Musterlösung Softwaretechnik

16.03.2006

Prof. Dr. Walter F. Tichy
M. Sc. A. Jannesari
Dipl.-Inform. G. Malpohl

Ab … Punkte…	…gab es:
0	5,0
22	4,0
25	3,7
28	3,3
31	3,0
34	2,7
37	2,3
40	2,0
43	1,7
46	1,3
49	1,0

Aufgabe 1: Aufwärmen (4+2+2+4+1+3= 16P)

a.) Kreuzen Sie an, ob die Aussage wahr oder falsch ist. (4 P)
Hinweis: Jedes korrekte Kreuz zählt 0,5 Punkte, jedes falsche Kreuz bewirkt 0,5 Punkte Abzug! Die Teilaufgabe wird mindestens mit 0 Punkten bewertet.

wahr	falsch	Aussage
	X	Das einzige Ziel der Softwaretechnik ist es, die Kosten der Erstellung von Software möglichst weitgehend zu senken.
	X	Das Pflichtenheft dient der Kommunikation mit dem Kunden und der Projektplanung.
	X	Qualitätsanforderungen wie die Benutzbarkeit erhöhen die Systemqualität und gehören zu den funktionale Anforderungen.
X		Komposition ist eine strengere Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben.
	X	„Was bauen wir?“ gehört zur Definitionsphase und „Wie strukturieren wir es?“ zur Implementierungsphase.
X		Ein sorgfältig aufgestellter Modulführer vermeidet Duplikation und erleichtert das Auffinden von betroffenen Modulen während der Wartung.
	X	Der Integrationstest im V-Modell ist der abschließende Test des Auftragnehmers in einer realistischen Umgebung ohne den Kunden.
X		Innerhalb einer Abnahme-Testserie ist es auch sinnvoll, Belastungs- oder Stresstests durchzuführen.

b.) Die Anforderungsvalidierung ist ein kritischer Schritt im Entwicklungsprozess. Nennen Sie vier der Anforderungsvalidierungskriterien. (2 P)

- Korrektheit (Correctness)
- Vollständigkeit (Completeness)
- Konsistenz (Consistency)
- Realismus (Realism)
- Verfolgbarkeit (Traceability)

0,5P pro Kriterium, max. 4 st.

c.) Welche Eigenschaften gelten für Schichten einer Schichtenarchitektur? (2 P)

- Die Dienstleistungen einer Schicht befinden sich auf demselben Abstraktionsniveau. (1P)
- Die Schichten sind entsprechend ihrem Abstraktionsniveau geordnet, so dass eine Schicht nur die Dienstleistungen der tieferen Schichten benötigt. (Oder: Austauschbarkeit der Schichten) (1P)

d.) Welche Informationen werden bei einem White-Box-Test von einem Testobjekt benötigt? (1 P)

- White-Box-Test Schnittsstellen und innere Struktur (Programmcode) des Testobjektes werden benötigt. (1P)

oder

- Kontrollfluss und Datenfluss (1P)

e.) Welche kontrollflussorientierten White-Box-Testarten kennen Sie? (3 P)

- Anweisungsüberdeckung, Zweigüberdeckung, Pfadüberdeckung (Vollständig, Boundary-Interior) und Bedingungsüberdeckung (Einfach, Mehrfach, Minimal mehrfach). (3P)

0,5P pro Testart

f.) Für welche Systeme ist das Prozessmodell *Prototyp* geeignet? (1 P)

- Das Prototypmodell ist geeignet für Systeme, für die keine vollständige Spezifikation ohne explorative Entwicklung oder Experimentation erstellt werden kann.

g.) Nennen Sie mindestens sechs *eXtreme Programming* (XP) Praktiken. (3 P)

- Planungsspiel
- Kleine Freigaben
- Systemmetapher
- Einfaches Design
- Test-First (Testgetriebene Entwicklung)
- Refaktorisieren
- Kollektiver Code-Besitz
- Andauernde Integration (Täglich)
- Paar-Programmierung
- Kunde bei Entwicklern
- Programmierrichtlinien
- Retrospektiven
- 40h Woche
- Kleine Teams
- Kurze Iterationen

0,5P pro Praktik

Aufgabe 2: Entwurfsmuster (1+2+5+3= 11P)

a.) Welchem Zweck dient das *Iterator-Muster*? (1 P)

- Es ermöglich den sequentiellen Zugriff auf die Elemente eines zusammengesetzten Objekts (0,5P), ohne seine zugrundeliegende Repräsentation offenzulegen (0,5P).

