# *SE Besprechung*

## Übung 5 – Verträge, Aufwand- und Risikoschätzung



Nicolas Hoby

# *Aufgabe 2.1 – Verträge*

- Schnittstellendefinition mit Verträgen
  - *@post result = ...* wenn die Methode einen Rückgabewert hat
  - *@post [Zustand nach Aufruf]* … wenn Methode etwas verändert

- Automatische Überprüfung (mit Hilfe des Interfaces, bzw. der Methoden)
  - *class= class@pre* kann mit dem Interface nicht geprüft werden

# *Aufgabe 2.1 – Verträge*

- Wenn eine Methode eine Zustandsvariable nicht verändert, so muss dies explizit zugesichert werden (allgemein Gültiges kann in Klasseninvariante verschoben werden)

- Gefahr implementierungsabhängiger Spezifikationen → nur solche Zustandsvariablen verwenden, welche eine Entsprechung im Problembereich / in der Anwendungsdomäne haben

# *Aufgabe 2.1 – Verträge*

- Preconditions (Voraussetzungen)
- Postconditions (Ergebniszusicherungen)
- Invariants (Invarianten)
- Obligations (Verpflichtungen)

„Wenn Du **das** tust (@pre), dann garantiere ich Dir **diese** Ergebnisse (@post)“.

**Etwas** gilt über die ganze Zeit → Klasseninvariante (@inv).

„Wenn Du diese Methode aufrufst, musst Du auch **folgendes** tun (@obligation)“.

# *Aufgabe 2.1 – Verträge*

- Voraussetzen (@pre), wenn es dem Aufrufer zugemutet werden kann
- Prüfen, wenn mit Falscheingaben gerechnet werden muss, (und nur wenn eine sinnvolle Behandlung von Fehlern möglich ist; sonst Exception werfen)

# Aufgabe 2.1 – Verträge

```
* …
* @pre    coefficientA != 0
*
* @post  discriminant = coefficientB^2 – 4 * coefficientA * coefficientC
*          && coefficientA = coefficientA@pre
*          && coefficientB = coefficientB@pre
*          && coefficientC = coefficientC@pre
*
* @return the discriminant of the equation
*/
public double discriminant() {...}
```

# *Aufgabe 2.1 – Verträge*

```
* …
* @post  result = coefficientA * x^2 + coefficientB * x
            + coefficientC
*           && coefficientA = coefficientA@pre
*           && coefficientB = coefficientB@pre
*           && coefficientC = coefficientC@pre
*
* @return the value of the polynomial in x
*/
public double evaluateP( double x) {...}
```

# *Aufgabe 2.1 – Verträge*

```
 * …
 * @post  if ( discriminant < 0 ) then ( result = 0 )
 *          && if ( discriminant == 0 ) then ( result = 1 )
 *          && if ( discriminant > 0 ) then ( result = 2 )
 *
 * @return the number of roots of the polynomial
 */
public int rootsNumber() {...}
```

# Aufgabe 2.1 – Verträge

```
 * …
 * @pre  discriminant > 0
 * @post  evaluateP( result ) = 0
 *          && coefficientA = coefficientA@pre
 *          && coefficientB = coefficientB@pre
 *          && coefficientC = coefficientC@pre
 *
 * @return the first root of the polynomial
 */
public double root1() {...}
```

# Aufgabe 2.1 – Verträge

```
 * …
 * @pre  discriminant = 0
 * @post  evaluateP( result ) = 0
 *          && coefficientA = coefficientA@pre
 *          && coefficientB = coefficientB@pre
 *          && coefficientC = coefficientC@pre
 *
 * @return the unique root of the polynomial
 */
public double uniqueRoot() {...}
```

