ASE 2007

Systematisches Testen

Gruppe E: Markus Graf, Patrick Hornig, Klaus Keller

Inhalt

- Lastenheft
- Pflichtenheft
- Systemübersicht
- Modultest
- Systemtest
- Jacareto
- QF-Test

Lastenheft

- Spezifikation
  - Knapper Text
  - Formel

    springt ins Auge
- Aufgaben
  - Pflichtenheft
  - Inspektion
  - Implementierung
  - Qualitätssicherungsmaßnahmen

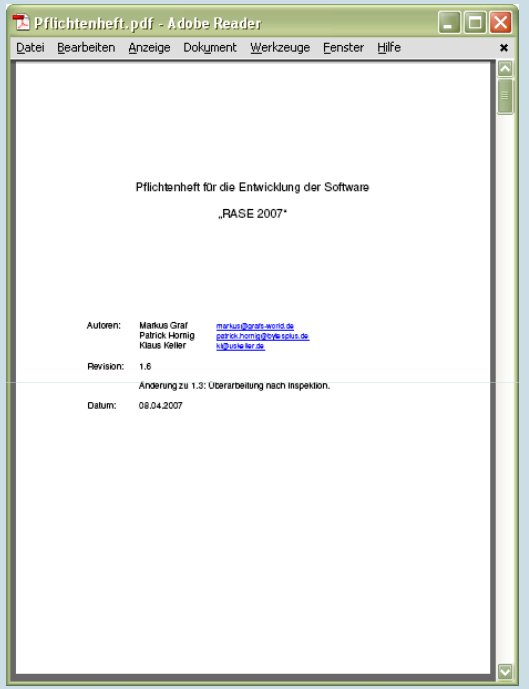
Spezifikation:
Ein Programm zur Ressourcen-Planung einer Arbeitsgruppe soll näherungsweise die prozentuale Auslastungen (AL) eines Mitarbeiters zu einem bestimmten Tag innerhalb eines Geschäftsjahres ermitteln. Die Eingabedaten sind eine beliebige Arbeitspaketnummer, Bearbeitungszeitraum (Z) und Bearbeitungsaufwand (AW) in Personentagen. Der Bearbeitungszeitraum wird durch zwei Eingaben Z=(B, E) vom Typ Datum (dd.mm.yyyy) angegeben. Der Bearbeitungsaufwand ist eine Fließkommazahl mit einer Nachkommastelle.
Für die Planung eines Kalendermonats stehen maximal 20 Arbeitstage (PTM) zur Verfügung. Die Zahl der Planungstage ist ein Erfahrungswert, der nur äußerst selten geändert wird. Die Differenz zwischen geplanten und tatsächlichen Arbeitstagen wird für Urlaub, Krankheit und Selbstverwaltung aufgebraucht und daher in der Planung vernachlässigt. Es wird näherungsweise mit 30 Tagen pro Monat gerechnet. Die Berechnung der Auslastung erfolgt nach dieser Formel:

$$AL = AW \bullet \frac{30}{(E - B + 1) \bullet PTM}$$

Die Oberfläche des Programms soll Bereiche zur Eingabe der benötigten Informationen bereitstellen. Einzelne Mitarbeiter sollen erstellt und gelöscht werden können (die Arbeitsgruppe soll anpassbar bleiben). Für jeden Mitarbeiter soll es möglich sein, Arbeitspakete nach oben beschriebenem Schema einzutragen. Zudem ist die Auslastung quartalsweise visuell darzustellen.

