
- Validierung = Gültigkeitsprüfung und verifikation (Nachweis der Richtigkeit und Korektheit) intigriert
- Kann auch für Hardwareprojekte angewendet werden
- Standartmodell für IT Vorhaben im Öffentlichenbereich

Vorteile:

- Gesammtmodell
- für große Projekte
- viele Aspekte
- Quallitätssicherung im Vordergrund
- Kann Angepasst erweitert werden

Nachteile:

- Sehr allgemein
- Nicht für kleine Projekte
- Casewerkzeuge erforderlich

Prototyping:

- Sind Ablauffähige nicht aber Vollfunktionstüchtige Modelle
- Zur Akquisition für Kunden um ersten eindruck zu bekommen

Vorteile:

- Entwicklungsrisiko wird reduziert
- Intigrierbarkeit
- Lösungsalternativen werden aufgezeigt
- Einbeziehung des Benutzers(Kunde)

Nachteile:

- Prototypen werden mitunter zusätzlich entwickelt und erhöhen Entwicklungsaufwand
- Häufig keine Klaren Beschränkung für Prototypen

Spiralmodell:

- Vereint das klassisch lineare (Wasserfallmodell) und das evulutionäre Modell (Prototyping)

- Softwareentwicklung mittels Sprirallmodell in 4 Schritten (zyklisch)

Vorteile:

- Fehlerfrüherkennung
- Regelmäßige Überprüfung der Zwischenprodukte
- Flexibel
- Risikominimierung
- Alternativen werden betrachtet

Nachteile:

- Hoher Managementaufwand
- Für große Projekte
- Risiken werden nicht konsequent berücksichtigt

SoftwareQuallitätskriterien was versteht man darunter

Software – Qualitätskriterien

- **Korrektheit** = entspricht Software den Anforderungen der Zielvorgabe (laut Pflichtenheft)
- **Zuverlässigkeit** = Software leistet über einen bestimmten Zeitraum fehlerfrei ihre Aufgaben
- **Robustheit** = -Absturzsicherheit,
  -Reaktion auf Benutzerfehler
  -unvorhergesehene Ereignisse wie (Stromausfall)
- **Vertrauenswürdigkeit** = Software darf im Fehlerfall keine „Katastrophen“ verursachen
- **Effizienz** = Auslastung der Ressourcen (Hardware, Zeit, Einsparung von Kosten)

- **Benutzerfreundlichkeit** = -klare Oberflächenstrukturen
  -unterschiedliche
  -Auswahloptionen (Menü, Tastatur, Symbolleisten, Befehle)
  -Hilfesysteme

- **Wartbarkeit** = -Einfache Fehlersuche durch modularen Aufbau
  -Austauschbarkeit der Module im Zusammenhang mit veränderten Anforderungen
  -Übersicht durch Dokumentation

Überblick über die Generation der Programmiersprachen

1. **Generation** Binärcode
   a. ***(0 und 1)***

2. **Generation** Maschinennahe Programmierung
   - ***Assembler***
     - Prozessor abhängige Befehle auf Grund des Befehlssatzes über die der Prozessor verfügt.
     - Nicht portierbar

3. **Generation** Problemorientierte Sprachen
   - ***(C, Basic, Pascal, Fortran)***

4. **Generation** Deklarative Sprachen
   - An Umgangssprache angelehnte Befehle, die zur Auslösung von „Programmen“ führen
   - ***Sql***

5. **Generation**
   - Künstliche Intelligenz
   - LISP, Prolog, SmalTalk
   - Lernfähigkeit der Sprachen

Werte zuweisung von Variablen Text, Zahl, Datum

Wertezuweisung

- Eingabe = 1686 (Ganzzahl)
- Eingabetext =`text`
- Dezimaleingabe = 12.68
- Datumseingabe = JJ/MM/TT (Amerikanisch)

Grundformeln

Ungerade / Gerade Zahl Mod 2
Aufsummieren Summe= Summe+Zahl
Zählervariable lauf = lauf + 1
Max kleinster Wert im Wertebereich des Datentypes
Min größter Wert im Wertebereich des Datentypes