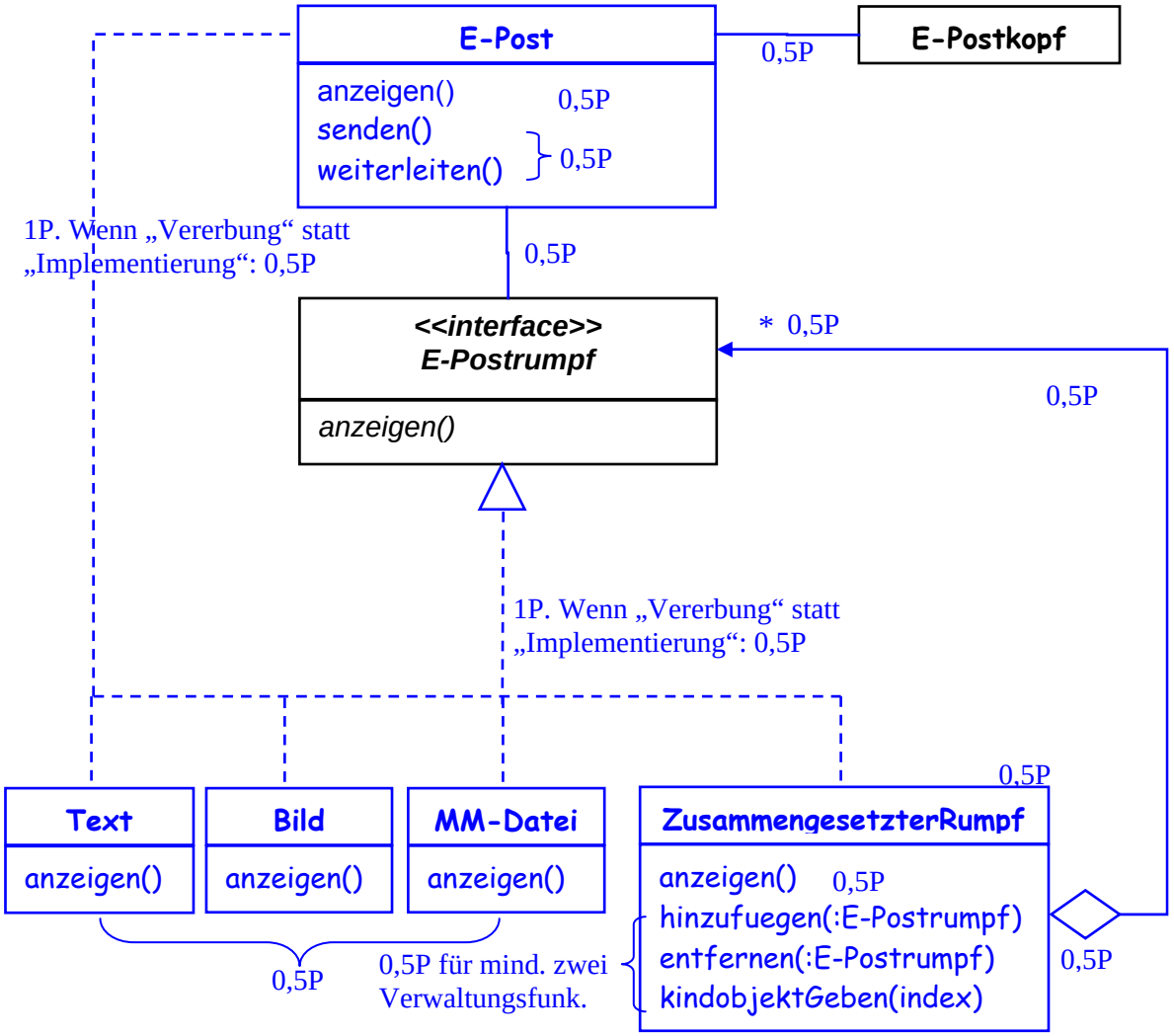
b.) Beschreiben Sie den Unterschied zwischen den Entwurfsmustern *Schablonenmethode* und *Fabrikmethode*. In welcher Beziehung stehen diese beiden Entwurfmuster? (2 P)