Pflichtenheft - Inspektion

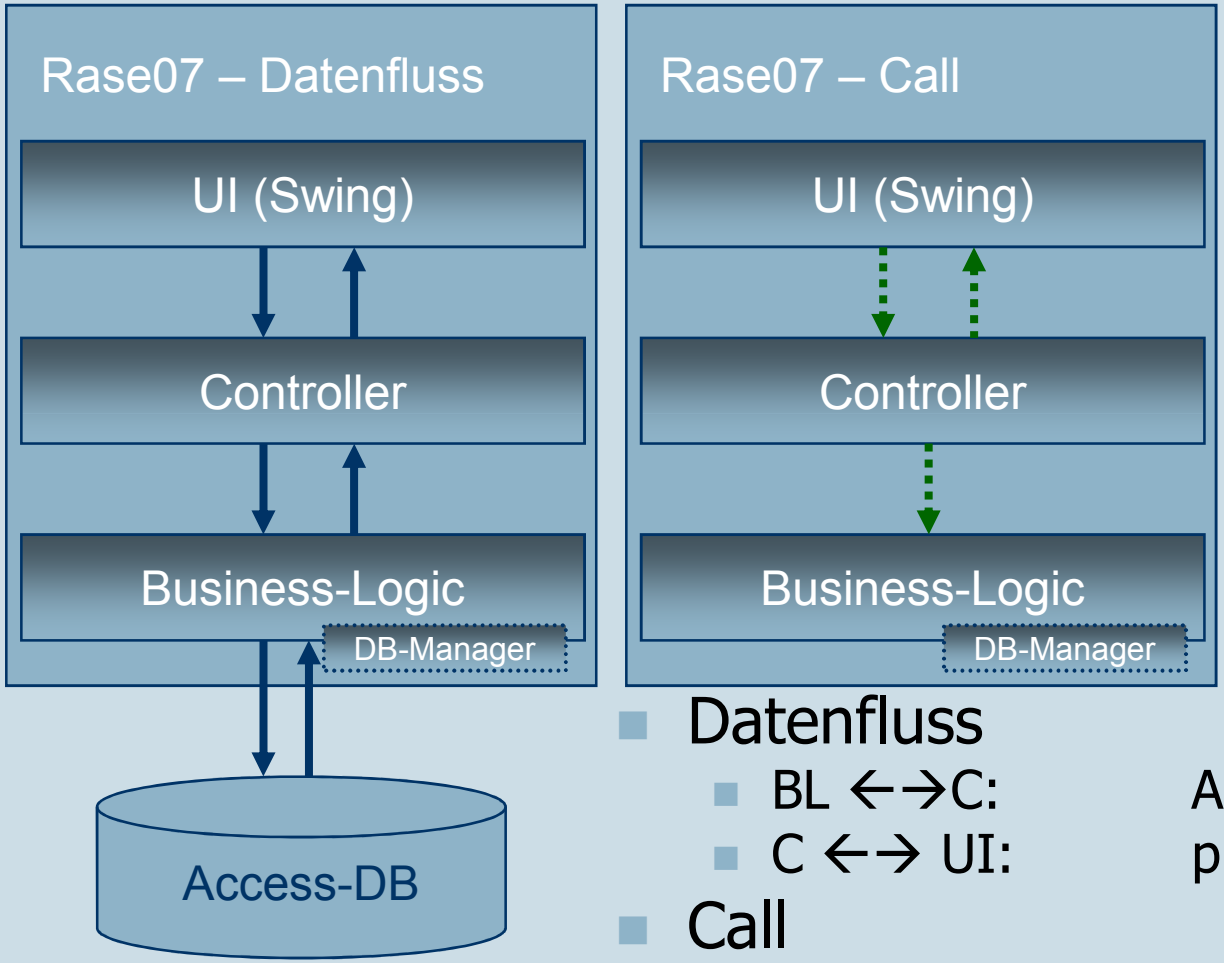
- Umfangreiches Pflichtenheft
  - 16 Seiten
- Statische Analyse um die Qualität zu prüfen → Inspektion
  - Dauer 45 Minuten
  - 5 Teilnehmer
    - 1 Moderator
    - 1 Inspektor
    - 2 Autoren
    - 1 Protokollführer
  - 20 Unklarheiten im Protokoll erfasst
    - 10 geklärt
    - 10 Änderungen am Dokument durchgeführt



