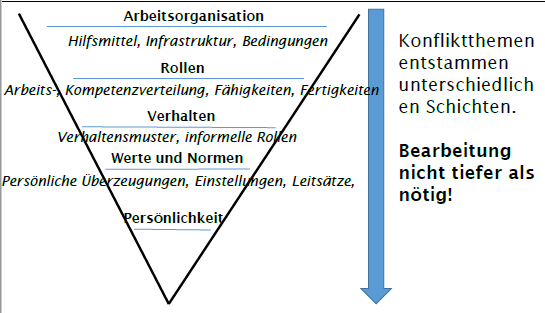
# Theorie

## Aufgabe 1 Kommunikation - Konflikte

**Beschreiben Sie die Grundprinzipien des Schichtenmodells zur Bearbeitung von Konflikten nach Berg.**



## Aufgabe 2 Software Qualität

1. **Nennen Sie genau drei Software Qualitätsmerkmale nach ISO/IEC 9126-1**

**Funktionalität, Effizienz, Zuverlässigkeit, Benutzbarkeit 🡪 Kunde**

**Wartbarkeit, Portabilität, Transparenz, Testbarkeit 🡪 Hersteller**

1. **Welches der Qualitätsmerkmale nach ISO/IEC 9126-1 muss man mit besonderem Bedacht optimieren und warum?**

**Effizienz: Optimierungen führen häufig zu Fehlern in der Implementierung („Zu viel optimiert“)**

## Aufgabe 3 Software Qualität

**In der Programmiersprache C können Ausdrücke wie dieser: „c = a--b;“ zu Fehlinterpretationen führen. Erklären Sie die Greedy Strategie, die hilft, den Ausdruck richtig zu interpretieren.**

Greedy-Strategie: Der Zeichenstrom wird sequentiell in einen Symbolstrom umgewandelt. Ein neues Symbol wird erst begonnen, wenn das alte nicht mehr vergrößert werden kann.

## Aufgabe 4 Software Qualität

**Nennen Sie Zum Thema Optimierungsfehler:**

1. **Zu welchem Zweck werden typischerweise Optimierungen an funktional korrektem Code vorgenommen? Nennen Sie genau zwei Gründe.**

Verbesserung der Performance, Lesbarkeit des Codes

1. **Welche besondere Gefahr besteht bei Code Optimierungen und warum?**

Es besteht die Gefahr, dass sich beim Optimieren Fehler in der Implementierung einschleichen, da „zu viel wegoptimiert“ wird.

1. **Wie können Sie dieser Gefahr entgegenwirken?**

Referenztests, d.h. die alte + neue Implementierung testen und die Ergebnisse vergleichen.

## Aufgabe 5 Code Kommentare

**Betrachten Sie folgenden Code Ausschnitt hinsichtlich der Dokumentation.**

****

1. **Was ist an dieser Dokumentation schlecht?**

Keine genauen Erklärungen, was die Funktion soll. Die Kommentare spiegeln nur wieder, was der Code macht, keine weitere Erklärung dazu.

1. **Wie können Sie die Dokumentation verbessern?**

Beispiel zur Verwendung einfügen, Bessere Erklärung der Funktionalität der Funktion

## Aufgabe 6 Exception Handling

**In Perspektiven des Exception Handlings: Beim Design der Exceptions, die eine Klasse werfen kann, sind verschiedene Perspektiven zu berücksichtigen.**

**a) Wer ist an der Information interessiert, die die geworfene Exception gibt?**

Benutzer, Entwickler

**b) Welche Informationen muss daher aus eine Exception ersichtlich sein?**

Entwickler: Art der Exception, Genaues Auftreten (Stacktrace)

Benutzer: Lokalisierte Nachricht (Für ihn verständlich), was passiert ist.

