SE Besprechung

Übung 5 – Verträge, Aufwand- und Risikoschätzung

Mengia Zollinger

Aufgabe 2.1 – Verträge

- **Anmerkungen:**
  - Generell gut gelöst worden
  - Zu wenig Aufmerksamkeit den Details geschenkt
- **Wertung:**
  - 0.25 Punkte/pre-condition and 0.25 Punkte/post-condition

- **Beispiel-Lösung:**

```
public class Matrix {
// @pre: h >= 0 and w >= 0;
// @post: height == h and width == w and rows.get(i).get(j) == 0, for
any i: 0 <= i < h and for any j: 0 <= j < w
public Matrix(int h, int w) {...}
```

Aufgabe 2.1 – Verträge

```
// @pre: 0 <= row < height() and 0 <= col < width()
// @post: height == height@PRE and width == width@PRE and result ==
(brief definition)
public Object get(int row, int col) {...}


// @pre: 0 <= row < height()and 0 <= col < width()
// @post: rows.get(row@PRE).get(col@PRE) == value and height ==
height@PRE and width == width@PRE and get(row, col) == value
public void set(int row, int col, Object value) {...}

// @pre: 0 <= r < height()
// @post: col == null for all col in rows.get(r@PRE) and height ==
height@PRE + 1 and width == width@PRE
public void addRow(int r) {...}
```

Aufgabe 2.1 – Verträge

```
// @pre: 0 <= c < width()
// @post: row.get(c@PRE) == null for all row in rows and height ==
height@PRE and width == width@PRE + 1
public void addCol(int c) {...}
```

```
// @pre: 0 <= r < height()
// @post: result == (brief definition) and height == height@PRE - 1
and width == width@PRE
public Vector removeRow(int r) {...}
```

```
// @pre: 0 <= c < width()
// @post: result == (brief definition) and height == height@PRE and
width == width@PRE - 1
public Vector removeCol(int c) {...}
```

Aufgabe 2.1 – Verträge

```
// @pre: -
// @post: height == height@PRE and width == width@PRE and result ==
width@PRE
public int width() {...}
}
```

- Häufige Fehler:
  - Java Code statt pre/post-condition – keine Implementierungen!

    Beispiel: `@post if (r > height@pre) then (height = r …) else …`
  - “=” statt “==”

    Beispiel: `height = height@pre + 1`
  - Variabeln erfunden, ohne je definiert zu haben

Aufgabe 2.2 – Function Points

- Dateneingaben
- Datenausgaben = 0
- Anfragen
- Externe Schnittstellen
- Interne Datenbestände

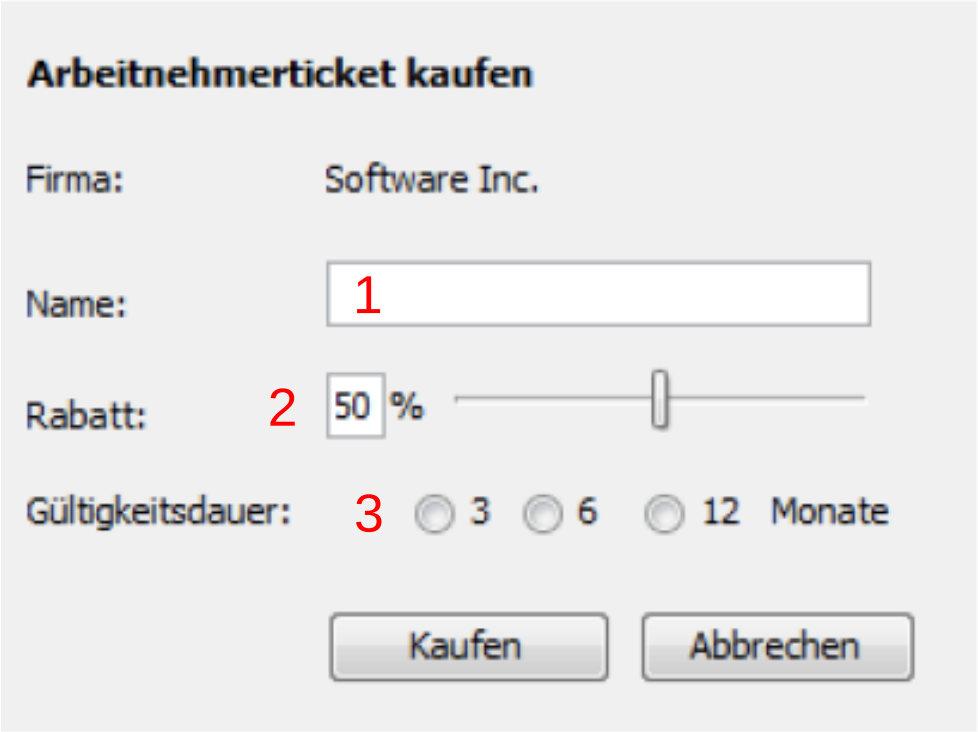
} (falls vorhanden → Annahme: *einfach*)