Architektur

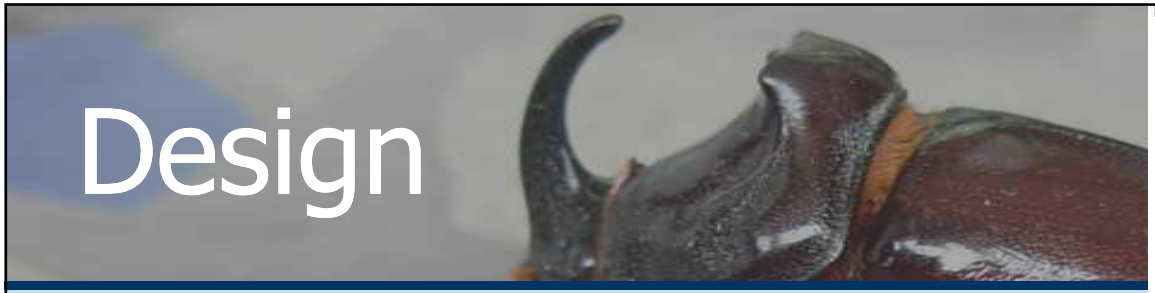
- MVC-Pattern
  - QUASAR-Ansatz
- 3 Personen → 3 Schichten
  - Grafische Oberfläche (Swing)
  - Controller
  - Business-Logic (+ Database-Manager)
- UI „hinter Schnittstelle“
  - UI problemlos wechselbar

