Testen mit Hilfe von UML

Dehla Sokenou
TU Berlin
Fakultät Elektrotechnik und Informatik
Fachgebiet Softwaretechnik
dsokenou@cs.tu-berlin.de

17.9.2001

Übersicht

- Einleitung
- UML und Testen
  - Klassendiagramm
  - Use Cases
  - Statecharts
  - Sequenzdiagramme
  - Weitere UML-Diagramme
- Zusammenfassung und Ausblick

Einleitung

- Warum Testen mit Hilfe von UML?
  - UML verbreitete Spezifikationssprache in der Objektorientierung
  - Früher Beginn der Testaktivitäten
  - Nutzung spezifikationsbasierter Testfälle zur Aufdeckung von Fehlern zwischen Modellierung und Implementierung

Einleitung

- Probleme der UML
  - UML-Modelle oft unvollständig
  - Mangelnde Konsistenz der Modelle
  - Informationen für Testaktivitäten fehlen
  - Integrierter Ansatz für Testfallerzeugung aus UML-Spezifikationen fehlt
- Aber: UML bietet viele Möglichkeiten der Erweiterung für Tests, z. B. OCL, Profile

UML-Diagramme

- Verschiedene Sichten auf ein Problem
  - Klassendiagramm
  - Use Cases
  - Statecharts
  - Sequenz-diagramme
  - Kollaborations-diagramme
  - Objektdiagramme
  - Aktivitäts-diagramme
  - Komponenten-diagramme
  - Deployment-diagramme

UML und Testen

- Unterstützung verschiedener Testphasen
  - Klassentest
  - Integrationstest
  - Systemtest
- Unterstützung von Testaktivitäten
  - Testfallgenerierung
  - Testdurchführung
  - Testauswertung

UML und Testen

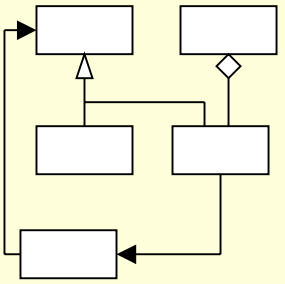
- Statische Sicht
  - Nutzbar insbesondere für Testvorbereitung und Testdurchführung
- Dynamische Sicht
  - Nutzbar insbesondere für Testfall-generierung und Testauswertung

Wichtige UML-Diagramme

- Konzentration auf wichtigste Diagramme:
  - Klassendiagramm
  - Use Cases
  - Statecharts
  - Sequenzdiagramme
- Andere Diagramme nur selten genutzt

Klassendiagramm

- Nutzbar insbesondere für Testvorbereitung
- Informationen über Klassen und Beziehungen
- Nutzung zur Testreihen-folgeermittlung

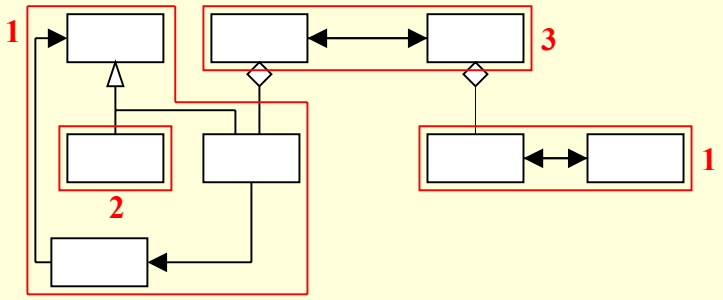


Klassendiagramm

- Testreihenfolgeermittlung nach Kung et al [Kung+1997]
  - Testreihenfolgeermittlung aus Code durch Rekonstruktion des Modells
  - Anwendbar auch ohne Implementierung
  - Berechnung der unabhängig testbaren Cluster (major test order)
  - Berechnung der Testreihenfolge innerhalb der Cluster (minor test order)