Element	Schwierigkeitsgrad			
	einfach	mittel	komplex	Summe
Dateneingaben	_____ x 3 = _____	_____ x 4 = _____	_____ x 6 = _____	_____
Datenausgaben	_____ x 4 = _____	_____ x 5 = _____	_____ x 7 = _____	_____
Anfragen	_____ x 3 = _____	_____ x 4 = _____	_____ x 6 = _____	_____
Ext. Schnittstellen	_____ x 5 = _____	_____ x 7 = _____	_____ x10 = _____	_____
Int. Datenbestände	_____ x 7 = _____	_____ x10 = _____	_____ x15 = _____	_____
			Function Point Rohwert (UFP)	

Aufgabe 2.2 – Function Points

- Dateneingaben
  - 3 Datenelemente

- Bei Klick auf *Abbrechen*: Eingaben werden nicht verwertet
- Klick auf *Kaufen*: keine zusätzliche Eingabe von Daten



Aufgabe 2.2 – Function Points

- Dateneingaben
  - 2 Datenbestände werden verarbeitet
    - „Arbeitgeber“ wird dabei verändert
    - „Ticket“ wird neu erstellt

Aufgabe 2.2 – Function Points

Anzahl bearbeiteter Datenbestände	Anzahl unterscheidbarer Datenelemente in der Eingabe		
	1–4	5–15	>15
0–1	einfach	einfach	mittel
2	einfach	mittel	komplex
>2	mittel	komplex	komplex

Element	Schwierigkeitsgrad			
	einfach	mittel	komplex	Summe
Dateneingaben	_____ x 3 = _____	_____ x 4 = _____	_____ x 6 = _____	_____
Datenausgaben	_____ x 4 = _____	_____ x 5 = _____	_____ x 7 = _____	_____
Anfragen	_____ x 3 = _____	_____ x 4 = _____	_____ x 6 = _____	_____
Ext. Schnittstellen	_____ x 5 = _____	_____ x 7 = _____	_____ x10 = _____	_____
Int. Datenbestände	_____ x 7 = _____	_____ x10 = _____	_____ x15 = _____	_____
			Function Point Rohwert (UFP)	

Aufgabe 2.2 – Function Points

- Dateneingaben: 1 (1 Fenster zur Dateneingabe)
- Datenausgaben: 0 (Annahme: Keine Rückgabe)
- Anfragen: 1 (Ticket kaufen)
- Externe Schnittstellen: 0 (alles innerhalb unseres Systems)
- Interne Datenbestände: 2 (Arbeitgeber & Ticket)

Element	Schwierigkeitsgrad einfach	mittel	komplex	Summe
Dateneingaben	1 x 3 =	x 4 =	x 6 =	3
Datenausgaben	x 4 =	x 5 =	x 7 =	
Anfragen	1 x 3 =	x 4 =	x 6 =	3
Ext. Schnittstellen	x 5 =	x 7 =	x10 =	
Int. Datenbestände	2 x 7 =	x10 =	x15 =	14
			Function Point Rohwert (UFP)	20