- Eine Fabrikmethode ist die Einschubmethode bei einer Schablonenmethode für Objekterzeugung. (2P)

Oder 0,5P pro Muster, wenn es beschrieben wurde:

- Schablonenmethode ermöglicht es Unterklassen, bestimmte Schritte eines Algorithmus zu überschreiben, ohne seine Struktur zu verändern (Verhaltensmuster).
- Fabrikmethoden ermöglichen es einer Klasse, die Erzeugung von Objekten an Unterklassen zu delegieren (Erzeugungsmuster).

c.) Vervollständigen Sie das untenstehende Klassendiagramm, das die Bestandteile eines vereinfachten E-Postrumpfs darstellen soll. (Die Klasse E-Postkopf spielt in diesem Aufgabenteil keine Rolle.)
Ein E-Postrumpf kann ein Text, ein Bild oder eine Multimedia-Datei sein, die alle die gemeinsame Schnittstelle E-Postrumpf implementieren. Außerdem kann ein E-Postrumpf noch aus mehreren der genannten Bestandteile bestehen. Das E-Mail-Programm soll den E-Postrumpf und seine Bestandteile einheitlich behandeln. Benutzen Sie bei Ihrem Entwurf ein Entwurfsmuster und benennen Sie dieses. Tragen Sie in Ihr Klassendiagramm die notwendigen Klassen, öffentlichen Methoden, Assoziationen und Vererbungsbeziehungen ein. (5 P)
Verwendetes Entwurfsmuster: _____ (Kompositum 0,5 P)



0,5P jeweils für die Funktionen „anzeigen()“ in den Klassen E-Post- und ZusammengesetzterRumpf, nur wenn „anzeigen()“ aus E-Postrumpf implementiert wurde.
0,5P jeweils für die Aggregation(◊), Assoziation und Kardialität(*), nur wenn Kompositum korrekt ist. (Komposition ist auch richtig)

d.) Fügen Sie dem obigen Diagramm die Klasse E-Post hinzu. Diese Klasse besitzt einen E-Postrumpf und einen E-Postkopf. Die E-Post kann versendet, weitergeleitet, und angezeigt werden. Weiterhin soll es möglich sein, dass eine E-Post andere E-Post-Objekte enthält. Benutzen Sie dazu wiederum das in Aufgabenteil c.) verwendete Entwurfsmuster. Erweitern Sie Ihr Klassendiagramm um die notwendigen Klassen, öffentlichen Methoden, Assoziationen und Vererbungsbeziehungen. (3 P)

Aufgabe 3: Analyse und UML (4+8= 12 P)

Das Unternehmen „CleanAnyWhere“ entwickelt eine neue Generation von Putzrobotern, welche das Reinigen an unzugänglichen Stellen ermöglichen sollen. Sie arbeiten dort als neuer Softwareentwickler und haben die Aufgabe, eine neue Version dieser Robotersteuerung zu entwerfen.

Der Roboter kann sich nur vorwärts bewegen. Richtungsänderungen können nur durch Drehen im Stand erfolgen.