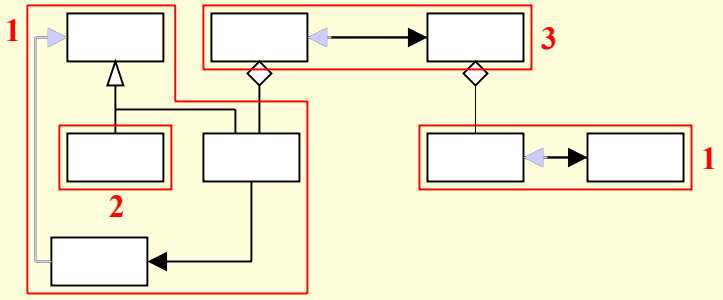
Klassendiagramm

- Technik von Kung et al - Beispiel
  - Bildung der Cluster



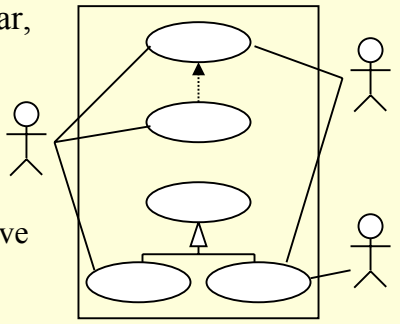
Klassendiagramm

- Technik von Kung et al - Beispiel
  - Aufbrechen der Zyklen



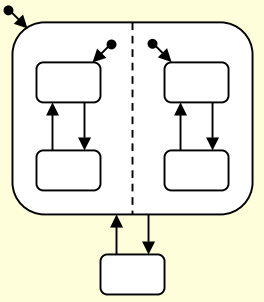
Use Cases

- Durch Erweiterung nutzbar, z. B. nach Binder [Binder1996]
  - Erweiterung durch EventTraces
  - Erweiterung durch relative Häufigkeit



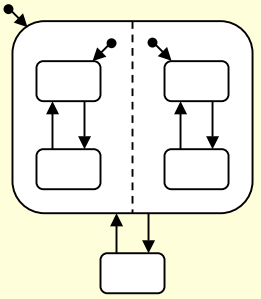
Statecharts

- Formalste UML-Diagrammart
- Beschreibung des Verhaltens einer Klasse
- Wichtigstes Diagramm für den Klassentest



Statecharts

- Viele aufeinander aufbauende Ansätze zur Generierung von Testfällen
- Kontrollfluß- und datenflußbasierte Techniken sowie Techniken aus der formalen Verifikation



Statecharts

- Kontrollflußbasierte Techniken
  - Analog zu traditionellen kontrollfluß-orientierten Techniken
  - Basierend auf Technik von Chow [Chow1978]

Statecharts

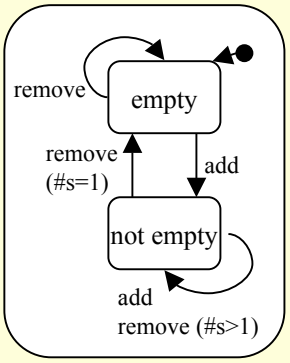
- Kontrollflußbasierte Technik analog zu traditionellen kontrollfluß-orientierten Techniken
  - Testfälle zur möglichst großen Überdeckung der Statecharts
  - Unterscheidung von Zustandsüberdeckung (Knoten), Transistionsüberdeckung (Kanten) und Pfadüberdeckung

Statecharts

- Kontrollflußbasierte Techniken basierend auf Technik von Chow
  - Ursprünglich nur flache Automaten
  - Zustands- und Transitionsüberdeckung
  - Keine Pfadüberdeckung
  - Jeder Zyklus einmal ausgeführt
  - Einfaches terminierendes Verfahren

