LEHRSTUHL FÜR INFORMATIK 3
RWTH Aachen
Prof. Dr.-Ing. Manfred Nagl,
Dipl.-Ing. Marwan Hassani, Dipl.-Inform. Erhard Weinell

Klausur zu „Einführung in die Softwaretechnik“

Wintersemester 2007/08
Datum: 14.02.2008

Hinweise:

- Das Deckblatt muss ausgefüllt und unterschrieben werden.
- Die Klausur nicht trennen!
- Die Aufgaben sind in den dafür vorgesehenen Freiräumen oder auf den Rückseiten der Blätter zu lösen. Sollte der Platz nicht ausreichen, so können *vom Lehrstuhl bereitgestellte* Zusatzblätter verwendet werden. Diese sind dann jeweils mit Namen und Matrikelnummer zu kennzeichnen.
- Außer (dokumentenechten) Schreibgeräten (kein Bleistift o.ä.; außerdem kein Rot oder Grün) sind keine weiteren Hilfsmittel erlaubt.
- Die Klausur besteht aus insgesamt **4** Aufgaben auf **12** Seiten.
- Die Klausur ist bestanden, wenn **27** von 54 Punkten erreicht wurden.
- Die Bearbeitungszeit beträgt **120** Minuten.

Viel Erfolg!

Name: _____
(in Druckschrift)

Matr.Nr.: _____ **LÖSUNG!**

Unterschrift: _____

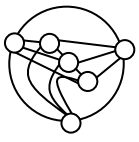
Aufgabe	Max. Punkte	Korrektur		Einsicht	
		Punkte	Kürzel	Punkte	Kürzel
1	10				
2	16				
3	14				
4	14				
∑	54				

Aufgabe 1 ***Wissensfragen*** 10 Punkte

(a) Nennen Sie vier Beispiele für *Planung* in *Entwicklungsprozessen*, und erläutern Sie diese kurz (max. 2 Sätze). Wählen Sie diese Beispiele aus mindestens zwei der diskutierten Kategorien/Ebenen.

(cf 3-58 .. 3-60) Techn. Ebene (feingranulare Ebene): Requirements Engineering: Planung im Sinne von Festlegung der Funktionalität des zukünftigen Systems
Architekturmodellierung: Planung im Sinne von Strukturierung des Bauplans des zukünftigen Systems
Adm. Ebene (grobgranulare Ebene): Schätzung: Planung im Sinne der Schätzung des Umfangs (z.B. in LOC), des Personalaufwands (in PM), der Kosten (in DM), der Zeiten (Anfangs-, Enddatum, Zeitdauer)
Vorabfestlegung von Konzepten, Sprachen, Modellen, Methoden, Richtlinien etc. (auf adm. Ebene wie auf techn. Ebene): grobe Vorgehensweise: Planung Lebenszyklusfestlegung, Gesamtproduktfestlegung Gröbstgranulare Ebene)

(b) Nennen Sie drei Beispiele für *Zusammenhänge* zwischen Haupt- und Nebenbereichen im *Arbeitsbereichsmodell*. Erläutern Sie jedes Beispiel kurz mit maximal 3 Sätzen.
(Die nebenstehende Skizze dient als Gedankenstütze)



(cf 2-17 .. 2-19) RE -> PO: Document(s) for project planning (black box estimation) depend on the requirements specification; the activities of requirements engineering are identified as activities to be planned, managed, supervised, and replanned on PO level.

PiL -> QA: The architecture is reviewed or checked giving rise to dependency relations between architecture documents and review protocols, check lists etc. Fur-thermore, the structure of module and integration test documents (test data collection, test drivers/stubs, test order determination plan etc.) is determined from the architecture especially when black-box test is used.
PiS -> Doc: The implementation ideas are described in the technical documentation.

(c) Nennen Sie drei *Dokumente*, die bei der *Anforderungsspezifikation* zur Entstehung des Pflichtenhefts beitragen, und skizzieren Sie kurz deren Inhalt.