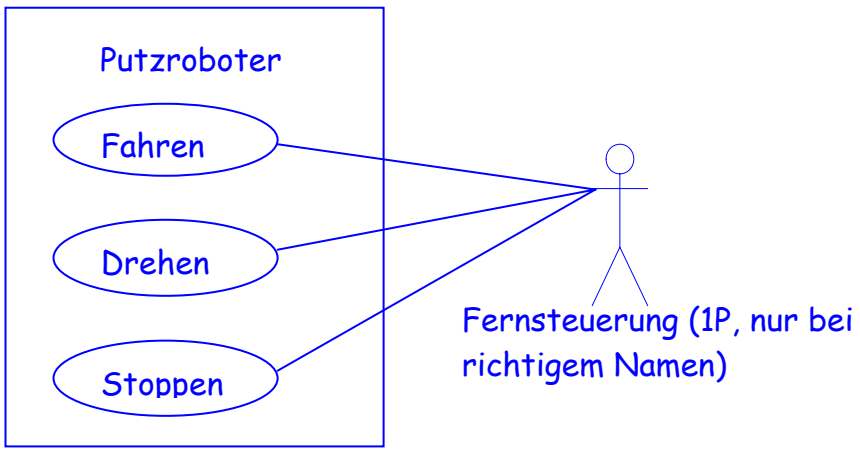
Der Roboter soll über verschiedene Befehle gesteuert werden. Mit dem Befehl „*Fahren*“ soll der Roboter vorwärts fahren und mit dem Befehl „*Drehen*“ soll sich der Roboter im Uhrzeigersinn drehen. Dazu werden die Robotermotoren gegenläufig eingeschaltet. Der Befehl „*Stoppen*“ hält den Roboter in seiner aktuellen Position. Bei einer Kollision mit einem Hindernis bzw. einer Wand führt der Roboter automatisch ein Rangiermanöver aus, das ihn vom Hindernis bzw. von der Wand löst. Zusätzlich kann das Rangiermanöver mit dem Stoppbefehl abgebrochen werden.
Der Sauger wird automatisch eingeschaltet, sobald der Roboter vorwärts fährt. Aber bei Drehen, Halten und Rangiermanövern wird er automatisch ausgeschaltet.

Die Befehle sollen mit Hilfe einer Fernsteuerung über die Infrarot-Schnittstelle an den Roboter gesendet werden. Die Fernsteuerung ist außerhalb des zu entwickelnden Systems.

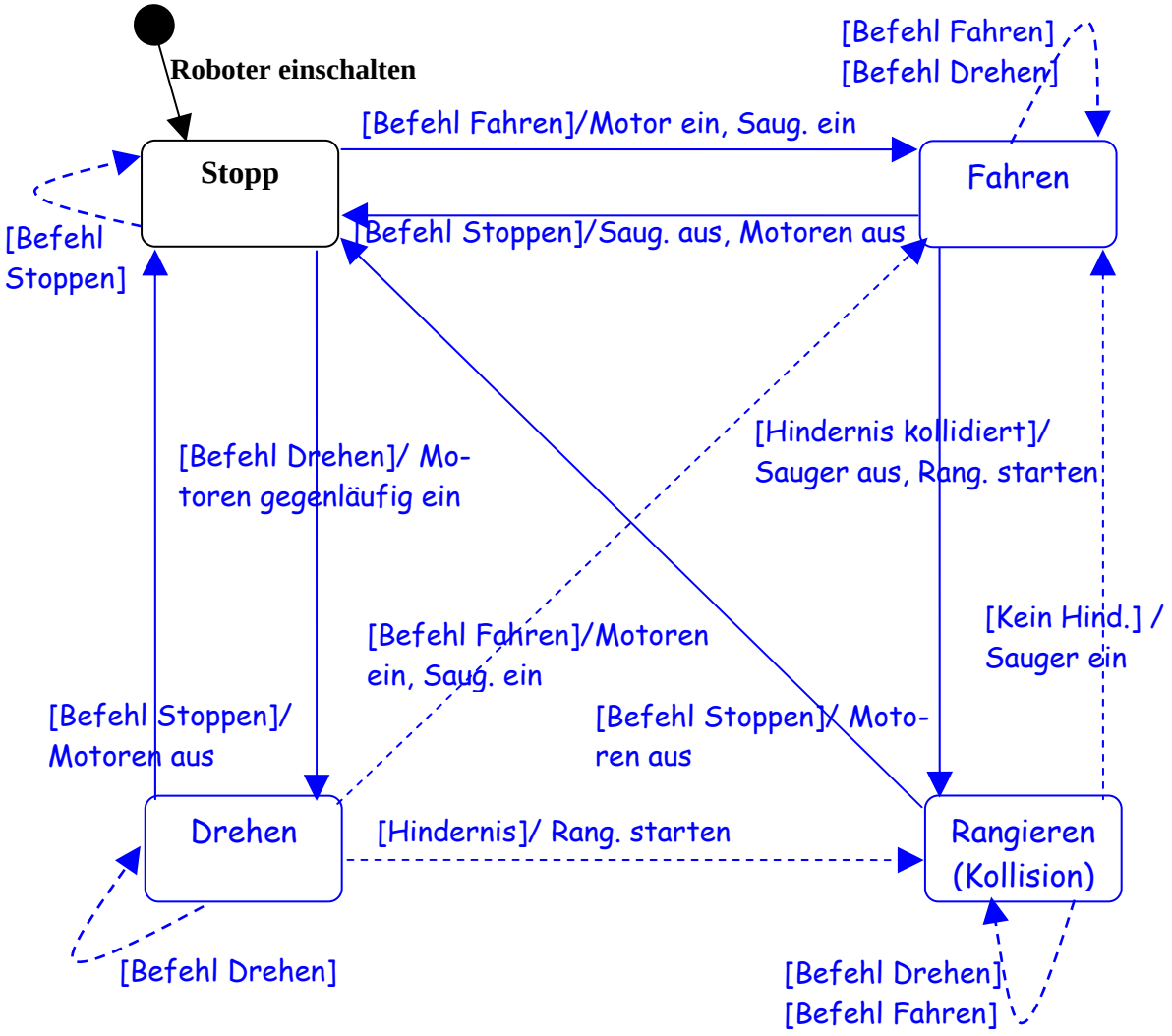
Obige Anforderungsbeschreibung soll mit Hilfe von Anwendungsfällen (Use-Cases) modelliert werden. Gehen Sie dabei wie folgt vor:

a.) Identifizieren Sie 3 Anwendungsfälle (Use-Cases) und die dazugehörigen Akteure. Berücksichtigen Sie dabei nur die elementaren Abläufe. Zeichnen Sie das entsprechende Anwendungsfall-Diagramm. (4 P)

- 1. Anwendungsfall: Fahren (1P mit dem Linie an Akteur)
- 2. Anwendungsfall: Drehen (1P mit dem Linie an Akteur);
- 3. Anwendungsfall: Stoppen (1P mit dem Linie an Akteur);



b.) Stellen Sie auf Basis der obigen Beschreibung ein Zustandsdiagramm für den Roboter auf. Benutzen Sie die in der Übung verwendete Notation. (8 P)



- Vollständigkeit der Zustände: 3P (1P pro Zustand)
- Vollständigkeit der Übergänge (Einschließlich der Bedingungen und Pfeile): 4P
- Korrektheit der Aktionen: 1P (Pro Fehler -0,5P)
- Gestrichelte Pfeile sind optional.

Aufgabe 4: Testen (4+5+4= 13P)

Gegeben sei folgender Ausschnitt eines Java-Programms. Es dient zur Suche nach einem Wert *x* in einem geordneten Feld *a* von ganzen Zahlen. Der hier verwendete Algorithmus heißt *binäre Suche*:
Man untersucht das Element in der Mitte des Feldes und abhängig vom Ergebnis setzt man die Suche entweder in der oberen oder der unteren Hälfte des Feldes fort.

```
public boolean binaereSuche(int x, int[] a)
{
  // a[] enthält sortierte ganze Zahlen (integer).
  // x ist der gesuchte Wert.
  // Wird x in a[] gefunden, dann wird „true“
zurückgegeben.

  int i = 0;
  int j = a.length;
  int mitte = 0;
  boolean gefunden = false;
  while ((i <= j) && !gefunden)
  {
    mitte = (i + j) / 2;
    gefunden = true;
    if (x < a[mitte])
    {
      j = mitte - 1;
      gefunden = false;
    }
    if (x > a[mitte])
    {
      i = mitte + 1;
      gefunden = false;
    }
  }
  return gefunden;
}
```