Architektur

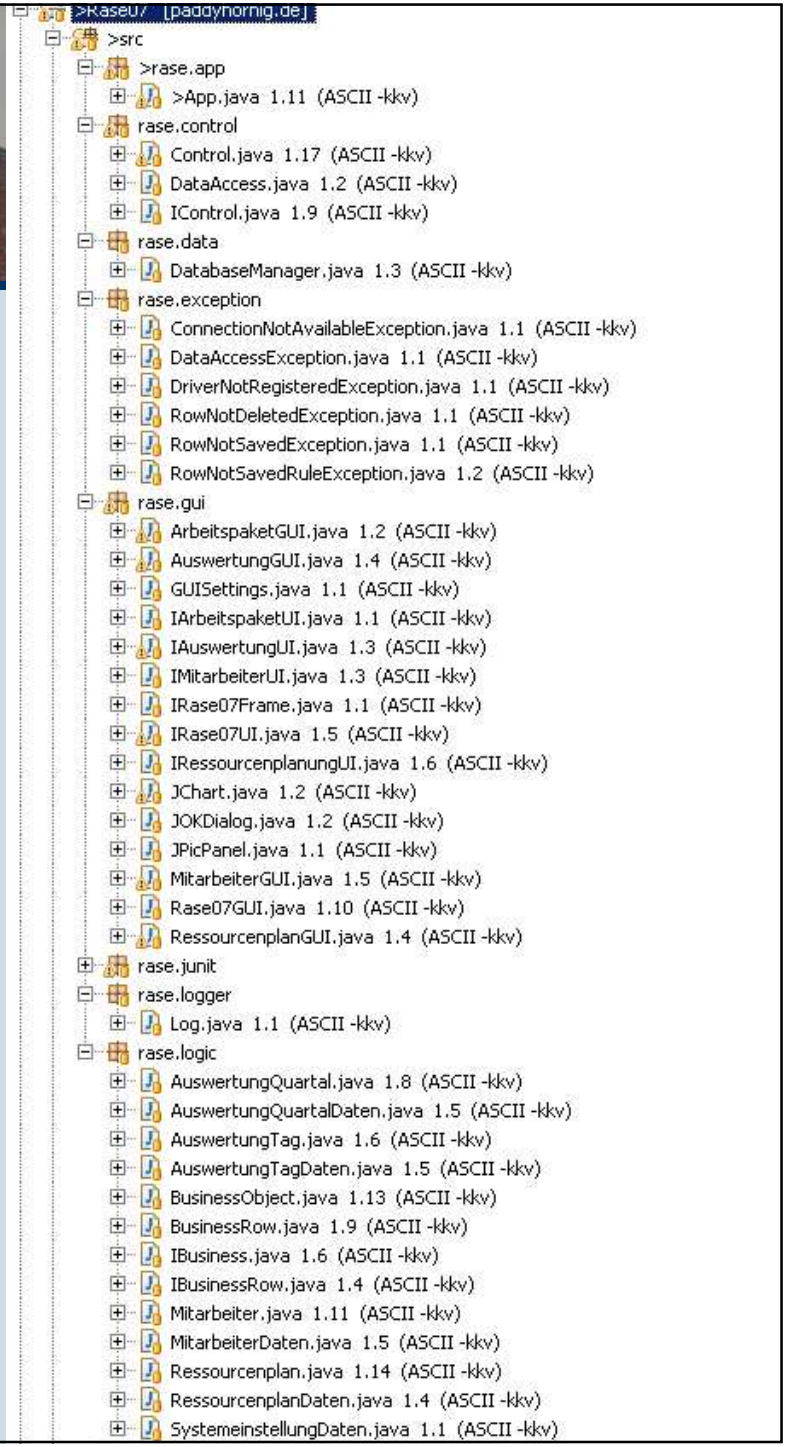


- Datenfluss
  - BL ←→C: ADT
  - C ←→ UI: primitiven Datentypen
- Call
  - UI ←→ C: „QUASAR"
  - C ←→ BL: reine Aufrufe in die BL

