TEST-UML

FernUniversität
Gesamthochschule in Hagen

SOFTWARE ENGINEERING
Praktische Informatik III

Testen und UML

Dr. Mario Winter FernUniversität Hagen

SOFTWARE ENGINEERING
Praktische Informatik III

Informatik an der FernUni

- Studienmöglichkeiten
  - Integrierter Diplomstudiengang Informatik
  - Modellstudiengang Bachelor of Science in Informatik
  - Zusatzstudiengang Praktische Informatik
  - Vorbereitende Studien für Erweiterungsprüfung zum Lehramt (Sekundarstufe I/II) im Fach Informatik
  - Informatik-Fächer im Diplomstudiengang Mathematik

Alle Kurse der Informatik zur Weiterbildung geeignet

Im Rahmen des aktuellen Lehrangebots belegbar

Strukturierte Weiterbildung

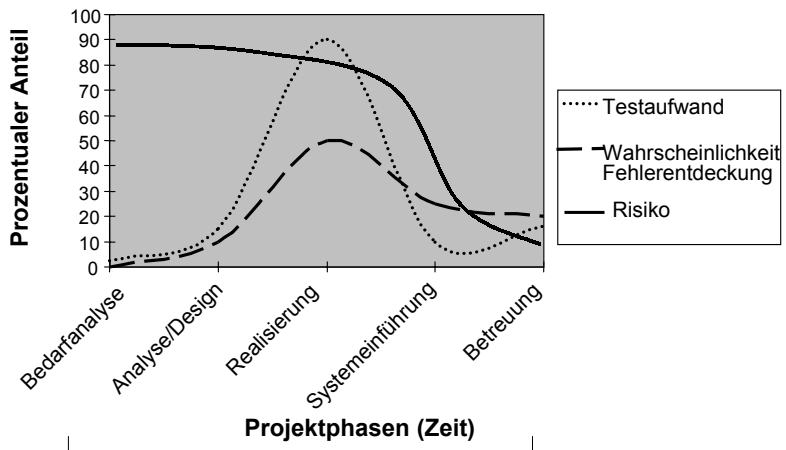
Weiterbildungspakete aus thematisch zusammenhängenden Kursen für bestimmte Adressatengruppen

In zwei bis vier Semestern studierbar

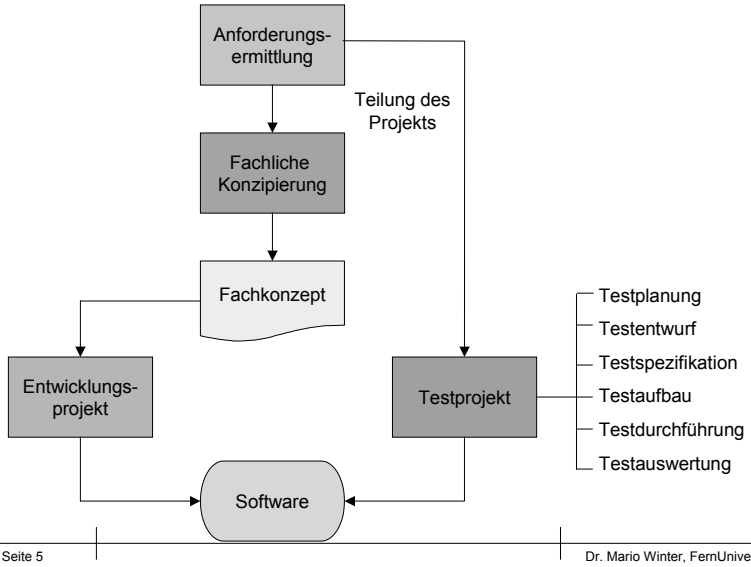
Überblick

- Warum projektbegleitend Testen?
- Was ist die UML?
- UML: Modell, Sichten, Diagramme
  - Funktionssicht
  - Struktursicht
  - Verhaltenssicht
- UML-Diagramme aus dem Blickwinkel des Testers ...
  - Anwendungsfalldiagramm und Aktivitätsdiagramm
  - Klassendiagramm und Objektdiagramm
  - Sequenzdiagramm und Kollaborationsdiagramm
  - Zustandsdiagramm
  - Komponentendiagramm und Verteilungsdiagramm
- Die Object Constraint Language OCL
- Zusammenfassung

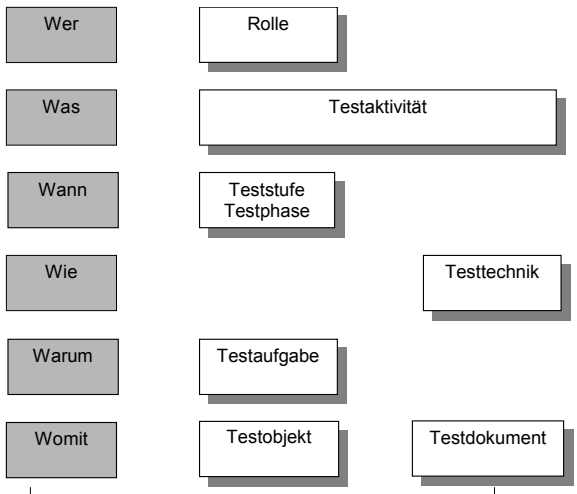
Warum Testen?