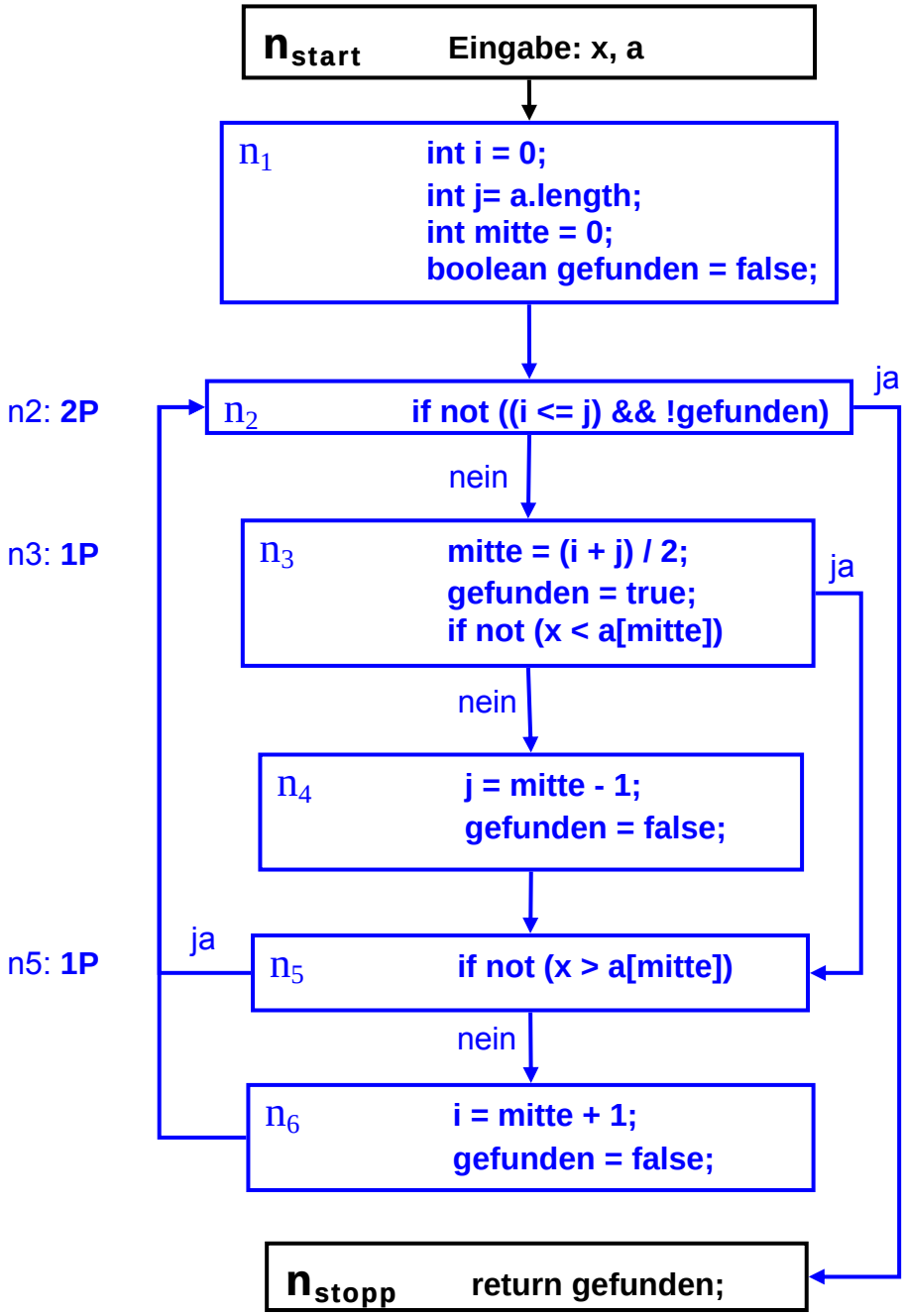
a.) Wandeln Sie obiges Programm mit einer strukturerhaltenden Transformation in eine der Definition der Vorlesung entsprechenden Zwischensprache um. (4 P)

Eingabeparameter: x, a

```
10:  int i = 0;
20:  int j= a.length;
30:  int mitte = 0;
40:  boolean gefunden = false;
50:  if not ((i <= j) && !gefunden)  goto 150; ( 1 P)
60:  mitte = (i + j) / 2;
70:  gefunden = true;
80:  if not (x < a[mitte]) goto 110; ( 1 P)
90:  j = mitte - 1;
100: gefunden = false;
110: if not (x > a[mitte]) goto 140; ( 1 P)
120: i = mitte + 1;
130: gefunden = false;
140: goto 50;
150: return gefunden;
160:
170:
180:
190:
200:
```