# Aufgabe 2.2 – Function Points

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

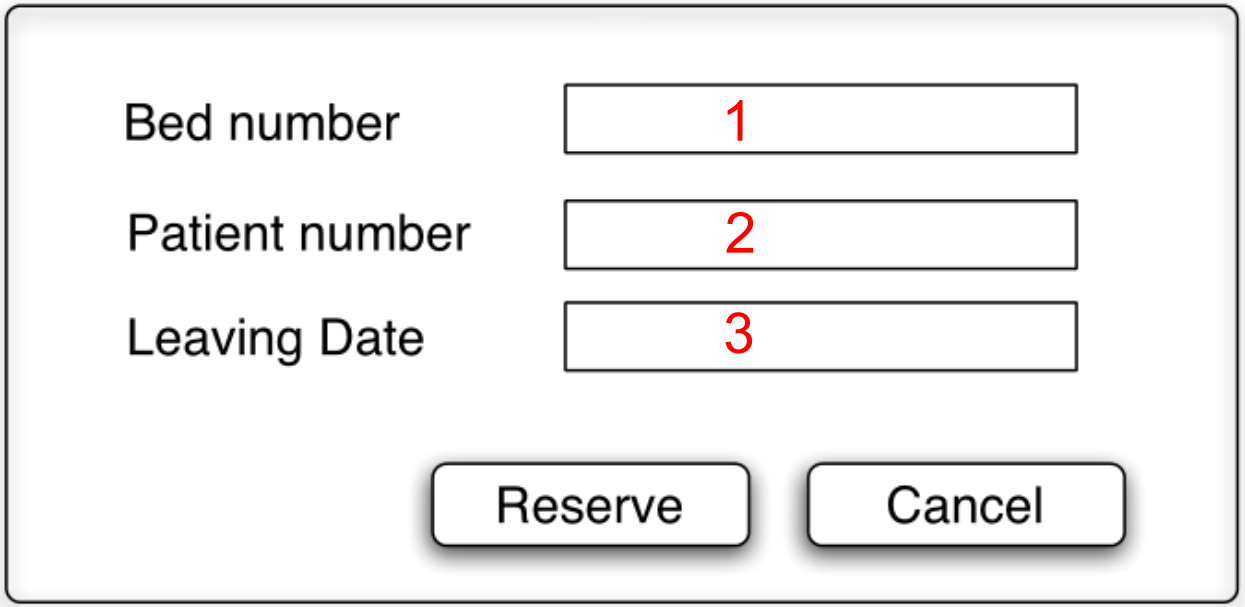
} (falls vorhanden → Annahme: *einfach*)

| 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*

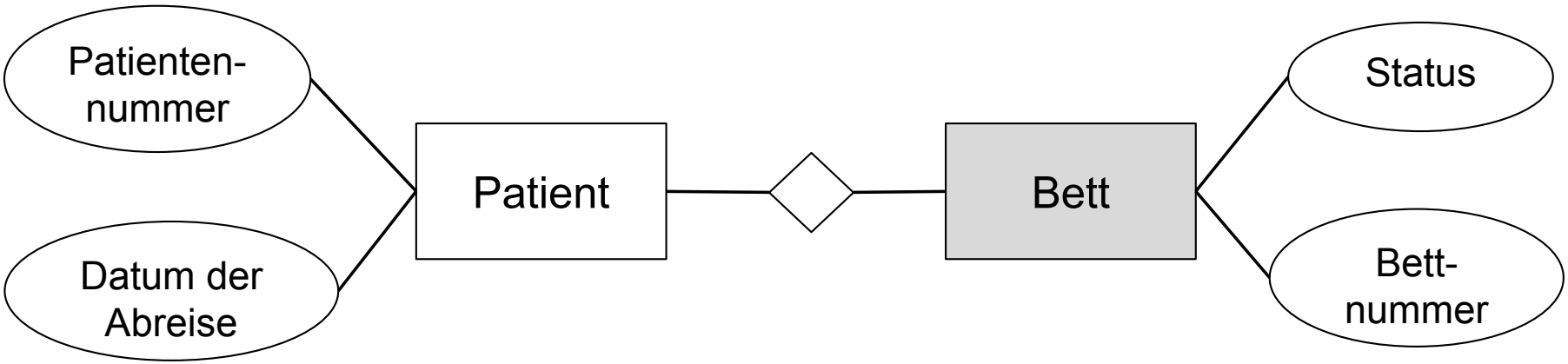
- Dateneingaben
  - 3 Datenelemente

- Bei Klick auf *Cancel*: Eingaben werden nicht verwertet
- Klick auf *Reserve*: keine zusätzliche Eingabe von Daten



# Aufgabe 2.2 – Function Points

- Dateneingaben
  - 2 Datenbestände werden verarbeitet
  - Nur Datenbestand „Bett“ wird dabei verändert



# 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 | | | Summe |
|---|---|---|---|---|
| | einfach | mittel | komplex | |
| 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