Projektbegleitend Testen!



Testprojekt / Testprozess

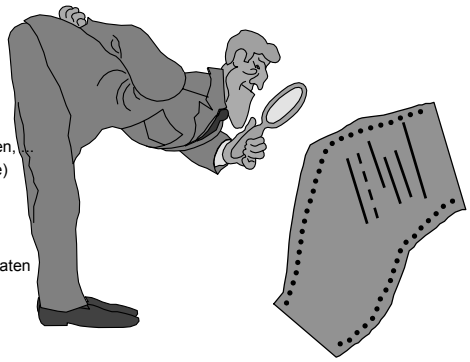


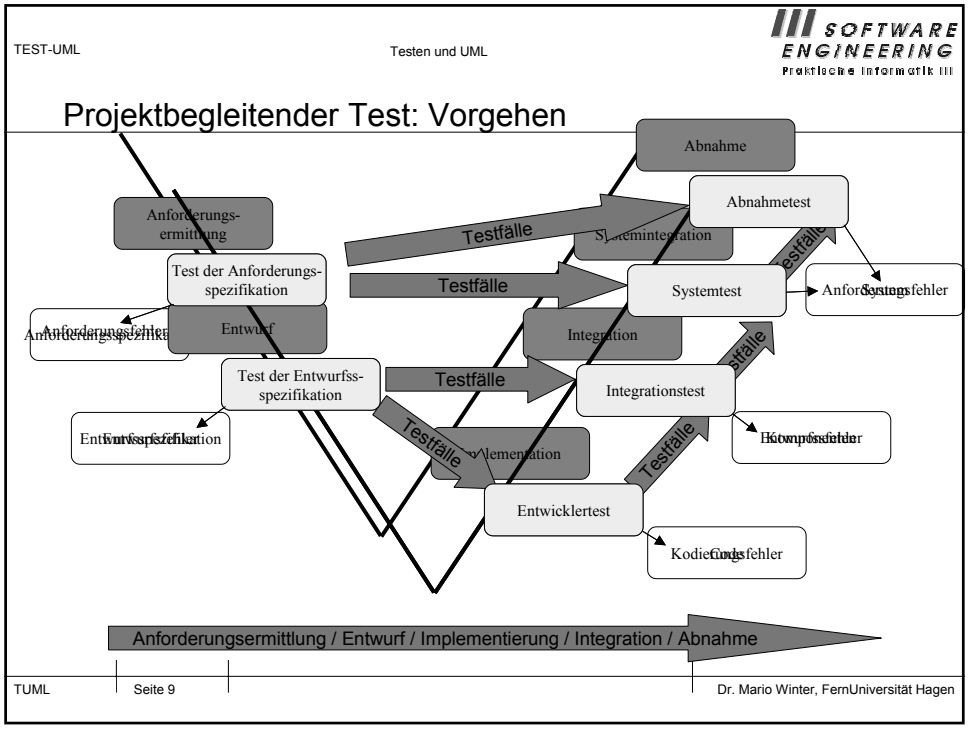
Teststufen und Testphasen

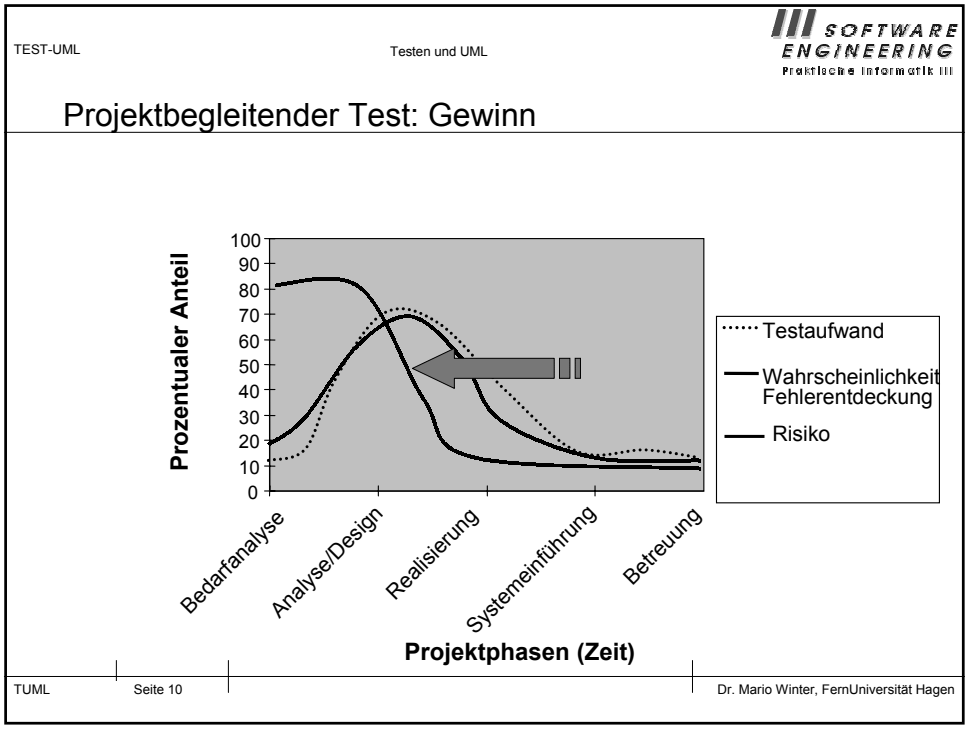
- Teststufen
  - Entwicklertest des Entwurfs
  - Entwicklertest der Klassen
  - Integrationstest der Klassen
  - Funktionstest der Komponenten
  - Integrationstest der Komponenten
  - Funktionstest des Systems
  - Integrationstest des Systems
  - Abnahmetest
- Testphasen
  - Testplanung
  - Testentwurf
  - Testinfrastrukturaufbau
  - Testfallermittlung
  - Testdatenermittlung
  - Testprozedurerstellung
  - Testausführung
  - Testauswertung