1P für "while": Bedingung 0,5P + Goto 0,5P
1P für erste "if" : Bedingung 0,5P + Goto 0,5P
1P für zweite "if" : Bedingung 0,5P + Goto 0,5P
1P für den Rest: Pro Fehler -1P

b.) Benutzen Sie die in Aufgabenteil a.) erstellte Repräsentation des Programms in der Zwischensprache, um einen Kontrollflussgraphen der Funktion *binaereSuche* zu erstellen. Wenden Sie dabei das aus der Vorlesung bekannte Verfahren an. (5 P)



1P für den Rest: Pro Fehler -1P

c.) Erstellen Sie Testfälle für die Anweisungsüberdeckung und die Zweigüberdeckung des obigen Programms. (4 P)

Anweisungsüberdeckung (2P) , Zweigüberdeckung (2P)
Erklärung: Wert x muss so gewählt werden, dass es größer als der erste und kleiner als der letzte Wert im Feld a sein.

Aufgabe 5a: ***Nur für Informatiker*** (5+3= 8P)

a.) XML

Gegeben sei die folgende XML-Dokumenttypdefinition (DTD). Schreiben Sie ein XML-Instanzdokument entsprechend der folgenden XML-DTD. (5 P)

```
<!DOCTYPE Buch [
  <!ELEMENT Buch              (Umschlag,Untertitel?,Teil+)>
  <!ELEMENT Umschlag          (Titel_Text)*>
  <!ELEMENT Titel_Text        (#PCDATA)>
  <!ELEMENT Untertitel        (#PCDATA)>
  <!ELEMENT Teil              (Abschnitt+)>
  <!ATTLIST Teil Ueberschrift CDATA #REQUIRED>
  <!ELEMENT Abschnitt         (Abschnitt_Text)>
  <!ELEMENT Abschnitt_Text    (p)+>
  <!ELEMENT p (#PCDATA)>
]>
```

```
<?xml version="1.0" encoding="utf-8"?>
<Buch>
 <Umschlag>
  <Titel_Text>Alles ….</Titel_Text> (optional)
 </Umschlag>

 <Untertitel>Mein…</Untertitel> (optional)

 <Teil Ueberschrift= "Willkommen …">

  <Abschnitt>

   <Abschnitt_Text>
    <p>Es … </p>
    <p>… </p> (optional)
   </Abschnitt_Text>

  </Abschnitt>

 </Teil>
</Buch>
```

Jeder Tag-Fehler (z.B. Fehlendes Tag, falsche Stelle, falsche Klammerung, …): -1P
Kleine Fehler (z.B. Klein geschrieben, mit Umlaut, Attribut vergessen, …): -0,5P

b.) Konfigurationsmanagement

Nennen Sie drei Probleme, die in einem Softwareentwicklungsprojekt mit langer Laufzeit auftreten können, wenn kein Konfigurationsmanagement durchgeführt wird. (3 P)

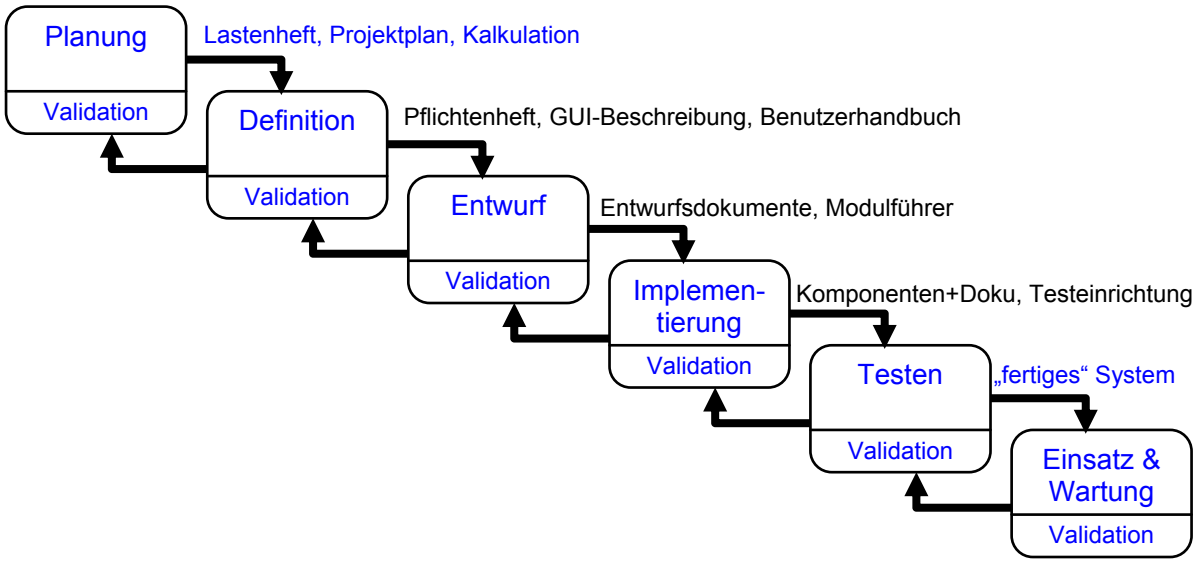
- Bereits behobene Fehler tauchen erneut auf.
- Quellcodemodule lassen sich nicht mehr kompilieren und linken, obwohl scheinbar keine Änderungen an den Quellcodemodulen vorgenommen wurden.
- Es lässt sich nicht nachvollziehen, ob und welche Fehler beseitigt wurden oder welche Funktionserweiterungen implementiert worden sind. (Traceability)
- Verlust von Änderungen.
- Es kann nicht nachvollzogen werden, wer eine Änderung gemacht hat. (Traceability)
- Es kann nicht festgestellt werden, welche Version beim Kunden installiert ist.