Aufgabe 2.2 – Function Points

- TDI: 4 Faktoren sehr hoch, 2 nicht vorhanden, Rest durchschnittlich

Nr.	Faktor	Wert	Einzusetzen sind Werte zwischen 0 und 5
1	Datenkommunikation	3	
2	Verteilte Funktionen	3	0 nicht vorhanden, kein Einfluss
3	Leistungsanforderungen	3	1 unbedeutender Einfluss
4	Belastung der Hardware	1	2 mäßiger Einfluss
5	Verlangte Transaktionsrate	3	3 durchschnittlicher Einfluss
6	Online-Dateneingabe	3	4 erheblicher Einfluss
7	Effiziente Benutzerschnittstelle	5	5 starker Einfluss
8	Online-Datenänderungen	3	
9	Komplexe Verarbeitungen	1	
10	Wiederverwendbarkeit	4	
11	Einfache Installation	3	
12	Einfache Benutzbarkeit	5	
13	Installation an mehreren Orten	3	
14	Änder- und Erweiterbarkeit	5	
	Summe der Faktoren (TDI)	45	

TDI : 3 * 5 + 1 * 4 + 2 * 1 + 8 * 3

= **45**

VAF : 0.65 + 0.01 * TDI =

0.65 + 0.01 * 45 = **1.1**

FP : UFP * VAF = 20 * 1.1 = **22**

Aufgabe 2.2 – Function Points

- Was braucht es noch?
  - Mittlerer Aufwand pro Function Point muss bekannt sein
  - Faktoren müssen unternehmensspezifisch kalibriert und projektspezifisch angepasst werden

Aufgabe 2.3 – COCOMO2, a)

- Wahl Projektleiter

Skalierungsfaktoren	**Programs4People (intern)**	**TravelSystemsIT (extern)**
Präzedenz*	tiefer	höher
Zusammenarbeit	höher	tiefer

Vertrautheit des Entwicklungsteams mit dem Produkt

Kostenfaktoren	**Programs4People (intern)**	**TravelSystemsIT (extern)**
Analyst Capability	tiefer	höher
Application Experience	tiefer	höher

Aufgabe 2.3 – COCOMO2, b)

- Outsourcing

Skalierungsfaktoren	**IT4U**	**ITIndia**
Zusammenarbeit	höher	tiefer

Kostenfaktoren	**IT4U**	**ITIndia**
Platform Experience	tiefer	höher
Language and Tool Exp.	tiefer	höher
Team Co-location and communications support	höher	tiefer

Aufgabe 2.3 – COCOMO2, c)

Eigenschaften, welche den Faktor **Risiko-Umgang** beeinflussen

- Benutzerschnittstelle nie klar definiert, aber wichtig
- Wiederzuverwendende Schnittstellen / Systemteile (→ Altlasten)
- Termindruck (→ Fehler)
- Budget-Überschreitung
- Grösse des Teams (→ Zeitproblem)
- Datenaustausch Uni (→ Gesetzliche Vorgaben?)
- Keine Outsourcing-Erfahrung (→ Zeitproblem)
- Indien: Umweltkatastrophen (Überschwemmungen) → Zeitproblem

Aufgabe 2.3 – COCOMO2, d)

KSLOC Berechnung

- Reuse Required: von nominal **1.14** auf very high **1.49**
- Formeln auf Folie 17 (Kapitel 16)

$$\text{Aufwand} = 2.45 * \text{KSLOC}^{B} * \prod_{i=1}^{17} EM_i$$

Wachstumsfaktor:

$$B = 1.01 + 0.01 \sum_{i=1}^{5} Sf_i$$

$$= 1.153$$

Skalierungsfaktoren:

Faktor	Sehr gering	Gering	Nominal	Hoch	Sehr hoch	Extra hoch
Präzedenz	4,05	3,24	2,43	1,62	0,81	0
Flexibilität	6,07	4,86	3,64	2,43	1,21	0
Risiko-Umgang	4,22	3,38	2,53	1,69	0,84	0
Zusammenarbeit	4,94	3,95	2,97	1,98	0,99	0
Prozessreife	4,54	3,64	2,73	1,82	0,91	0

Aufgabe 2.3 – COCOMO2, d)