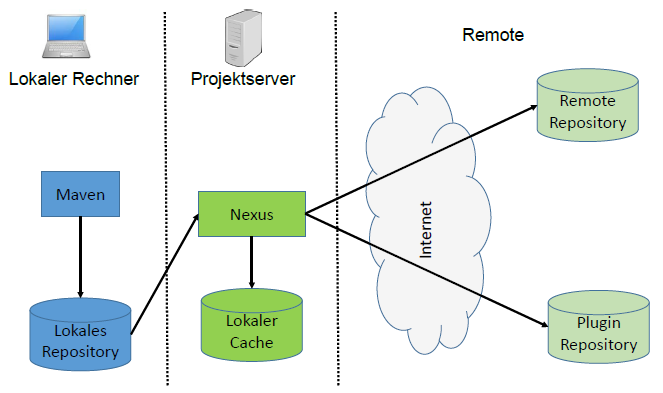
## Aufgabe 7 Continuous Integration

**Sie setzen eine Continuous Integration Infrastruktur mit den Produkten Subversion, Maven, Nexus, Hudson auf.**

1. **Warum benötigen Sie Nexus oder ein ähnliches Produkt?**

* Remote Repositoriesaußerhalb eigener Kontrolle
* Umgang mit lizensierter Software?
* Eigene Artefakte sollen nicht publiziert werden, aber innerhalb der Firma zugänglich sein.
* Repository Manager als Proxy zu den Remote Repositories

1. **Wie arbeiten Nexus und Maven zusammen?**



1. **Anhand welcher Koordinaten lokalisiert Maven Artefakte anderer Projekte?**

* **groupId** 🡪 Gruppe, Organisation, …
* **artifactId** 🡪 Projekt innerhalb der Organisation
* **version** 🡪 Spezifisches Release des Projekts
* **packaging** 🡪 Projekttyp (Optional)

## Aufgabe 8 JUnit

**Erklären Sie die folgenden Begriffe im Zusammenhang mit JUnit**

1. **Testklasse**

Klasse, die einen Test für genau eine Systemklasse beschreibt und diesen ausführt.

1. **TestSuite**

Eine Klasse, die es ermöglicht, mehr als eine Systemklasse gleichzeitig zu testen.

1. **TestFixture**

Dienen zur Vor-/ Nachbereitung der Tests.

## Aufgabe 9 JUnit

**Was sind parametrisierbare Tests mit JUnit und wofür werden sie verwendet?**

Parametrisierter Test: Instanzen für das Kreuzprodukt aus Test Daten Elementen und Testmethoden.

Verwendet zur Vereinfachung von mehreren Tests der gleichen Art mit unterschiedlichen Werten: z.B. Fibonaccizahlen mit 10 verschiedenen Eingaben testen. 🡪 Mit parametrisierbaren Tests über Array möglich.

## Aufgabe 10 Scrum

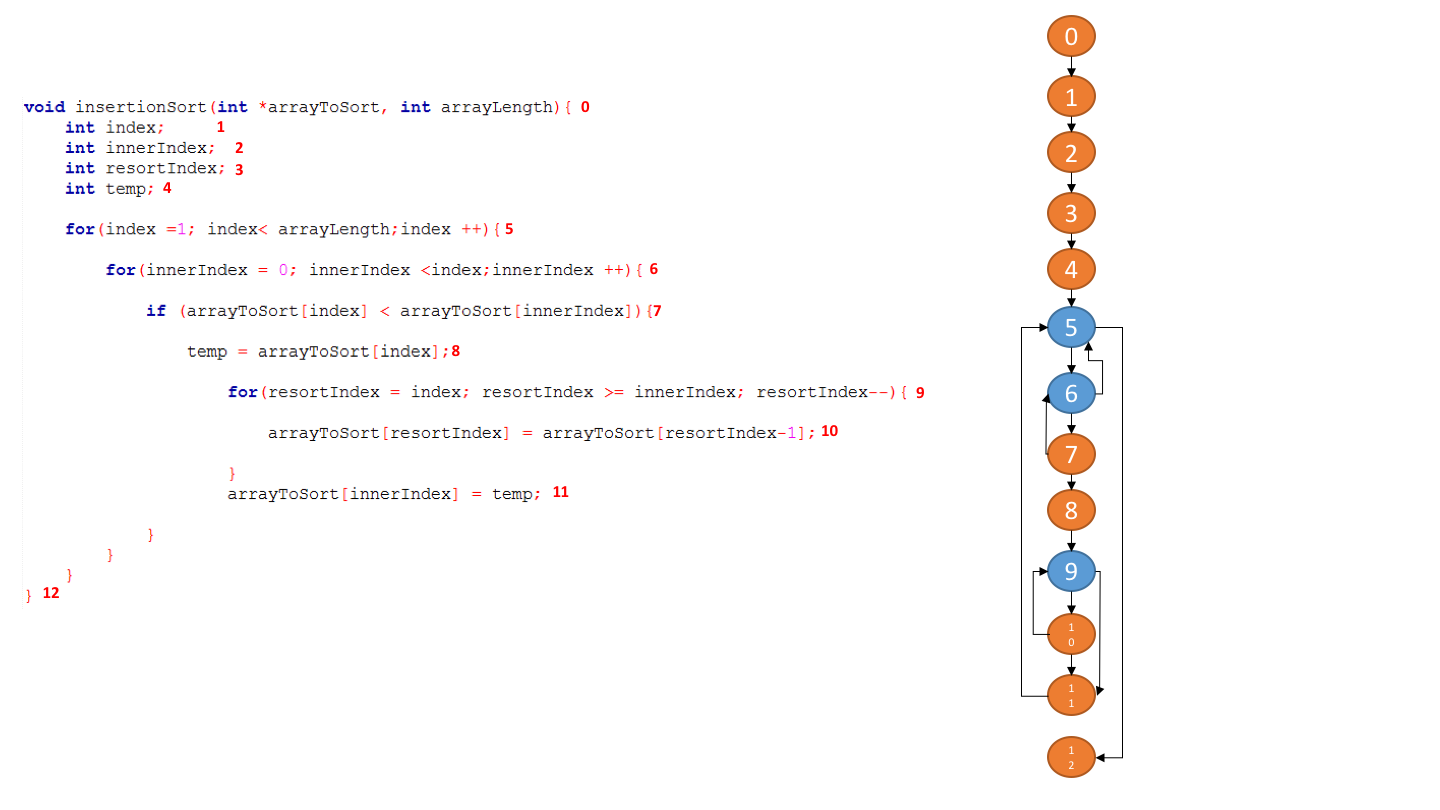
In einem großen Scrum Projekt mit **mehr als einem** Scrum Team müssen sich die Teams abstimmen. An welchen Stellen in Ihrem Sprint können Sie das tun? Nennen Sie 5 verschiedene Möglichkeiten, die sich nicht ausschließen, sondern ergänzen.

