

Zusammenfassung

Anwendungsentwicklung & Programmierung



March 2, 2024

Marius roßgotterer

Table of Contents

[Software-Engineering 2](#_Toc160482736)

[Software 2](#_Toc160482737)

[Software-Produkt 2](#_Toc160482738)

[Aufgabe 2](#_Toc160482739)

[Definition Softwareengineering 2](#_Toc160482740)

[Sechs Qualitätsmerkmale für Softwareprodukte 2](#_Toc160482741)

[Functionality 2](#_Toc160482742)

[Reliability 2](#_Toc160482743)

[Useability 2](#_Toc160482744)

[Efficiency 2](#_Toc160482745)

[Maintainability 2](#_Toc160482746)

[Portability 2](#_Toc160482747)

[Prozess der Softwareentwicklung 3](#_Toc160482748)

[Grundprinzipien der Phasenmodelle 3](#_Toc160482749)

[Compiler & Interpreter 3](#_Toc160482750)

[Ausführen von Programmiersprachen 3](#_Toc160482751)

[Assembler 3](#_Toc160482752)

[Compiler 3](#_Toc160482753)

[Interpreter 3](#_Toc160482754)

[Die Einteilung der Sprachen 3](#_Toc160482755)

[Folgende Sprachen können nicht in kompiliert / interpretiert eingeteilt werden 4](#_Toc160482756)

[Adressen und Zeiger 4](#_Toc160482757)

[Zeigerdeklaration 4](#_Toc160482758)

[Adressoperator & 4](#_Toc160482759)

[Die Bedeutung von Zeigern 4](#_Toc160482760)

[Übergabe als Wert 4](#_Toc160482761)

# Software-Engineering

Forderung nach qualitativ hochwertiger, meist komplexer Software.  
Software-Engineering als Teildisziplin der Informatik.

Software  
Alle nicht physischen Funktionsbestandteile eines Computers bzw. eines technischen Gegenstands.

Software-Produkt  
In sich abgeschlossenes Produkt.

AufgabeDem Anwender ein für seine Zwecke optimales System zur Verfügung zu stellen.

Definition SoftwareengineeringEntwicklung, Einsatz und Pflege von **qualitativ hochwertiger Software** unter **Einsatz von:**

* Wissenschaftlichen **Methoden**
* **Geplanten Vorgehensmodellen**
* **Werkzeugen**
* **Quantifizierbaren Zielen**

## Sechs Qualitätsmerkmale für Softwareprodukte

### Functionality

Funktionalität, Korrektheit, Angemessenheit

### Reliability

Zuverlässlichkeit, Fehlertoleranz, Wiederherstellbarkeit

### Useability

Benutzbarkeit, Verständlichkeit, Bedienbarkeit

### Efficiency

Effizienz, Wirtschaftlichkeit, Zeitverhalten, Verbrauchsverhalten

### Maintainability

Wartbarkeit, Analysierbarkeit, Testbarkeit

### Portability

Übertragbarkeit, Anpassbarkeit, Installierbarkeit, Konformität

Die Lösung **complexer Probleme** erfordert grundsätzlich **systematische Planung**.

## Prozess der Softwareentwicklung

Beschreibung der Softwareentwicklungsprozesse durch **Vorgehensmodelle**, speziell durch sogenannte **Phasenmodelle**.

## Grundprinzipien der Phasenmodelle

* Es gibt mehrere voneinander unterscheidbare Phasen
* Jede Phase verwendet bestimmte Methoden und bringt gewisse Ergebnisse hervor, die als Dokument aufgefasst werden
* Die Phasen laufen nacheinander ab. Rückführungen, die erneute Ausführung von Phasen bewirken, sind aber möglich
* Rückführungen werden als **Iteration** bezeichnet

# Compiler & Interpreter

## Ausführen von Programmiersprachen

Bevor ein Programm auf einem Rechnersystem ausgeführt werden kann, muss dieses erst in die Maschinensprache übersetzt werden.

### Assembler

Assemblercode ist maschinennah, mnemotechnisch aufgebaut und wird in Maschinensprache assembliert. 1:1 Übersetzung in Maschinencode! z.B. MOV ADD INC JMP – Jede CPU-Familie hat einen eigenen Assembler.

### Compiler

Kompilierte Sprachen werden am Stück in selbstständig lauffähige Programme in Maschinencode übersetzt (bezogen auf ein Betriebssystem)

### Interpreter

Skriptsprachen werden zur Laufzeit übersetzt (nicht direkt in Maschinencode, sondern in einen Zwischencode). Sie benötigen zur Ausführung ein Shell- oder Wirtprogramm. Zwischensprache wird auf Wirtprogramm ausgeführt.

## Die Einteilung der Sprachen

|  |  |
| --- | --- |
| Compiler | Interpreter |
| C | Perl |
| C++ | PHP |
| Pascal | VBA |
| COBOL | JavaScript |
| Smalltalk | Python |

### Folgende Sprachen können nicht in kompiliert / interpretiert eingeteilt werden

HTML / XMLSeitenbeschreibungs- (**H**yper **T**ext **M**arkup **L**anguage) bzw. Strukturbeschreibungssprache (E**x**tensible **M**arkup **L**anguage).

#### Java

Wird mit dem Compiler (**J**ava **D**evelopment **K**it) in Bytecode (virtueller Maschinencode, binär) übersetzt, benötigt dann aber eine Java Virtual Machine zur Ausführung (**J**ava **R**untime **E**nvironment).

#### C#, VB.NET

Microsoft .NET Sprachen werden in die **C**ommon **I**ntermediate **L**anguage kompiliert und benötigen zur Ausführung den **J**ust **I**n **T**ime-Compiler des .NET-Frameworks.

# Adressen und Zeiger

short zahl = 5;  
short\* pZahl;  
pZahl = &Zahl;

pZahl zeigt auf die Speicheradresse

Speicher | 0x…100 | 0x…101 | 0x…102 | 0x…103 |

|00000000|00000101|  
 reservierter Speicher für Zahl

**Ein Zeiger ist eine Variable, die eine Speicheradresse enthält!**

### Zeigerdeklaration

Datentyp\* Zeigername;  
z.B. double\* pZeiger;

### Adressoperator &

Liefert die Speicheradresse einer Variable z.B. &zahl;

Zeiger ohne \* -> Adresse: cout << pZahl; //0x…101  
Zeiger mit \* -> Wert: cout << \*pZahl; //5

## Die Bedeutung von Zeigern

### Übergabe als Wert

|  |  |
| --- | --- |
| Speicher | Stack |
| a = 8 | 5 |
| b = 5 | 6 |

Vertausche(a,b); void Vertausche(int a, int b)  
**Im Stack werden die lokalen Kopien verauscht!**a = 8  
b = 5

Vertausche(&a,&b); void Vertausche(int\* a, int\* b)

**Mit Pointern wird auf die Originale zugegriffen  
=> weniger Speicher und Rechenzeitbedarf**