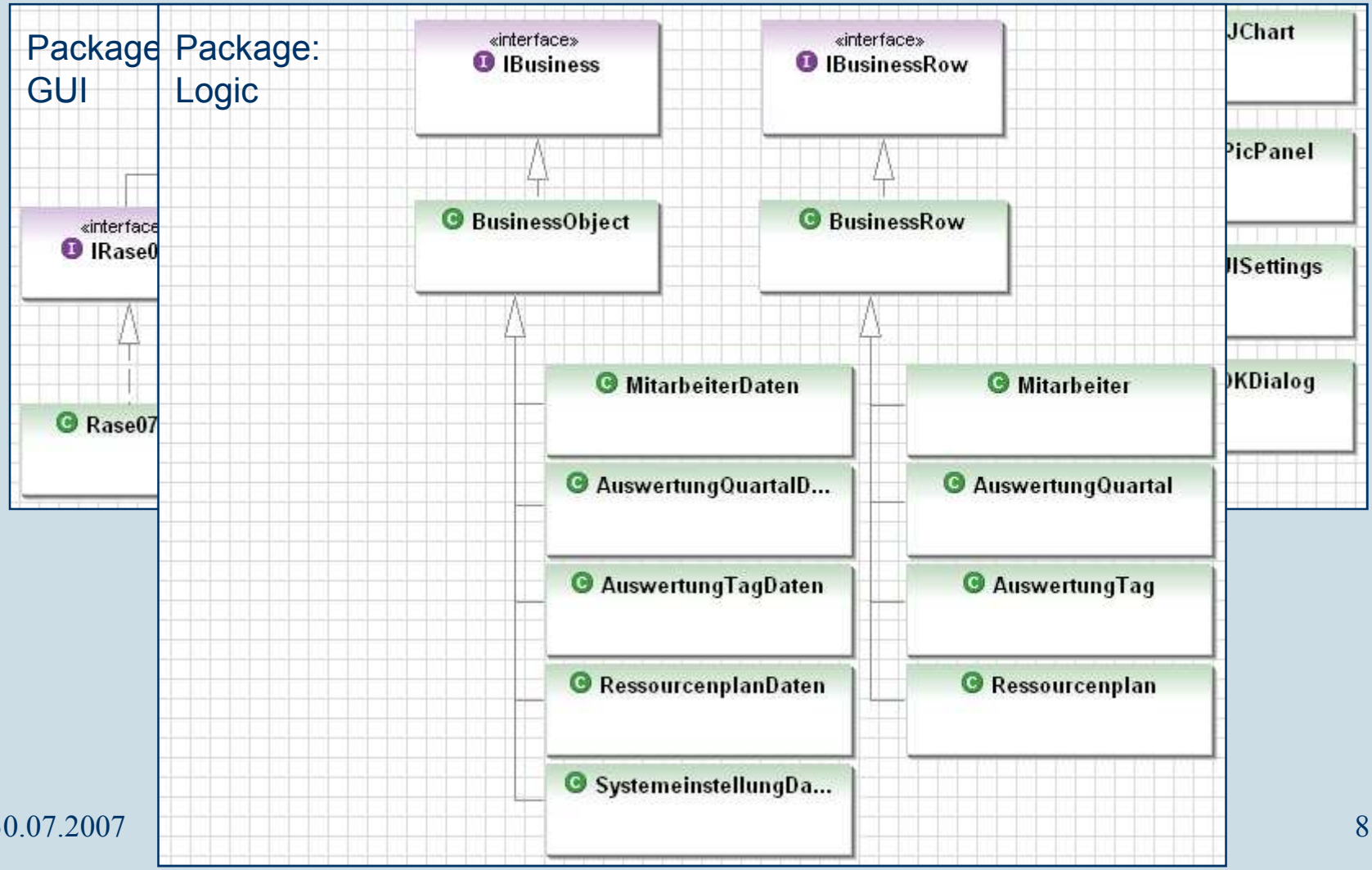
Design



- 7 packages
  - UI, Control und Logic
  - Exception, App, Logger, Data (DB-Manager)
- 9 Interfaces
- 30 Klassen (ohne JUnit-Klassen)



Klassendiagramme



Aufwand

- Überschaubarer Aufwand für die Grundimplementierung (ca. 3 PT)
- Wille zur Perfektion → ca. 8 PT
  - GUI (Chart)
  - Framework:
    - C.R.U.D: create, retrieve, update, delete

Modultest - Erwartungen

- Modultest (White-box)
  - Testfallableitungen
  - → Äquivalenzklassen
  - JUnit & djUnit
- ☺ gutes Design → gut testbar
- ☹ viele Klassen → sehr aufwändig
- ☹ UI schwer durch JUnit / djUnit zu testen

Modultest - Einschränkungen

- Testfälle nur für BL-Schicht
- Keine Tests für UI
  - Grund: leitet Benutzereingaben nur weiter
- und Controller
  - Grund: Benutzereingaben werden nur auf BL-Methoden und „Berechnungen“ der BL auf Anzeige-Methode der UI gemappt

Modultest - Testfallableitungen

Gefundener Fehler:

In Java möglich 31.02.2007 als Datum anzulegen
→ fälschlicherweise als gültige Eingabe akzeptiert.

Spezifik. →

- Funktion
- Eingabe
- Pflichtfeld

	Gültig	**Ungültig**
PTM	1.) {1,2..30}	2.) <1 3.) >30
Pers.nr	4.) *	5.) *leer*
Name	6.) *	7.) *leer*
AW	8.) 0<=AW	9.) AW<0
...	...	

→

	I	*II*	*III*	*IV*	*V*	*VII*	*VIII*	*...*
PTM	1	15	30	0	31	30	30	
Pers.nr.	01	02	03	01	01			
Name	A	B	C	A	A	A		...
AW	1.0	1.0	1.0	1.0	1.0	1.0	1.0	
...	...							

Modultest mit JUnit / djUnit

JUnit

- ☺ Wenig Aufwand, Konsequent
  → Unverzichtbar
- ☺ Viele Äquivalenzklassen in einer Testmethode
- ☹ Programmierarbeit!
  - Nie Fehlerfrei
  - → *Fehler in Testimplementierung*
- ☹ Daraus folgt möglicher:
  - Minus * Minus – Test
  - Siehe Systemtest
    AW als ganze Zahl!
- ☹ Exceptions
  - Bei starker Schachtelung von Exceptions zu aufwendig (Kosten / Nutzen)

djUnit

- ☺ Erkennen von lückenhaften Tests (Anweisungsüberdeckung)
- ☺ Kein Mehraufwand zu JUnit
- ☹ Tests und Risiko
  - → „Unwahrscheinliche" Exception
  - → Einfache Tests ≠ Wichtige Tests
- ☹ Grüne Balken (100%) verleiten
  zu denken, man ist fertig
- ☹ Rote Balken (<100%) verleiten
  „die Methode halt mal aufzurufen"
- ☹ Reuse von Klassen
  Manche Codeabschnitte werden nicht benötigt