Testtechniken

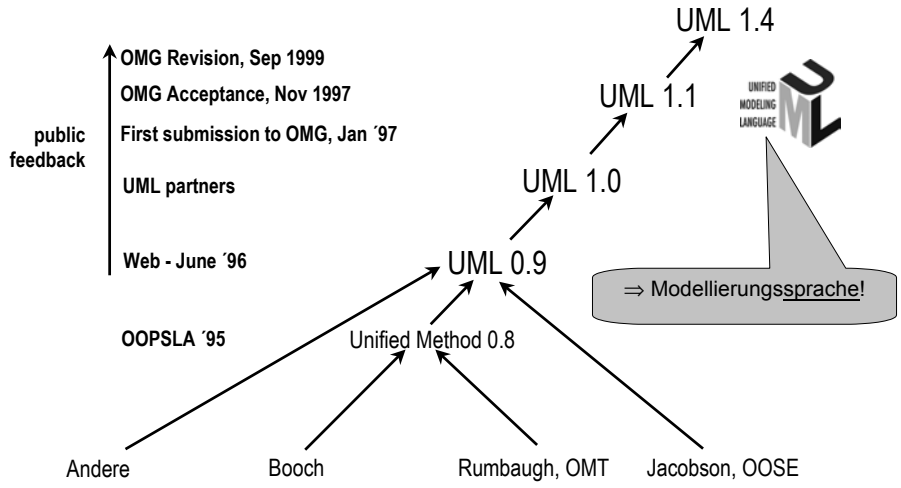
- Statisch
  - Reviews, Inspektionen
  - Walk Throughs
- Dynamisch
  - Funktional
    - Grenzwerte, Äquivalenzklassen, ...
    - Schnittstellenbasiert (Verträge)
    - Szenarienbasiert
  - Zustandsbasiert
    - Über Instanzvariablen
    - Über explizite Zustandsautomaten
  - Strukturell
    - Kontrollfluss
    - Daten/Objektfluss
    - Botschaftsfluss




TEST-UML
Testen und UML
SOFTWARE ENGINEERING
Praktische Informatik III
Projektbegleitender Test: Vorgehen
Abnahme
Abnahmetest
Anforderungs-ermittlung
Testfälle
Test der Anforderungs-spezifikation
Testfälle
Systemtest
Entwurf
Integration
Test der Entwurfss-spezifikation
Testfälle
Integrationstest
Testfälle
Testfälle
Entwicklertest
Anforderungsermittlung / Entwurf / Implementierung / Integration / Abnahme
TUML
Seite 9
Dr. Mario Winter, FernUniversität Hagen


TEST-UML
Testen und UML
SOFTWARE ENGINEERING
Praktische Informatik III
Projektbegleitender Test: Gewinn
Prozentualer Anteil
100
90
80
70
60
50
40
30
20
10
0
Testaufwand
Wahrscheinlichkeit Fehlerentdeckung
Risiko
Bedarfanalyse
Analyse/Design
Realisierung
Systemeinführung
Betreuung
Projektphasen (Zeit)
TUML
Seite 10
Dr. Mario Winter, FernUniversität Hagen

Was ist die UML?