(cf 4-7) Bestandsbericht (ex. Arbeitsabläufe, Produkte, Organisation)
Bedarfsskizze (abgeleitete Funktionalität) Lösungsalternativen (jeweilige Festlegungen)
Lastenheft (informelle Beschreibung von Funktionen, Daten, Qualitätsanforderungen) Projektplan (Organisatorische und kalkulatorische Werte) Durchführbarkeitsstudie

Aufgabe 2 ***Anforderungsspezifikation*** 16 Punkte

Die Billig-Fluggesellschaft *Nur-zu-Fuß-ist-günstiger* (*NuzFig*) *AG* entwickelt ein System zur Flugbuchung und Kundenverwaltung. Unterstützen Sie *NuzFig* bei der Anforderungsspezifikation:

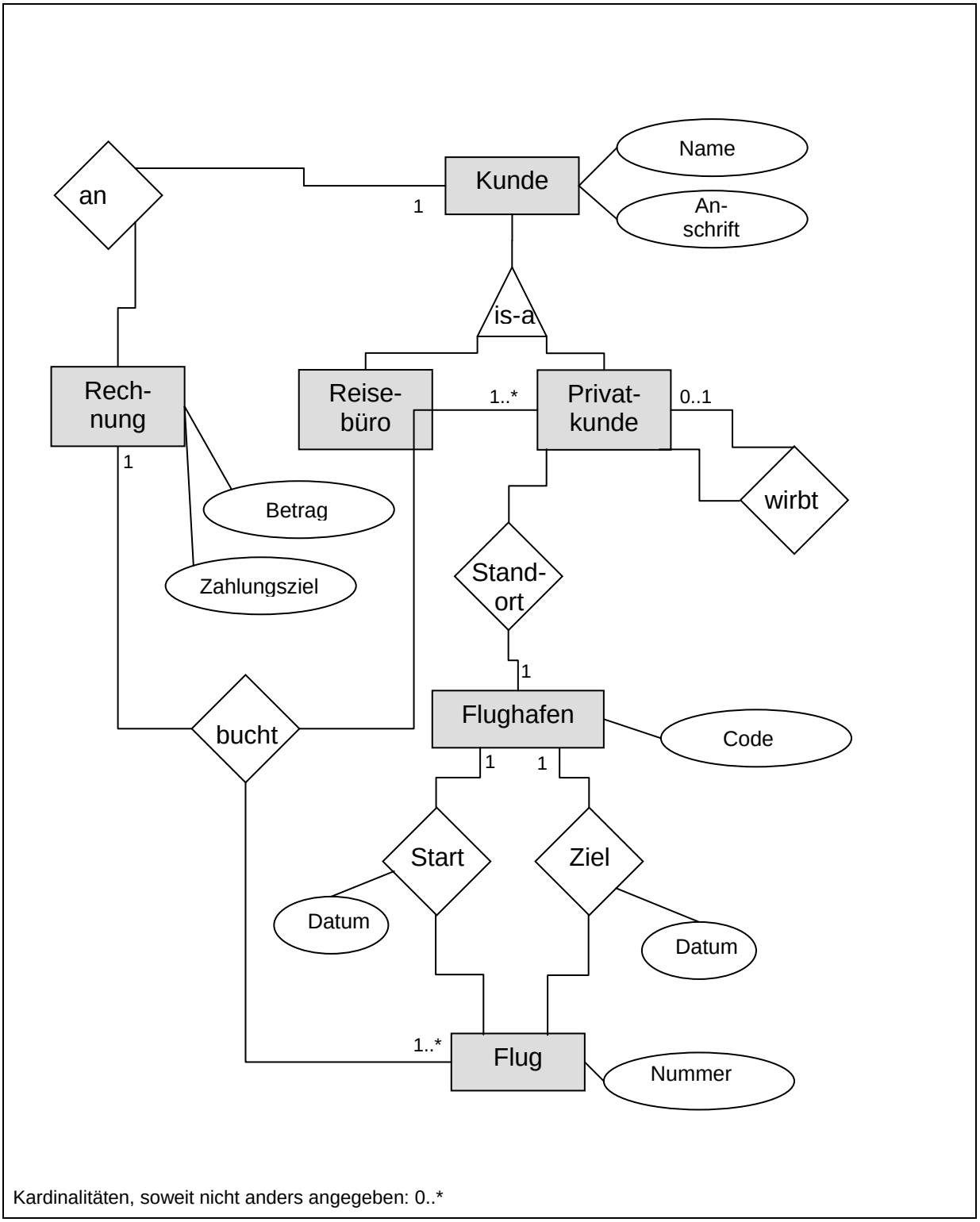
(a) Datenmodellierung mittels Extended-Entity-Relationship-Diagramm