Systemtest – exemplarischer Testfall

Pflichten-heft

AW	Der Bearbeitungsaufwand (die aufgewendeten Personentage des Mitarbeiters). Eine Fließkommazahl mit einer Nachkommastelle.

	Gültig	**Ungültig**
...	...	
AW	*5.)* 7.0 *6.)* 0.0 *7.)* 2.2 8.) -9.0	*9.)* 2.25
...	...	

	I	II	III	IV	V	VI	VII	...	IX
...	...								
AW	7.0	7.0	7.0	0.0	2.25	2.2	-9.0	...	0.0
...	...								
Ä.K	1., 3., **5.**, 10.	2., 3., **5.**, 10.	1., 4., **5.**, 10.	1., 3., **6.**, 10.	1., 3., **9.**, 10.	1., 3., **7.**, 10.	1., 3., **8.**, 10.	...	1., 3., **6.**, 11.

Systemtests - Auswertung

Test	AW	B	E	*PTM*	AL [in %]	AL (correct)
I	7,0	12.01.2007	22.01.2007	*20*	95,45	95,45
II				***nicht***		
III				***änderbar***		
IV	0,0	01.05.2007	30.05.2007	*20*	0	0,00
V	2,25	01.05.2007	30.05.2007			
VI	2,2	01.05.2007	30.05.2007			
VII	-9,0	01.05.2007	30.05.2007	*20*	-45	-45,00
VIII	0	16.05.2007	66.05.2007	*20*	Meldung	Meldung
IX	0	30.05.2007	01.05.2007	*20*	Meldung	Meldung

AL= 2,2 * 30 / (30 * 20) = 11

10 % entspricht 2 * 30 / (30 * 20)

Gefundener Fehler gegen die Spezifikation:

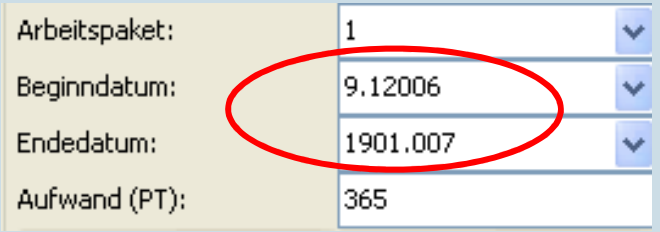
AW:
Der Bearbeitungsaufwand wird nicht als Fließkommazahl betrachtet, obwohl so spezifiziert wurde („Eine Fließkommazahl mit einer Nachkommastelle“).

Jacareto

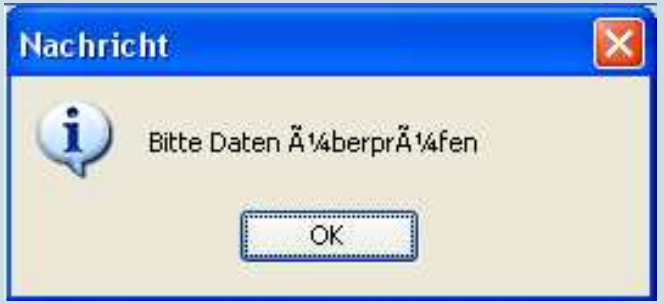
- ☺ Automatisieren von Blackbox Tests
- ☹ Umständliche Konfiguration
- ☹ nicht intuitiv bedienbar
- ☹ Beenden Anwendung inkl. CleverPHL
- ☹ Leere Maske
  - → Zwangsrefresh des Fensters
- ☹ Leere Meldungsfenster
  - → kein Zwangsrefresh möglich

Automatisierte Systemtests

- Replay
  - Eingaben gehen verloren (ConcurrentModificationExc.)