* Scrum of Scrums
* ProductOwner-Team
* ScrumMaster Group
* Virtuelle Teams für spezifische Querschnittsaspekte (Architektur, Dokumentation, Tests, …)
* Gemeinsame Planning Meetings
* Evtl weitere Meetings

# Teil 2 Praxis

## Aufgabe 1 Testen

**1. Erstellen Sie einen Kontrollflussgraphen für folgende Funktion.**

**2. Definieren Sie möglichst wenige Testfälle, so dass eine vollständige Anweisungsüberdeckung gegeben ist.**

„Testmenge wird so gewählt, dass alle Knoten des Kontrollflussgraphen mindestens einmal durchlaufen werden.“ 🡪 1 Testfall:

Array mit [2;1] und Länge 2

**3. Definieren Sie Testfälle, so dass eine vollständige Zweigüberdeckung gegeben ist.**

„Jede Kante des Kontrollflussgraphen muss durch mindestens einen Testfall durchlaufen werden.“ 🡪 3 Testfälle:

* x < 0
* x => 0 und x < 2, also 0 <= x < 2
* x > 0 und x >= 2, also x >= 2

**4. Erklären Sie das Konzept der vollständigen Pfadüberdeckung.**

„Dann erfüllt, wenn für jeden möglichen Pfad von Eingangsknoten zu Ausgangsknoten ein separater Testfall existiert.“

**5. Wie viele Testfälle benötigen Sie für das Programm oben für eine vollständige Pfadüberdeckung?**

2n

Könnte man vermutlich genauer analysieren…

**6. Mit welcher Pfadüberdeckungsmethode können Sie diese Zahl einschränken (Name und Erklärung)? Definieren Sie Testfälle entsprechend dieser Methode.**