Erstellen Sie ein Extended-Entity-Relationship-Diagramm zur Beschreibung des Datenmodells der *NuzFig AG*. Sie entnehmen die folgenden Angaben aus dem bereits erstellten Lastenheft:

- Bezüglich Kunden werden der Name und die Anschrift vermerkt. Es existieren zwei Arten von Kunden: Privatkunden und Reisebüros (die Flüge an Privatkunden weitervermitteln). Privatkunden können andere Privatkunden werben, was zur Zusendung von Prämien gespeichert werden muss.
- Das NuzFig-System verwaltet angesteuerte Flughäfen, die durch ihren IATA-Code identifiziert werden. Jedem Privatkunden wird ein ‚Heimatstandort' zugeordnet, um die Flugbuchung im Internet zu vereinfachen.
- Ein Flug, gekennzeichnet durch eine Flugnummer, erfolgt zwischen je zwei Flughäfen (Start / Ziel, einschließlich An- und Abflugsdatum).
- Flüge werden personengebunden für einen Privatkunden gebucht. Buchungen sind Bestandteil einer Rechnung, wobei eine Rechnung auch mehrere Flugbuchungen beinhalten kann (z.B. für mehrere Reisende). Eine Rechnung benennt den eingeforderten Betrag und das erwartete Zahlungsziel (Frist). Der Rechnungsempfänger ist das vermittelnde Reisebüro, bzw. der Privatkunde bei Direktvermittlung.

7 Punkte

Lösung Aufgabe 2a:



Kardinalitäten, soweit nicht anders angegeben: 0..*

(b) Datenfluss mittels SA-Datenflussdiagramm

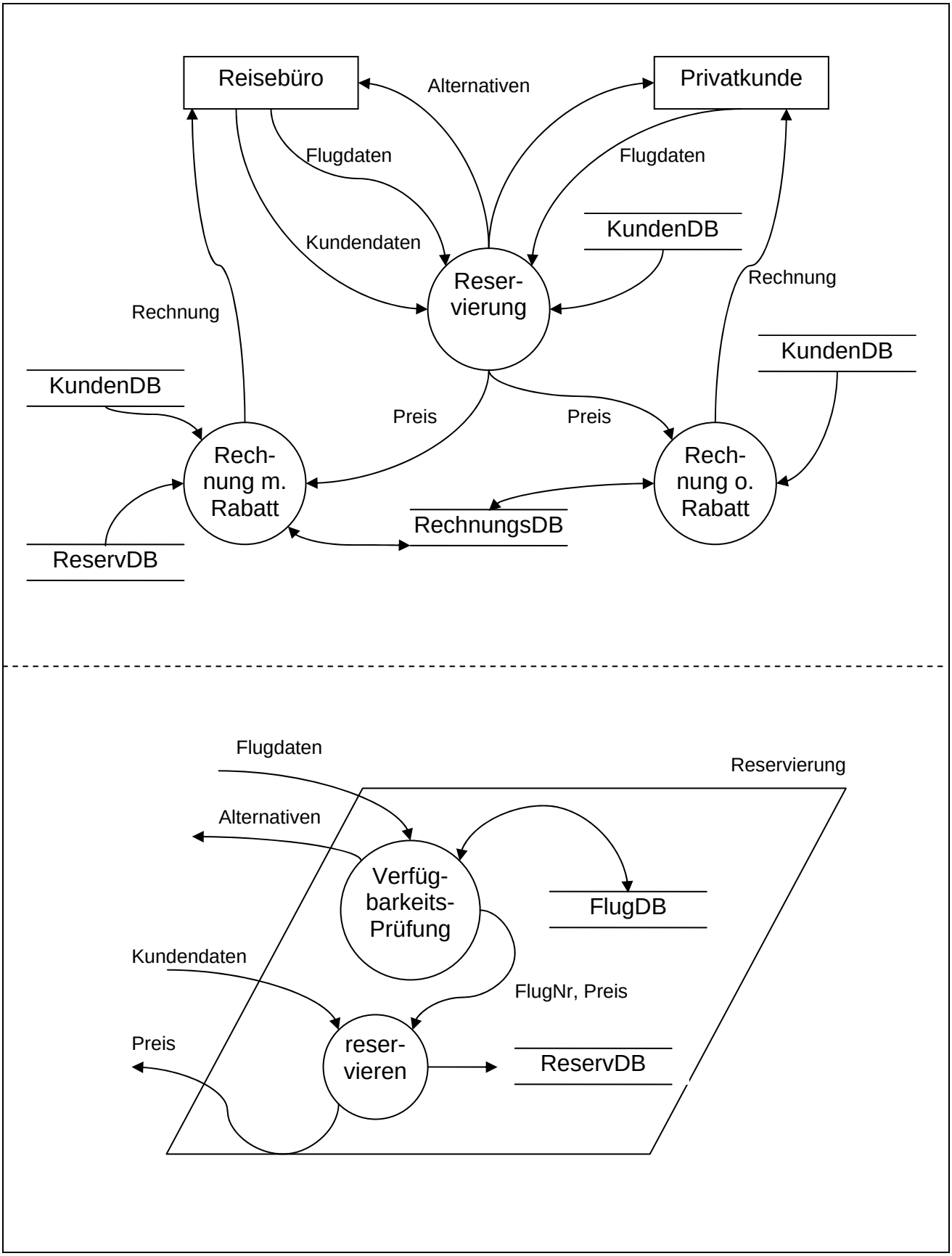
Erstellen Sie ein Datenflussdiagramm zur Beschreibung des Funktionsmodells der *NuzFig AG*. Sie entnehmen die folgenden Angaben aus dem bereits erstellten Lastenheft:

- Flugbuchungen können direkt durch einen Privatkunden (per Webformular) oder durch ein Reisebüro erfolgen. Angegeben werden An- und Abflugdatum sowie die Flughäfen (allg.: „*Flugdaten*“). Der Reisende kann bei Privatkunden anhand der Logindaten aus der *Kundendatenbank* entnommen werden, Reisebüros geben den Reisenden explizit an.
- Anschließend wird die *Flugdatenbank* auf Verfügbarkeit eines Fluges an den gewünschten Tagen durchsucht. Dies liefert im Erfolgsfall eine Flugnummer und einen Preis, woraufhin die Daten des Reisenden zusammen mit der Flugnummer in der *Reservierungsdatenbank* abgelegt werden. Andernfalls können alternative Flüge zu anderen Uhrzeiten oder benachbarten Flughäfen vorgeschlagen werden.
- Bei erfolgreicher Reservierung wird eine Rechnung gemäß des ermittelten Preises an den Kunden verfasst. Diese wird in der *Rechnungsdatenbank* abgelegt. Abhängig von ihren bisherigen Vermittlungen (zu entnehmen aus der *Reservierungsdatenbank*) wird Reisebüros dabei ein bestimmter Rabattsatz auf die Endsumme gewährt. Privatkunden erhalten keinen Rabatt. Die erstellte Rechnung wird dem Kunden abschließend zugesandt. Dessen Adresse kann der *Kundendatenbank* entnommen werden.

Randbedingung: Zur Erreichen der vollen Punktzahl müssen Sie für die Flugreservierung ein *hierarchisches* DFD sinnvoll einsetzen.
Hinweis: Es bietet sich an, einzelne Datenspeicher mehrfach in das Diagramm einzutragen, um unübersichtliche Flusskanten zu vermeiden.

9 Punkte

Lösung Aufgabe 2b:



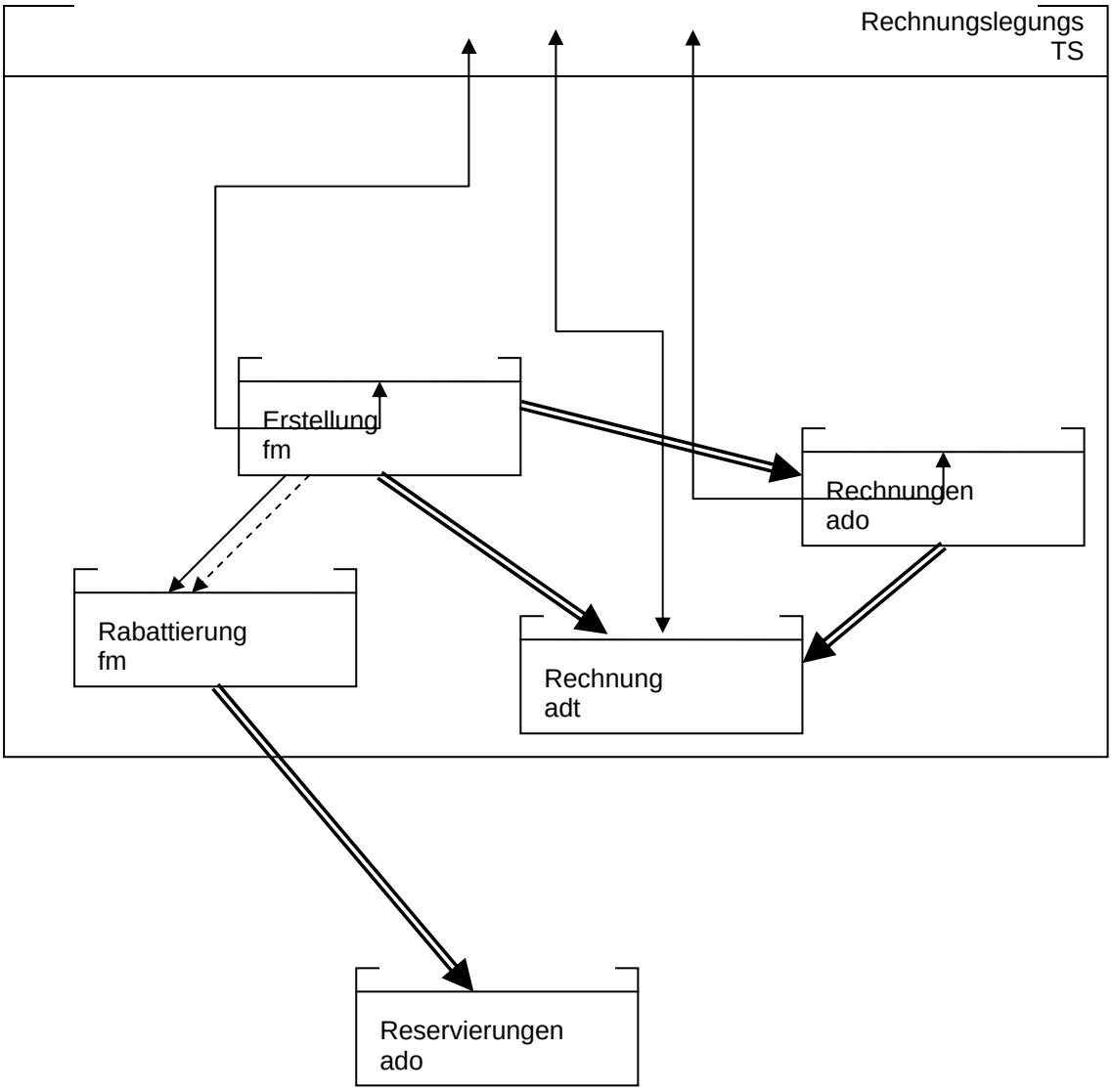
Aufgabe 3 ***Entwurfsspezifikation*** 14 Punkte

In dieser Aufgabe wird das System der NuzFig AG weiter verfeinert.

(a) Erstellen Sie ein *Architekturdiagramm* für das Teilsystem *Rechnungslegung,* welches folgende Anforderungen erfüllt. Dabei sind die einzelnen Module mit Namen und Typ, sowie Abhängigkeiten untereinander anzugeben.

- Rechnungen und Reservierungen werden als Datensätze abgespeichert, wobei Rechnungen innerhalb, Reservierungen außerhalb des Teilsystems einzuordnen sind.
- Rechnungen sollen typisiert als Eintrags-Kollektionssystem realisiert werden, das auch außerhalb des Teilsystems zugreifbar ist. Für Reservierungen genügt ein einfacher Datenspeicher.
- Das Erstellen der Rechnung ist vom Ermitteln des möglichen Rabattsatzes zu trennen, um eine einfachere Anpassung der jeweiligen Routinen zu ermöglichen. Die Rabattermittlung muss nicht von außerhalb des Teilsystems zugreifbar sein, sondern dient nur zur Unterstützung der Rechnungserstellung.