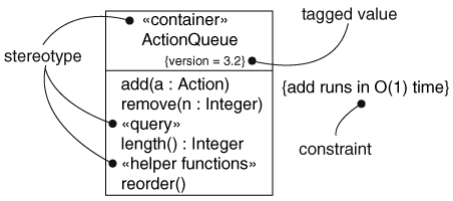


UML - Modellierungselemente

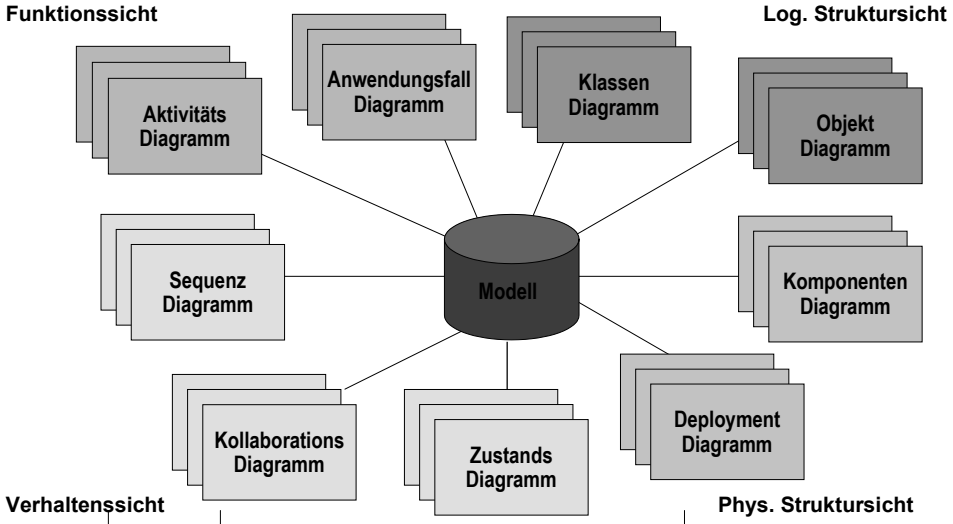
- Strukturelle Elemente
  - Klasse, Interface, Kollaboration, Anwendungsfall, Aktive Klasse, Komponente, Knoten
- Beziehungen
  - Abhängigkeit
  - Assoziation (Aggregation, Komposition)
  - Generalisierung
  - Realisierung
- Verhaltensorientierte Elemente
  - Interaktion, Nachricht, Aktivierung, Aktivität, Zustandsmaschine
- Gruppierende Elemente
  - Paket, Teilsystem
- Andere Elemente
  - Notiz

UML - Erweiterungsmechanismen

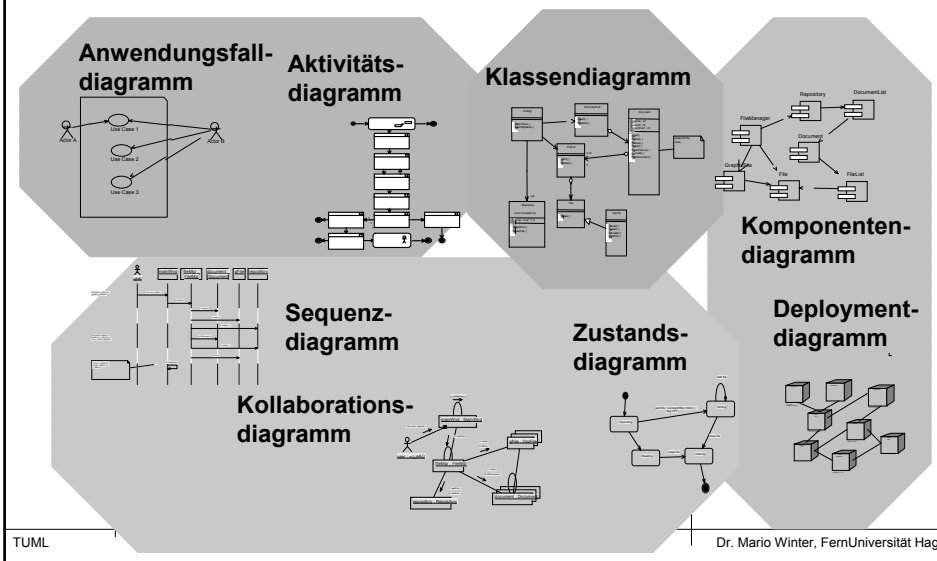
- Stereotype
- Tagged values
- Constraints



UML: Modell, Sichten und Diagramme

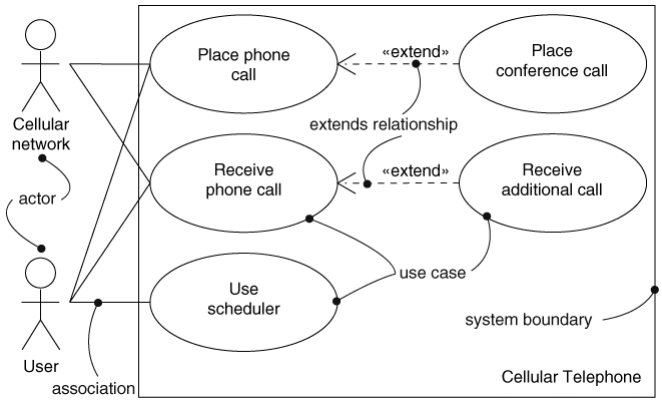


Visuelle Modellierung mit UML Diagrammen



Anwendungsfalldiagramm

- Von außen beobachtbare Systemfunktionalität

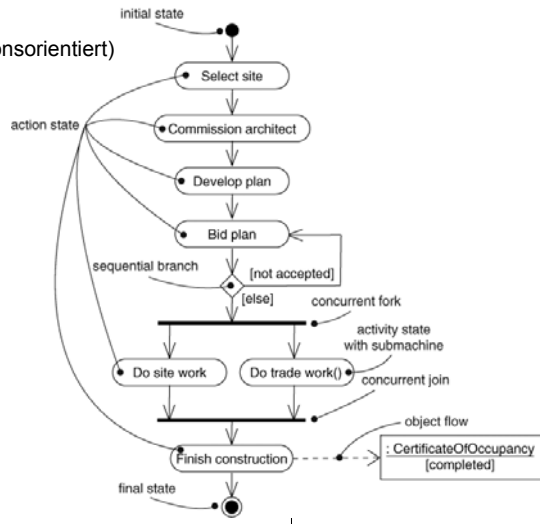


Anwendungsfalldiagramm

- Von außen beobachtbare Systemfunktionalität
- Anforderungsermittlung
- Zweck
  - Spezifiziert den Kontext eines Systems
  - Erfasst die funktionalen Anforderungen an das System
  - Validierung der Systemarchitektur
  - Treibt die Implementation und dient zur Testfallgenerierung
- Erstellt von Analytikern und Domänenexperten
- Testaspekte
  - Grundlage des System- und Abnahmetests
  - Sollte nicht-funktionale Anforderungen für Performanz- und Lasttests umfassen
  - Strukturierung der ”Testware”