  - GUI-Elemente werden nicht mehr gefunden oder sind nicht mehr ansprechbar (z.B. durch Modalen Dialog) → Timeout



Jacareto: CleverPHL und Picorder

CleverPHL

☺ Aufnahmewerkzeug mit guten Ansätzen

☹ keine nutzbaren Erkenntnisse, Steuerelemente, Meldungen bei Replay nicht angezeigt
→ Automatisierter Lauf unmöglich

☹ Eingaben gehen „verloren“
- → Wie hilft das Werkzeug dann zur Fehlerfindung?
- → Für diesen Zweck nicht wirklich einsetzbar

Picorder

☺ Kommandozeilenorientiert
- → Automatisierbar starten

☹ Direkt in XML konfigurieren oder über CleverPHL konfigurieren und dann aus *cleverphl_data|starters* ins Verzeichnis *jacareto|starter* kopieren → Achtung: Adminrechte!

☹ siehe Nachteile von CleverPHL

QF-Test - Quality First Software

- Analyse von QF-Test
  (Quality First Software GmbH)

- Aus dem Handbuch von QF-Test

> … „Mit Hilfe von QF-Test können Funktionalitätstests von Java Programmen automatisiert werden, sofern diese eine grafische Benutzeroberfläche aufweisen.“ …

QF-Test - Verfügbarkeit

- Kostenlose Version verfügbar unter
  - http://www.qfs.de/de/qftest/download.html
- Einschränkungen ohne Lizenz
  - Es können keine Testsuites gespeichert werden.
  - Es können nur die Testsuites geladen werden, die mit QF-Test ausgeliefert werden.
- Problemloses anfordern einer 4-Wochen-Evaluationslizenz
  - ☺ 2 Tage menschliche Bearbeitungszeit
  - ☹ Bei Verwendung von freemail-Adressen nur einwöchige Evaluationslizenz, nach Rückfrage 4 Wochen Lizenz erhalten
- Supportangebot kommt per Anruf
  - ☺ Voller Support während der Evaluationsphase
  - ☺ Dieser wird aktiv angeboten

QF-Test - Installation & Einrichten

- Einfache Installation
  - ☺ ohne Handbuch möglich
  - ☺ Tutorials
  - ☹ JDK / JRE „Instrumentieren“
    - Was ist das → Handbuch!
    - Beim 1. Start → Administratorrechte notwendig!
- Starten des „System Under Test“ ist leicht
  - ☺ Assistent führt mit wenig Mausklicks zum Ziel
  - ☺ Richtiger Knoten im Baum wählen ist intuitiv
- Einrichten von Testfällen
  - ☺ Sehr intuitiv trotz vielfältiger Möglichkeiten