8 Punkte



(b) Spezifizieren Sie das Modul zur Ablage von *Rechnungen* formal. Vervollständigen Sie dazu die folgende Spezifikation nach *Parnas* and den unterschlängelten Stellen.

6 Punkte

```
module Rechnungen:
     function Erstellen(n, betrag)
     (* Erstellt eine neue Rechnung *)
        parameters: Rechnungsnummer n, int betrag

        effect:  if IstVorhanden(n) = true then
                     ERROR1
                 else
                     IstVorhanden(n) = true,
                     ErmittleBetrag(n) = betrag




     function ErmittleBetrag(n)
     (* Ermittelt den Betrag einer existierenden Rechnung *)
        parameters: Rechnungsnummer n
        possible values: int

        initial value: undefined

        effect: if IstVorhanden(n) = false then ERROR2



     function IstVorhanden(n)
     (* Prüft, ob eine Rechnung existiert *)
        parameters: Rechnungsnummer n
        possible values: boolean


        initial value: false

        effect: none
```

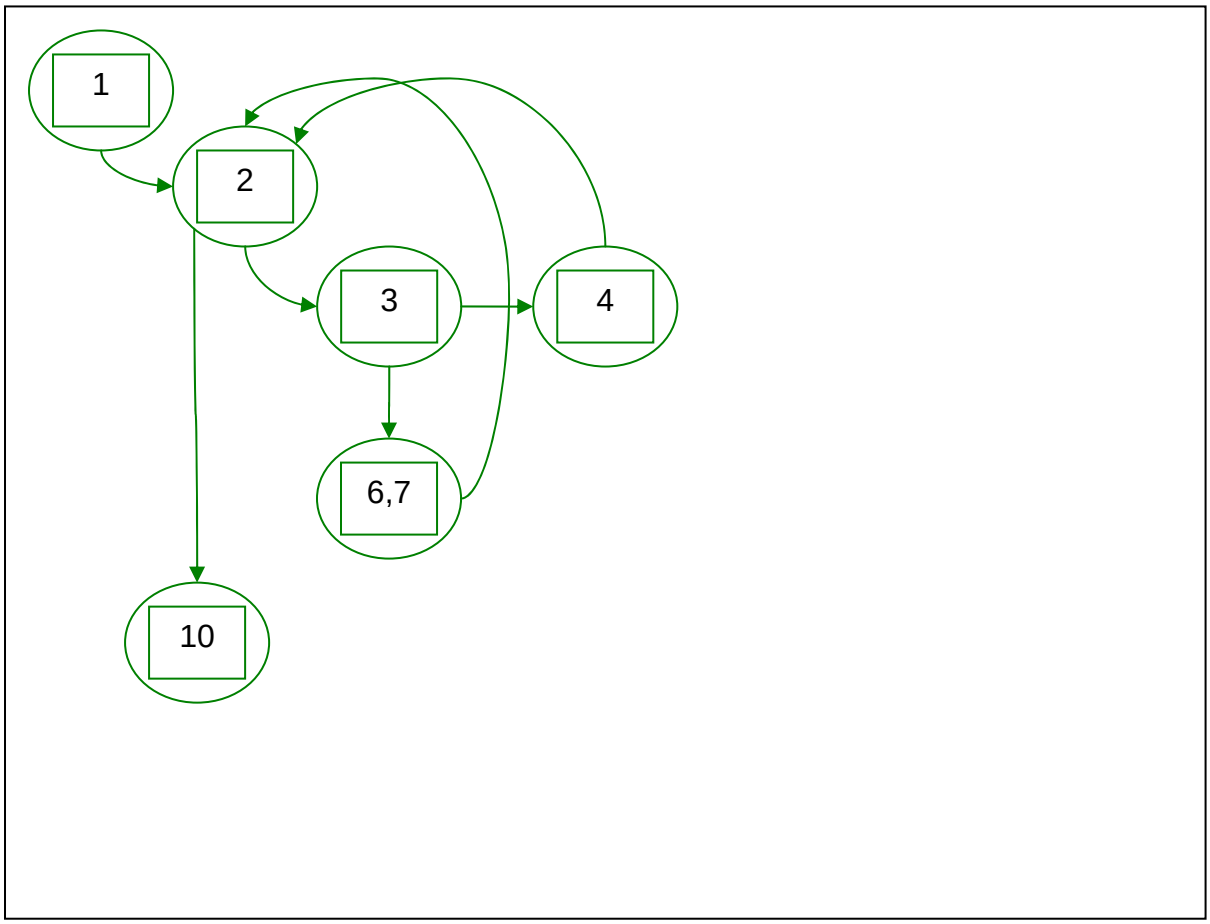
Aufgabe 4 — *Qualitätssicherung* — 14 Punkte