Aktivitätsdiagramm

- Dynamisches Verhalten (Aktionsorientiert)

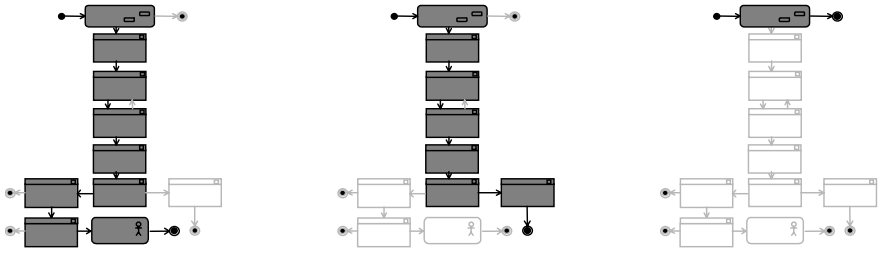


Aktivitätsdiagramm

- Enthält das dynamische Verhalten (Aktionsorientiert)
- Zweck
  - Modellierung von Geschäftsprozessen (business workflows)
  - Modellierung des Ablaufs von Anwendungsfällen
  - Modellierung des Ablaufs von Operationen
- Erstellt von Analytikern und Entwicklern
- Testaspekte
  - System- und Abnahmetest
  - Grundlage für konkrete Testszenarien

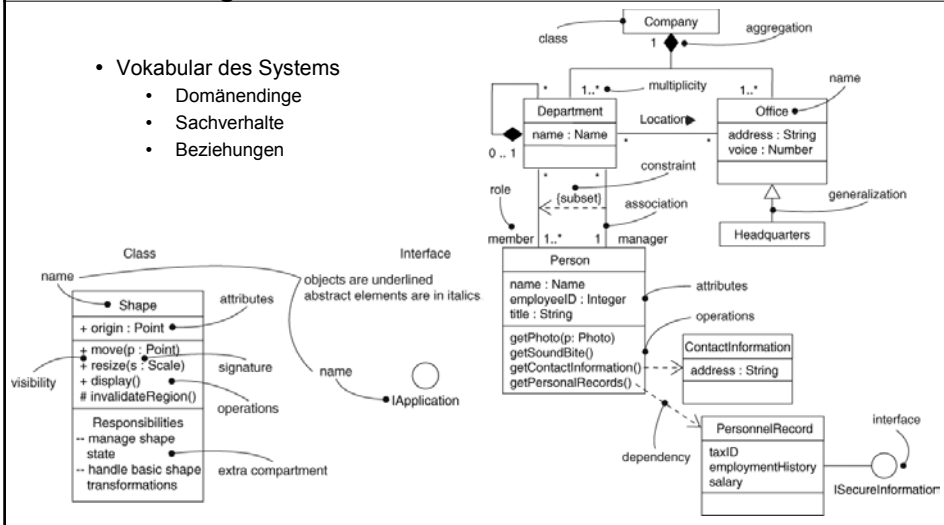
Ermittlung von Testfällen aus Aktivitätsdiagrammen

- Testszenarien so auswählen, dass die geforderte Überdeckung des Aktivitätsdiagramms erzielt wird
- Anzahl der zu testenden Szenarien bezüglich der Überdeckung aller Knoten (Aktionen) und Kanten (Übergänge) im Aktivitätsdiagramm minimieren
- Überdeckungsmetriken zur Generierung weiterer Testszenarien verwenden.



Klassendiagramm

- Vokabular des Systems
  - Domänendinge
  - Sachverhalte
  - Beziehungen

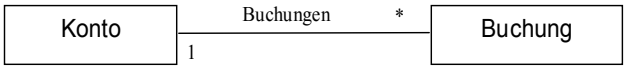


Klassendiagramm

- Enthält das Vokabular des Systems
- Aufgebaut und verfeinert während der Entwicklung
- Zweck
  - Bezeichnet und modelliert wichtige Domänendinge
  - Spezifiziert Zusammenhänge
  - Spezifiziert das logische Datenbankschema
- Erstellt von Analytikern, Architekten und Entwicklern
- Testaspekte
  - Entwicklertest und Integrationstest
  - Strukturelle Tests bzgl. erlaubter Objektkonstellationen
  - Ermittlung einer Integrations-Strategie