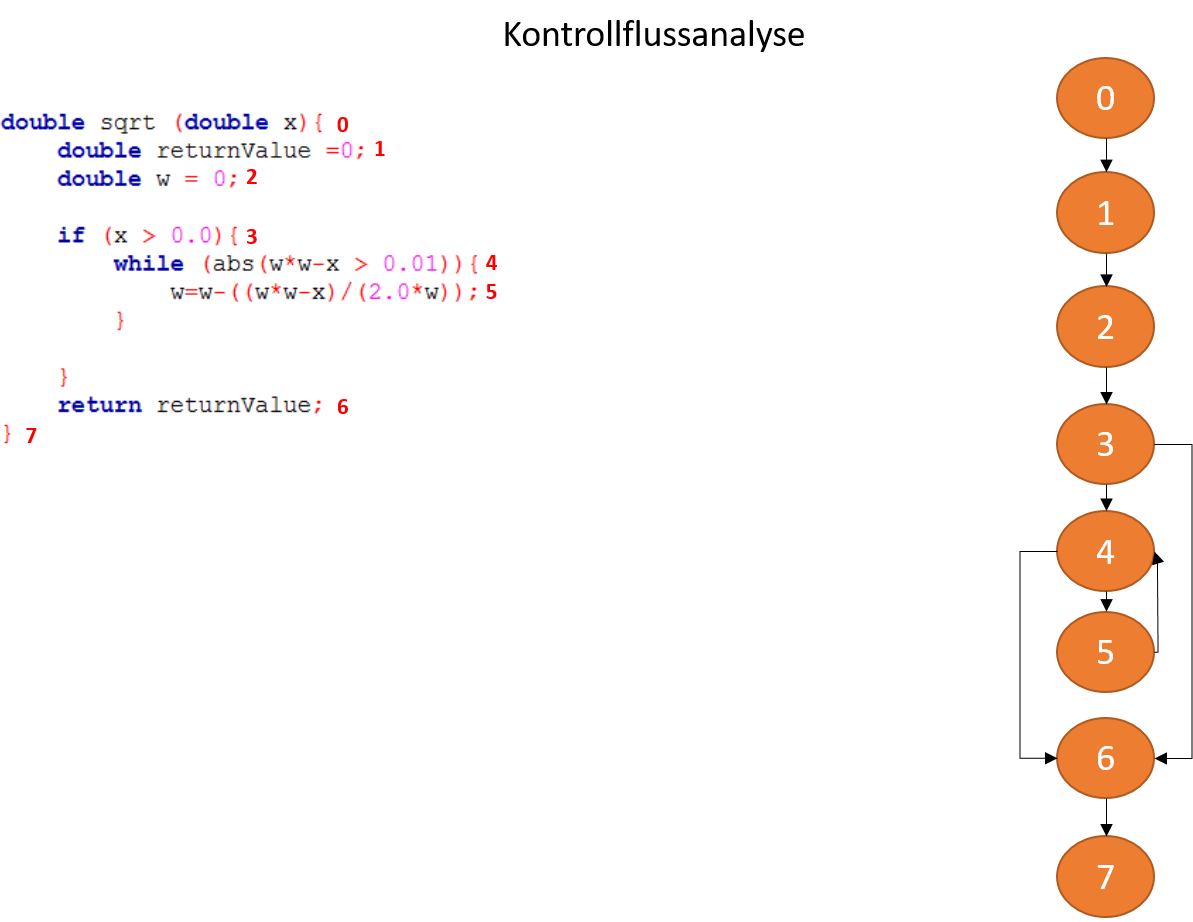
Boundary Interior Pfadüberdeckung:

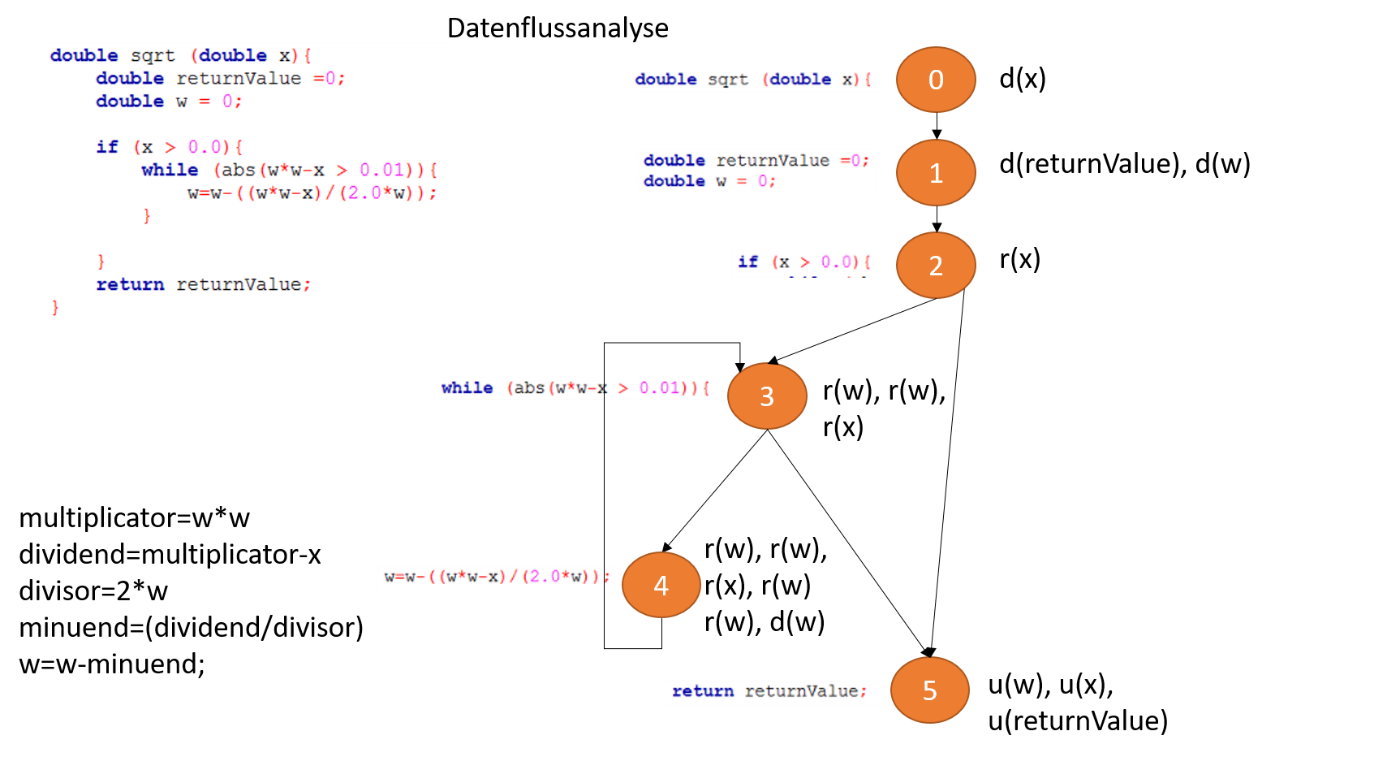
Für jede Schleife drei Gruppen von Testfällen:

* Äußere Pfade: Schleifen werden nicht betreten.
* Grenzpfade: Genau eine Iteration.
* Innere Pfade: Mindestens eine weitere Iteration. Testfälle werden so gewählt, dass innerhalb der ersten beiden Iterationen alle möglichen Pfade abgearbeitet werden.

## Aufgabe 2 Anomalienanalyse

**Erstellen Sie einen Kontrollflussgraphen und eine Datenflussanalyse des folgenden Programms und markieren Sie gegebenenfalls Anomalien.**





|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| x | d |  | r | r | r | u |
| returnValue |  | d |  |  |  | u |
| w |  | d |  | rr | rrrrd | u |

* Anomalie: du (returnValue wird nur definiert, aber nie geschrieben und dann zurückgegeben)

## Aufgabe 3 Anforderungen

**Betrachten Sie folgende Anforderung an ein Softwaresystem:**

***„Das System soll bei Störung gestoppt werden.“***

1. **Benennen Sie zwei typische Fehler, die der Autor der Anforderung bei der Formulierung gemacht hat.**

Keine genaue Spezifikation, was eine Störung genau ist. Benutzerrollen werden nicht verwendet.

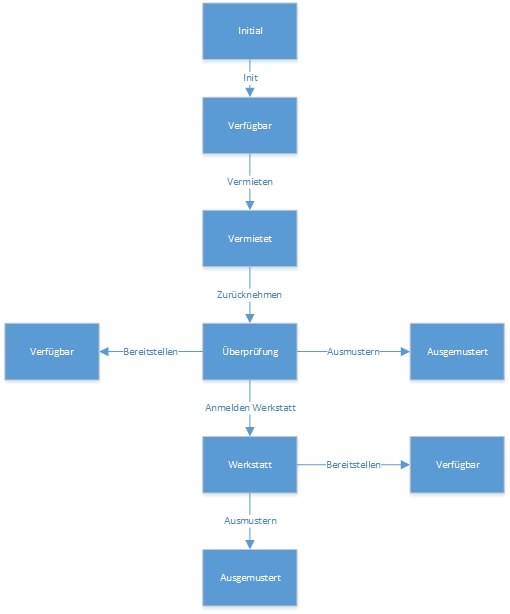
1. **Welche Information fehlt bei der Anforderung dadurch, dass der Autor die unter a) aufgeführten Fehler begangen hat?**

Keine genaue Spezifikation, was mit System gemeint ist und was eine Störung ist.

## Aufgabe 4 Zustandsbehafteter Software Test

1. **Betrachten Sie folgendes Zustandsdiagramm eines Mietwagens und erstellen Sie einen Übergangsbaum**





1. **Nach welchem Prinzip erstellen Sie mithilfe des Übergangsbaums Testfälle?**

Vollständige Zweigüberdeckung: „Jede Kante des Kontrollflussgraphen muss durch mindestens einen Testfall durchlaufen werden.“