- Dateneingaben: 1 (1 Fenster zur Dateneingabe)
- Datenausgaben: 0 (Keine Rückgabe)
- Anfragen: 1 (Status des Bettes prüfen)
- Externe Schnittstellen: 0 (alles innerhalb unseres Systems)
- Interne Datenbestände: 2 (Patient & Bett)

| Element | Schwierigkeitsgrad | | | Summe |
|---|---|---|---|---|
| | einfach | mittel | komplex | |
| 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 |
|---|---|---|
| 1 | Datenkommunikation | 3 |
| 2 | Verteilte Funktionen | 3 |
| 3 | Leistungsanforderungen | 3 |
| 4 | Belastung der Hardware | 3 |
| 5 | Verlangte Transaktionsrate | 3 |
| 6 | Online-Dateneingabe | 3 |
| 7 | Effiziente Benutzerschnittstelle | 5 |
| 8 | Online-Datenänderungen | 3 |
| 9 | Komplexe Verarbeitungen | 0 |
| 10 | Wiederverwendbarkeit | 5 |
| 11 | Einfache Installation | 3 |
| 12 | Einfache Benutzbarkeit | 5 |
| 13 | Installation an mehreren Orten | 0 |
| 14 | Änder- und Erweiterbarkeit | 5 |
| | Summe der Faktoren (TDI) | 44 |

Einzusetzen sind Werte zwischen 0 und 5

0 nicht vorhanden, kein Einfluss
1 unbedeutender Einfluss
2 mäßiger Einfluss
3 durchschnittlicher Einfluss
4 erheblicher Einfluss
5 starker Einfluss

$4 * 5 + 8 * 3 = 44$ (TDI)

$0.65 + 0.01 * 44 = 1.09$ (VAF)

UFP * VAF = FP

$20 * 1.09 =$ **21.8**

# *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*** | **IT4U** | **OnlineSystemsIT** |
|---|---|---|
| Präzedenz | tiefer | höher |
| Zusammenarbeit | höher | tiefer |

| ***Kostenfaktoren*** | **IT4U** | **OnlineSystemsIT** |
|---|---|---|
| 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 *Flexibilität* beeinflussen

- Schon vorhandene, wiederzuverwendende Schnittstellen / Systemteile (→ weniger flexibel)
- Termindruck / Zeitvorgaben durch den Kunden (→ weniger flexibel)
- Einflussnahme des Kunden (je mehr der Kunde mitentscheidet, desto weniger flexibel sind wir. Auf der anderen Seite müssen wir flexibel sein, um auf laufende Kundenwünsche einzugehen.)
- Grösse des Teams (je grösser, desto weniger flexibel)

# Aufgabe 2.3 – COCOMO2, d)

KSLOC Berechnung

- Reliability required: von **1.00** auf **1.39**
- Formeln auf Folie 17 (Kapitel 16)

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

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

$$= 1.153$$

$(\text{KSLOC}_O)$ $\prod_{i=1}^{17} EM_i = 1.00 \text{ ^ } 17 = 1.00$

$(\text{KSLOC}_N)$ $\prod_{i=1}^{17} EM_i = 1.00 \text{ ^ } 16 * 1.39 = 1.39$

# Aufgabe 2.3 – COCOMO2, d)

KSLOC Berechnung

- Reliability required: von **1.00** auf **1.39**
- Formeln auf Folie 17 (Kapitel 16)

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

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

$$= 1.153$$

| 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

- Reliability required: von **1.00** auf **1.39**

| 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 | |

$_i$ = 1.00 ^ 17 = 1.00

$_i$ = 1.00 ^ 16 * 1.39 = 1.39

# Aufgabe 2.3 – COCOMO2, d)

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

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

$$1.00 * 2.45 * KSLOC_O^B = 1.39 * 2.45 * KSLOC_N^B \quad | /2.45$$

$$KSLOC_O^B = 1.39 * KSLOC_N^B \quad | KSLOC_N^B = (x * KSLOC_O)^B$$

$$KSLOC_O^B = 1.39 * (x * KSLOC_O)^B$$

$$KSLOC_O^B = 1.39 * x^B * KSLOC_O^B \quad | /KSLOC_O^B$$

$$1 = 1.39 * x^B \quad | /1.39$$

$$1 / 1.39 = x^{1.153} \quad | \sqrt[1.153]{}$$

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

$$0.752 = x$$

➔ KSLOC muss um mind. **<u>24.8%</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*

- Risikoabschätzung für das konkrete Projekt
  - Möglicher Verlust (Zeit, Geld, Kontrolle, Qualität)
- **Vorbeugung** gegen Bedrohungen
  - Risiko vermeiden, mindern, auf Andere abwälzen, Planung für Worst Case

- Risiko = Schadenshöhe * Eintrittswahrscheinlichkeit

- Keine Risiken aus dem Skript (z.B. falsche Funktionalität)

# *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 der Spitalleitung
- Übersetzungsprobleme (mehrsprachiges GUI)
- Falscher Technologieeinsatz