Ermittlung von Testfällen aus Assoziationen

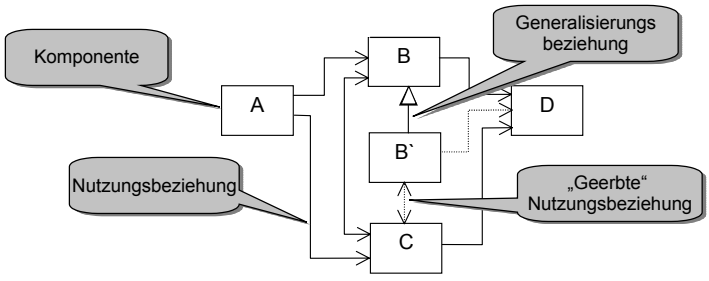
- Testfallmatrix, in der für eine Assoziation/Aggregation/Komposition die zu testenden Anzahlen verbundenen Instanzen angegeben werden



Konto / Buchung	Min. – 1=0	Min.=Max.= typisch = 1	Max. + 1 = 2
Min (0)	N/A	OK	N/A
Typisch (z.B. 2000)	Fail	OK	Fail
Max. (z.B. 2^16)	-	OK	-
Max +1	-	Fail	-

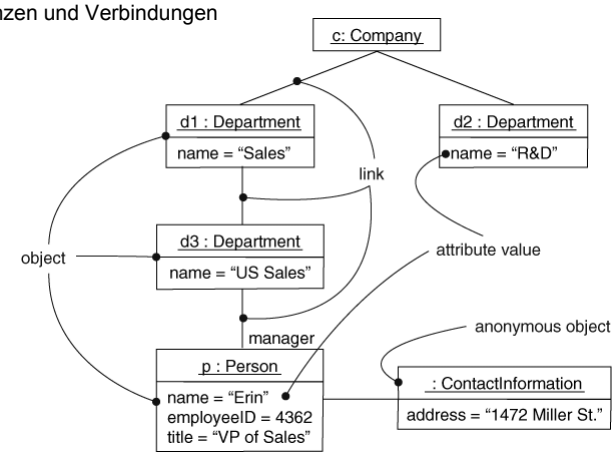
Ermittlung einer Integrationsstrategie

1. Fokussierung auf die zu integrierenden Komponenten
   - Klassendiagramm, Interaktions- sowie Komponentendiagramme betrachten
   - Menge zu integrierender Komponenten eingrenzen
   - Nutzungs- und ggf. Generalisierungsbeziehungen ermitteln
   - Ergebnis: „Komponentengraph“
2. Benutzungshierarchie bottom-up, Generalisierungshierarchie top-down



Objektdiagramm

- Instanzen und Verbindungen



Objektdiagramm

- Enthält Instanzen und Verbindungen (links)
- Erstellt bei Anforderungsermittlung und Entwurf
- Zweck
  - Illustriert Daten-/Objektstrukturen
  - Speziefiziert Augenblickskonstellationen (snapshots)
- Erstellt von Analytikern, Architekten und Entwicklern
- Testaspekte
  - Entwicklertest und Integrationstest
  - Strukturelle Tests bzgl. besonderer Objektkonstellationen

Interaktionsdiagramme

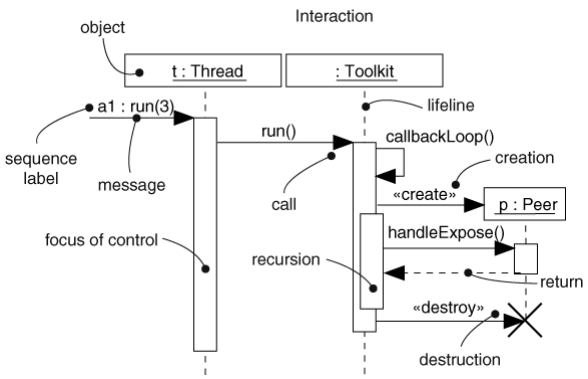
Sequenzdiagramm
- Ablauforientierte Modellierung dynamischen Verhaltens
- Zweck
  - Modellierung von Operationsabläufen (Kontrollfluss)
  - Illustration typischer Szenarien

Kollaborationsdiagramm
- Nachrichtenorientierte Modellierung dynamischen Verhaltens
- Zweck
  - Modellierung von Operationsabläufen (Kontrollfluss)
  - Illustration der Koordinierung in Objektstrukturen

- Erstellt von Analytikern, Entwicklern und Testern
- Testaspekte
  - Entwicklertest und Integrationstest
  - Ablauforientierter Test

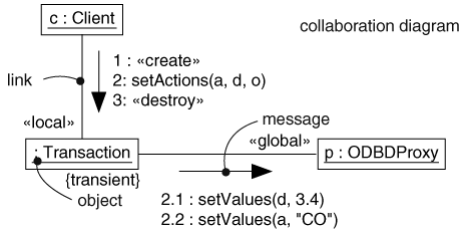
Sequenzdiagramm

- Dynamisches Verhalten (Ablauforientiert)