KSLOC Berechnung

- Reuse Required: von **1.14** auf **1.49**
- Formeln auf Folie 17 (Kapitel 16)

$$\text{Aufwand} = 2.45 * \text{KSLOC}^{B} * \prod_{i=1}^{17} EM_i$$

Factor	Very low	Low	Nominal	High	Very High	Extra high
Reliability required	0.75	0.88	1.00	1.15	1.39	
Database size	0.93	1.00	1.09	1.19		
Product complexity	0.75	0.88	1.00	1.15	1.30	1.66
Reuse required	0.91	1.00	1.14	1.29	1.49	
Documentation required	0.89	0.95	1.00	1.06	1.13	
Execution time constraint			1.00	1.11	1.31	1.67
Storage constraint			1.00	1.06	1.21	1.57
Platform volatility		0.87	1.00	1.15	1.30	
Analyst capability	1.50	1.22	1.00	0.83	0.67	
Programmer capability	1.37	1.16	1.00	0.87	0.74	
Personnel continuity (turnover)	1.24	1.10	1.00	0.92	0.84	
Application experience	1.22	1.10	1.00	0.89	0.81	
Platform experience	1.25	1.12	1.00	0.88	0.81	
Language and tool experience	1.22	1.10	1.00	0.91	0.84	
Use of software tools	1.24	1.12	1.00	0.86	0.72	
Team co-location and communications support	1.25	1.10	1.00	0.92	0.84	0.78
Required development schedule	1.29	1.10	1.00	1.00	1.00	

$$(\text{KSLOC}_O) \quad \prod_{i=1}^{17} EM_i = 1.14 * EM_{Rest}$$

$$(\text{KSLOC}_N) \quad \prod_{i=1}^{17} EM_i = 1.49 * EM_{Rest}$$

Aufgabe 2.3 – COCOMO2, d)

$$\text{Aufwand} = \prod_{i=1}^{17} EM_i * 2.45 * KSLOC^B$$

$$\text{Aufwand}_{Old} >= \text{Aufwand}_{Neu}$$

$$1.14 * EM_{Rest} * 2.45 * KSLOC_O^B = 1.49 * EM_{Rest} * 2.45 * KSLOC_N^B \quad | \ /2.45, /EM_{Rest}$$

$$1.14 * KSLOC_O^B = 1.49 * KSLOC_N^B \quad | \ KSLOC_N^B = (x * KSLOC_O)^B$$

$$1.14 * KSLOC_O^B = 1.49 * (x * KSLOC_O)^B$$

$$1.14 * KSLOC_O^B = 1.49 * x^B * KSLOC_O^B \quad | \ /KSLOC_O^B$$

$$1.14 = 1.49 * x^B \quad | \ /1.49$$

$$1.14 / 1.49 = x^{1.153} \quad | \ \sqrt[1.153]{}$$

$$(1.14 / 1.39)^{1 / 1.153} = x$$

$$0.7929 = x$$

➔ KSLOC muss um mind. **<u>20.7%</u>** reduziert werden

Aufgabe 2.3 – COCOMO2, e)

Voraussetzung, damit COCOMO2 zuverlässige Werte liefert?

- Faktoren müssen unternehmensspezifisch kalibriert werden
  - Erfahrungen aus vergangenen Projekten (vom eigenen Unternehmen oder evtl. von anderen mit ähnlichen Projekten)
  - Berechnungen auf konkrete Bedingungen anpassen

(stabile Umgebung & einfache Anwendungssoftware gilt nur für COCOMO (1))