(a) **White-Box Testen:** Betrachten Sie den nachfolgenden Algorithmus *Gnomesort*, der ein Feld ganzer Zahlen aufsteigend sortiert. Die Arbeitsweise ähnelt dem bekannten Bubblesort, jedoch kann die Laufrichtung umgekehrt werden, wodurch in einigen Fällen weniger Vertauschungen erforderlich sind. Das Feld a sei einsbasiert (Indizes 1..len(a)).

```
procedure gnomeSort(int[] a) is
 1  int i := 1
 2  while (i >= 1 and i <= len(a)) do
 3    if (i = 1 or a[i-1] <= a[i]) then
 4      i := i + 1
 5    else
 6      swap (a[i], a[i-1])
 7      i := i – 1
 8    end
 9  end
10  return
end
```

Erstellen Sie zu diesem Programm den entsprechenden *Kontrollflussgraphen*. Geben Sie als Knotenbeschriftung die abgedeckten Zeilennummern an.

6 Punkte



(b) Geben Sie für obiges Beispiel *Testfälle* und *zugehörige Pfade* im Kontrollflussgraphen an. Decken Sie damit das Programm gemäß des sogenannten *Boundary-Interior Überdeckungskriteriums* ab, in dem *alle* Programmpfade bei

1. keinmaligem,
2. einmaligem, und
3. zweimaligem

Durchlaufen der Schleife überprüft werden. Da Verzweigung und Schleifendurchlauf von denselben Variablen abhängen, kann nicht für jeden gewünschten Pfad auch ein Testfall angegeben werden. Nennen Sie diejenigen nach Boundary-Interior zu prüfenden Pfade, für die *keine* geeigneten Testdaten bei festgelegter Schleifenzahl ermittelt werden können. Geben Sie schließlich auch einen Testfall (mit > 2 Schleifendurchläufen) an, bei dem *zwei* Tauschoperationen durchgeführt werden.

Hinweis: Als Testdaten benötigten Sie Felder höchstens der Länge drei. Die Anzahl der vorgegebenen Zeilen entspricht nicht notwendigerweise der Pfadanzahl.

8 Punkte

Testfälle / Programmpfade:

Testdaten (`int[]` a)	Programmpfad	Anzahl Schleifendurchläufe
[ ]	1 – 2 – 12	0
[1]	1 – 2 – 3 – 4 – 2 – 12	1
[1,2]	1 – 2 – 3 – 4 – 2 – 3 – 4 – 2 – 12	2

Nicht testbare Programmpfade des *Boundary-Interior* Kriteriums:

Programmpfad	Anzahl Schleifendurchläufe
1 – 2 – 3 – 6,7 – 2 – 12	1
1 – 2 – 3 – 4 – 2 – 3 – 6,7 – 2 – 12	2
1 – 2 – 3 – 6,7 – 2 – 3 – 4 – 2 – 12	2
1 – 2 – 3 – 6,7 – 2 – 3 – 6,7 – 2 – 12	2

Pfad mit zwei Vertauschungen: Kürzen Sie bitte einen Schleifendurchlauf durch den THEN-Zweig mit *A*, einen Durchlauf des ELSE-Zweigs mit *B* ab.

Testdaten (`int[]` a)	Programmpfad:	Anzahl Schleifendurchläufe
[2,3,1] [3,1,2]	1 – A – A – B – B – A – A – A – 12 1 – A – B – A – A – B – A – A – 12	7