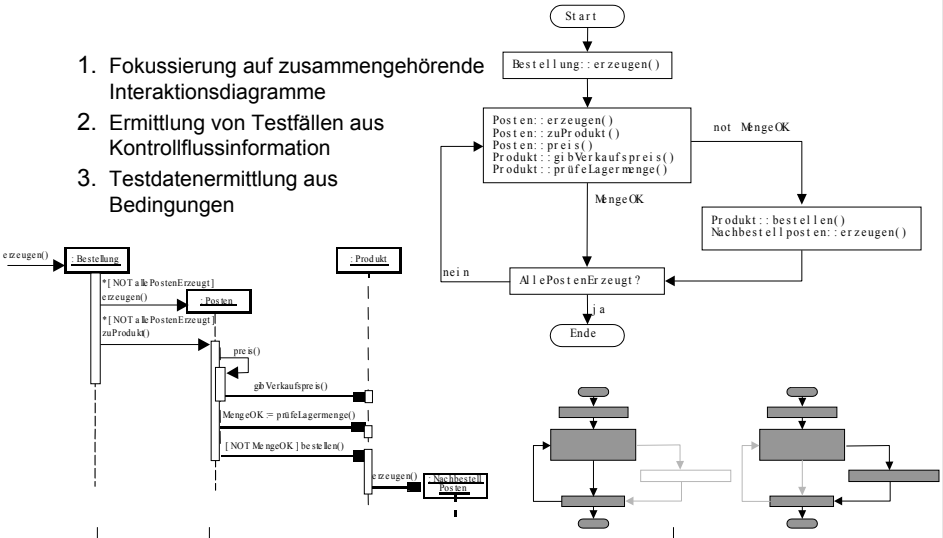
Kollaborationsdiagramm

- Dynamisches Verhalten (Nachrichtenorientiert)



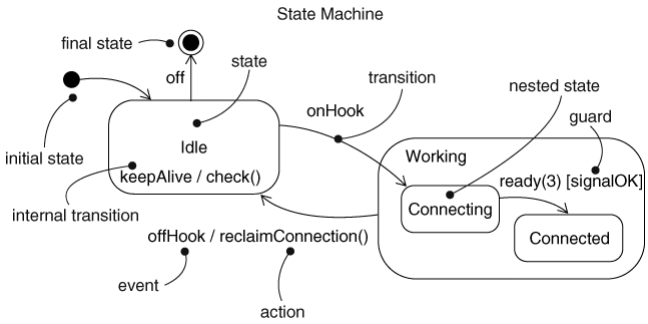
Ermittlung von Testfällen aus Sequenzdiagrammen

1. Fokussierung auf zusammengehörende Interaktionsdiagramme
2. Ermittlung von Testfällen aus Kontrollflussinformation
3. Testdatenermittlung aus Bedingungen



Zustandsdiagramm (Statechart)

- Dynamisches Verhalten (Ereignisorientiert)

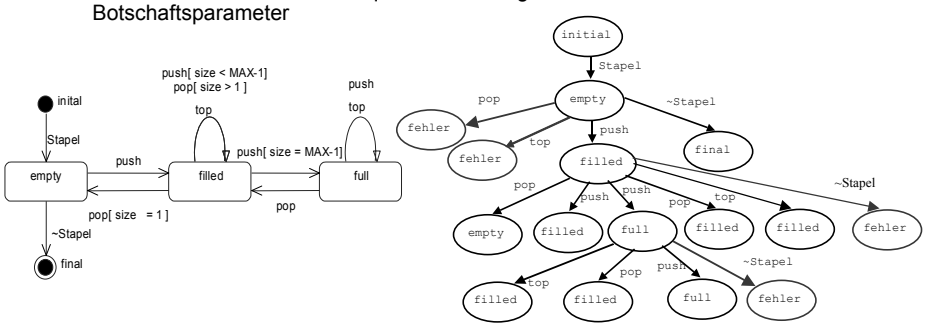


Zustandsdiagramm (Statechart)

- Enthält das dynamische Verhalten (Ereignisorientiert)
- Zweck
  - Modellierung des Objektverhaltens (object lifecycle)
  - Modellierung reaktiver Objekte (Benutzungsschnittstellen, Geräte, etc.)
- Erstellt von Analytikern, Entwicklern und Testern
- Testaspekte
  - Entwicklertest und Funktionstest (Systemtest)
  - Test bzgl. des konformen Verhaltens
  - Robustheitstest

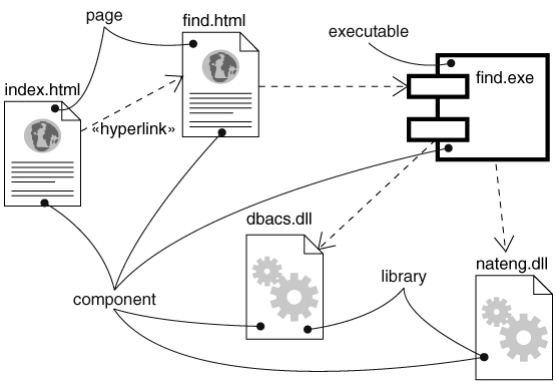
Ermittlung von Testfällen aus Zustandsdiagrammen

1. Fokussierung auf das Zustandsdiagramm
2. Ableiten des Übergangsbaumes für den Zustands-Konformanztest
3. Erweitern des Übergangsbaumes für den Zustands-Robustheitstest
4. Generieren der Botschaftssequenzen und Ergänzen der Botschaftsparameter