(Jedes Problem 1P)

Aufgabe 5b: *Nur für Informationswirte* (4+4= 8P)

a.) Prozessmodelle

Ein häufig angewendetes Prozessmodell für die Softwareentwicklung ist das Wasserfallmodell. Benennen Sie die einzelnen Phasen des Wasserfallmodells und die Dokumente, die von einer in die nächste Phase übergeben werden.



1P : 0,5P für die Phasen (ohne Validation) + 0,5P für die Produkte (Mindestens eins für die fehlenden Phasen).

Nennen Sie drei mögliche Gründe für Rückkopplungen zu Vorgängerphasen beim rückgekoppelten Wasserfallmodell. (4 P)

1P pro Grund aus unterschiedlichen Kategorien:

- unvollständig
- widersprüchlich
- mehrdeutig
- verbesserbar
- fehlerhaft
- unmöglich

z.B. :

- Unvollständige Spezifikationen, z.B. eine Anforderung ist im Systemmodell nicht berücksichtigt. → unvollständig
- Widersprüchliche Spezifikationen, z.B. widersprüchliche Spezifikation der Softwarekomponenten. → widersprüchlich
- Nicht eindeutige Spezifikationen, z.B. Anforderung, welche auf mehrere Arten interpretiert werden kann.
- Nicht realisierbarer Entwurf. → unmöglich

b.) Objektorientierte Konzepte

Erklären Sie kurz die Begriffe *Klasse*, *Vererbung* und *Polymorphie* in Zusammenhang mit objektorientierte Programmierung. Wie unterscheidet sich eine *Klasse* von einem *Objekt*? (4 P)

Jeweils 1P gab es für Erklärungen, die erkennen ließen, dass der Begriff verstanden wurde.

Beispiele:

- **Klasse**
- Klasse ist die Menge aller Objekte eines bestimmten Typs und beschreibt ihre Struktur. (Definition eines Objekttyps) Oder: Die Klasse repräsentiert das Konzept oder den abstakten Begriff, den wir uns von ihren Objekten machen.

- **Vererbung**
- Übertragung von Eigenschaften/Attributen auf andere Klassen. (Darf nicht über "Vererben" definiert werden.)

- **Polymorphie**

- Polymorphie (Vielgestaltigkeit) bedeutet unterschiedliche Wirkung von Botschaften in Abhängigkeit vom Empfänger.

Oder:

- Statische Polymorphie (Überladen): Es kann mehrere Methoden mit dem gleichen Namen geben (Signatur muss unterschiedlich sein, damit der Compiler weiß, welche er gerade zu verwenden hat).

Oder:

- Dynamische Polymorphie („Verwendung der Vererbung"): Es wird diejenige Methode mit der angegebenen Signatur aufgerufen, die in der Vererbungshierarchie von der Klasse der aktuellen Instanz aus gesehen am speziellsten ist.

- Ein **Objekt** ist eine konkrete Ausprägung (Instanz) einer Klasse.