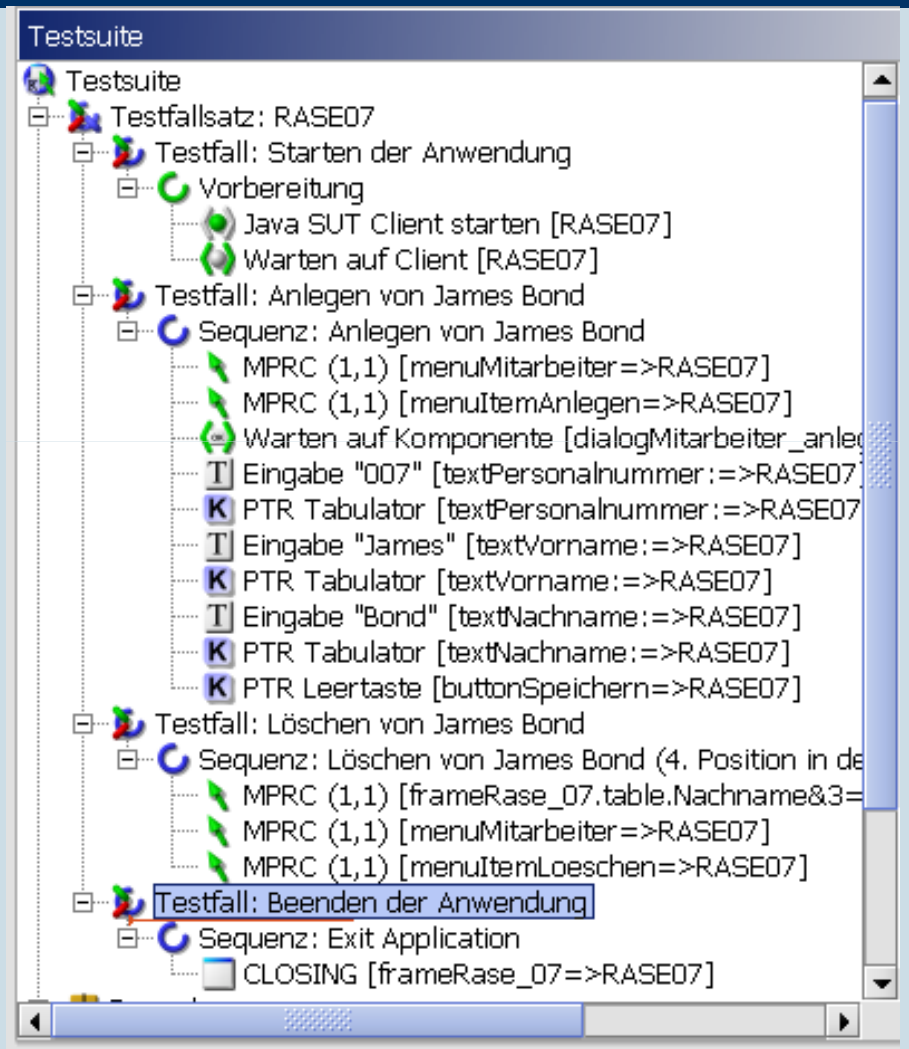
QF-Test - Bedienung

- Vorteile
  - ☺ Kontextmenüs vereinfachen Bedienung
  - ☺ Knotennamen sind eindeutig und intuitiv
  - ☺ Verschieben von Knoten im Baum möglich
  - ☺ Vordefinierte Knotenstruktur als Ausgangspunkte
  - ☺ Gute Hilfe und Tutorials
  - ☺ Übersichtlich

QF-Test - Bedienung

- Nachteil
  - ☹ Neuer Knoten (Aufnahme, Assistent) wird nicht am ausgewählten Knoten eingefügt, sondern unter Extrasequenzen
  - ☹ Manchmal wird die Vorbereitung eines Tests nicht gestartet

QF-Test - Testfall: Anlegen/löschen Benutzer



- Anlegen
  - Problem: ID Schon vergeben?
- Löschen
  - Problem: Richtigen Satz ausgewählt

QF-Test - Fazit Testfälle

- Testen von Datenbankanwendungen schwierig
  - Neuanlage und Löschen
  - → Viel Überlegungen wg. Konsistenz
  - → Zu löschender Satz ausgewählt? Wirklich?
    - → Bei ASE Projekt nicht möglich, da
    - Keine Sortierung von Daten
    - keine Suche (um evtl. anhand der ID zu löschen)
  - Neuanlage wenn ID schon vergeben
    - → Unvorhergesehene Meldung erscheint → Testsuite hält, da plötzlich anderes Verhalten zugrunde liegt

QF-Test versus Jacareto

	
Einfache Installation und Konfiguration	Komplizierte Installation und Konfiguration
Intuitive Bedienung	Eher schwierige Bedienung
Preis je nach Anzahl der Lizenzen von €1.895 - €18.857,50	Kostenloses Open Source Projekt
Ausgereiftes Produkt mit Support-Unterstützung	Eher Beta Version (neueste Version Jacareto 0.7.12)

Ende

Vielen Dank.

Gibt es noch Fragen?