Komponentendiagramm

- Physikalische Struktur der Implementation

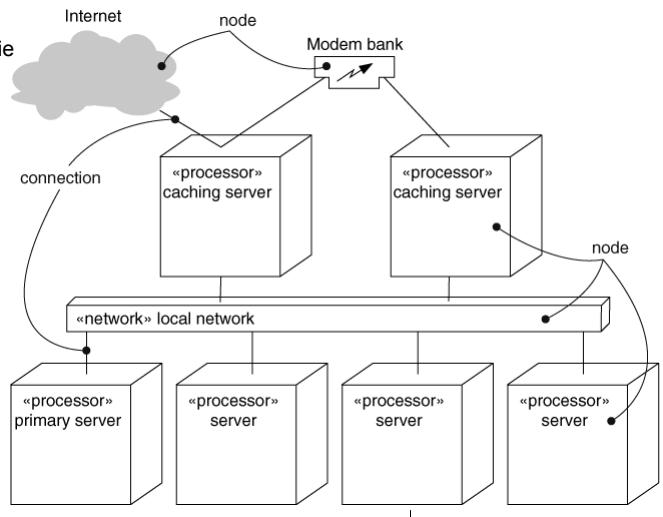


Komponentendiagramm

- Enthält die physikalische Struktur der Implementation
- Teil der Architekturspezifikation
- Zweck
  - Organisation des Quellcodes
  - Erzeugung eines ausführbaren Releases (build)
  - Spezifikation der physikalischen Dataenbank
- Erstellt von Entwicklern
- Testaspekte
  - Installationstest, Systemtest, Abnahmetest
  - Organisation der Testware
  - Ermittlung einer Integrationsstrategie

Deploymentdiagramm

- Hardwaretopologie



Deploymentdiagramm

- Enthält die Topologie der Hardware
- Teil der Architekturspezifikation
- Zweck
  - Spezifiziert die Verteilung der Komponenten
  - Identifikation von Flaschenhälsen (performance bottlenecks)
- Estelllt von Architekten, Netzwerkspezialisten, and Systemingenieuren
- Testaspekte
  - Installationstest, Systemtest
  - Sicherheits-, Performanz- und Lasttest

Object Constraint Language OCL

- Deklarative Spezifikationssprache
- Modellbasierte Spezifikationen
  - Atomare Datentypen
  - Logische Ausdrücke
  - Mengentheoretische Operationen
- Zweck
  - Zusicherungen für Modellelemente
  - Klassen- und Operationsspezifikationen
- Estellt von Analytikern und Entwicklern
- Testaspekte
  - Präzise Spezifikation von Testfällen und Testdaten
  - Funktionale Tests

Testfallermittlung aus OCL-Operationsspezifikationen

Klasseninvariante

Spalte = Testfall

Context ExtendedStack	A	B	C	D	E	F
invariant@ self.size() >= 0 **AND**	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
invariant@ self.size() <= self.MAXSIZE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
ExtendedStack() pre@ true	TRUE					
ExtendedStack()post@ self.size() = 0 **AND**	TRUE					
ExtendedStack()post@ self.MAXSIZE = maxSize	TRUE					
push() pre@ self.size() < self.MAXSIZE	TRUE	TRUE				
push() post@ self.size()=self.size()@pre+1	–	TRUE				
top() pre@ self.size() > 0	FALSE	TRUE	TRUE			
top() post@ return != null	–	–	TRUE			
pop() pre@ self.size() > 0	FALSE	TRUE	TRUE	TRUE		
pop() @post self.size() = self.size()@pre – 1	–	–	–	TRUE		
all() pre@ true	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
all() post@ self.size() > 0 **AND**	–	–	–	–	FALSE	TRUE
all() post@ return.size() = self.size() **OR**	–	–	–	–	dc	TRUE
all() post@ **not** self.size() > 0 **AND**	–	–	–	–	TRUE	FALSE
all() post@ return = null	–	–	–	–	TRUE	dc

Vor-bedingung

Nach-bedingung

Don't Care

Testen und UML

- "Lingua Franca" für Entwickler und Tester
- Hilfe in allen Teststufen
  - ++ Entwicklertest
  - + Funktionstest
  - + Integrationstest
  - + System- und Abnahmetest
  - - Sicherheits-, Performanz und Lasttest
- Unterstützt herkömmliche Testtechniken
  - Funktionstest
  - Ablauftest
  - Strukturtest
- OCL ermöglicht präzise Spezifikationen

Literatur und weitere Information

- Bücher zum objektorientierten Test
  - Robert Binder: Testing Object-Oriented Systems. Addison Wesley, Upper Saddle River, 1999
  - John McGregor, David Sykes: A Practical Guide to Testing Object-Oriented Software. Addison Wesley, Upper Saddle River, 2001
  - Noack, J.: Techniken der objektorientierten Softwareentwicklung, Springer, Berlin, 2001
  - Harry Sneed und Mario Winter: Testen objektorientierter Software – Das Praxishandbuch. Hanser, München, 2001
- Seminare
  - Plenum Institut: `http://pin.plenum.de/pIN_Floater.htm`
  - GI-DIA: `http://www.dia-bonn.de/`
- Organisationen
  - GI-FG 2.1.7 TAV AK Test objektorientierter Programme (TOOP)
    `http://www.informatik.fernuni-hagen.de/import/pi3/GI`
    - 320+ Annotierte Referenzen zu OOT
    - Fragebogen OO-Testwerkzeuge (OO-CAST)
    - OO-Testbarkeit
    - OO-Reviews