Aufgabe 2.4 – Risikoschätzung

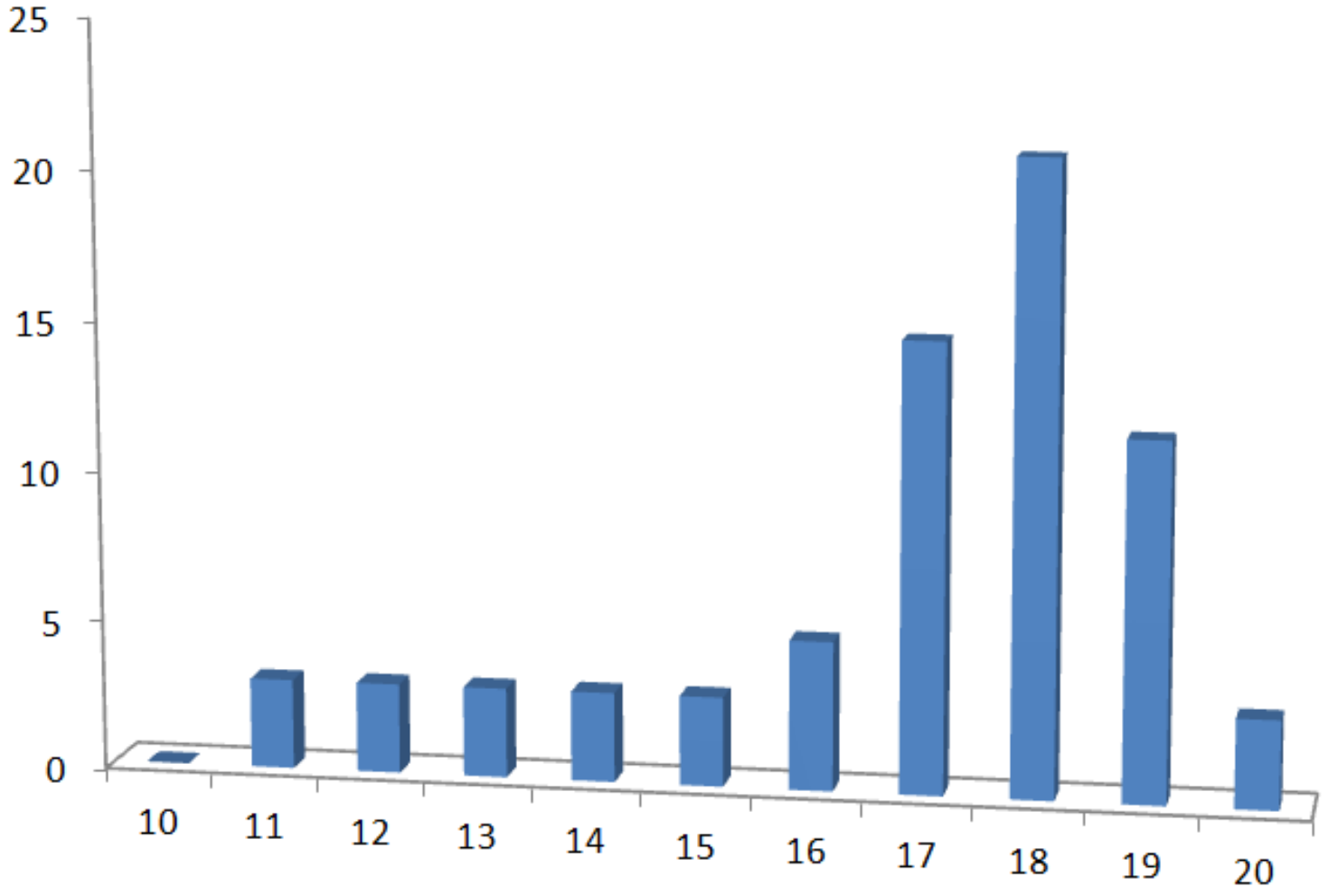
- Risikoanalyse
  - Möglicher Verlust (Zeit, Geld, Kontrolle, Qualität)
- Risikobewertung
  - Schadenshöhe * Eintrittswahrscheinlichkeit
- Massnahmen zur Vorbeugung
  - Risiko vermeiden, mindern, Planung für Worst Case

- Keine Risiken aus dem Skript

Aufgabe 2.4 – Risikoschätzung

- Unmotivierte Entwickler
  - E = 3, S = 4, Risiko = 12
  - Arbeitsbedingungen verbessern, Firmenfeier, Anreize schaffen
- Feuer / Virus → Datenverlust
  - E = 2, S = 9, Risiko = 18
  - Regelmässige Backups (auswärtiger Server), Rauchverbot, Firewalls
- Zahlungsunfähigkeit des Arbeitgebers
- Übersetzungsprobleme (mehrsprachiges GUI)
- Falscher Technologieeinsatz
- Interne Kommunikationsprobleme
- Patentverletzungen

Übung 5 Resultate



Fragen?