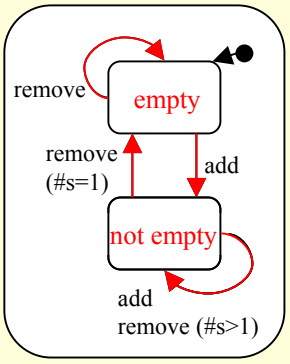
Statecharts

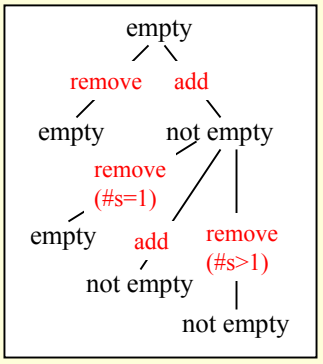
- Technik von Chow - Beispiel



Statecharts

- Technik von Chow - Beispiel



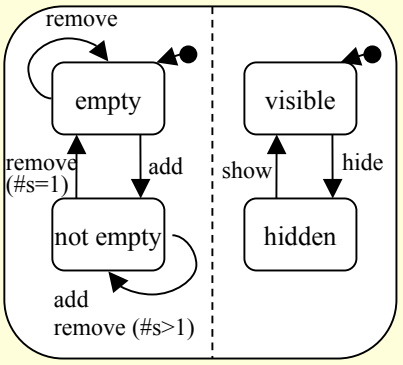


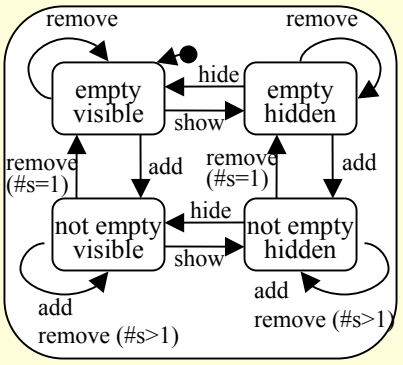
Statecharts

- Hierarchie und Nebenläufigkeit
  - Technik von Kim et al [Kim+1995] zur Elimination von Hierarchie und Nebenläufigkeit
  - Technik von Kung et al [Kung+1994] zur Berechnung von Testfällen ohne Elimination von Hierarchie
  - Beide Ansätze aufbauend auf Technik von Chow

Statecharts

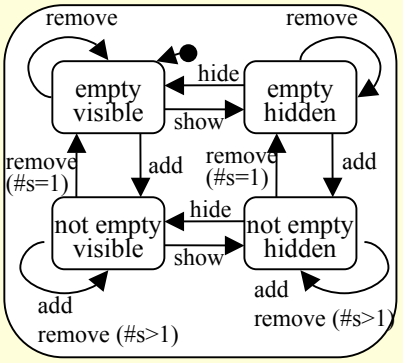
- Hierarchie und Nebenläufigkeit - Beispiel

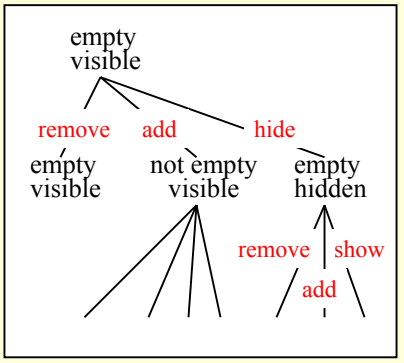




Statecharts

- Hierarchie und Nebenläufigkeit - Beispiel



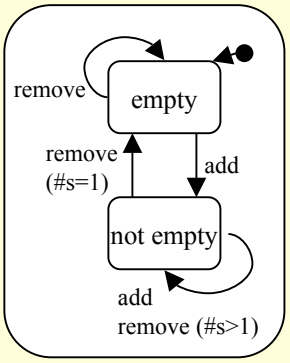


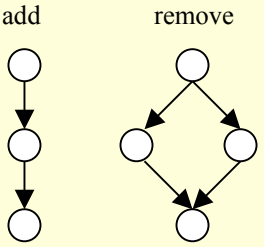
Statecharts

- Integration WhiteBox-Technik
  - Technik zur Kombination von zustands-basierten und kontrollflußorientierten Techniken [Binder1996]
  - Erweiterung von Zustandsautomaten um Kontrollflußgraphen der entsprechenden Methoden
  - Generierung der Testfälle z. B. nach Chow

Statecharts

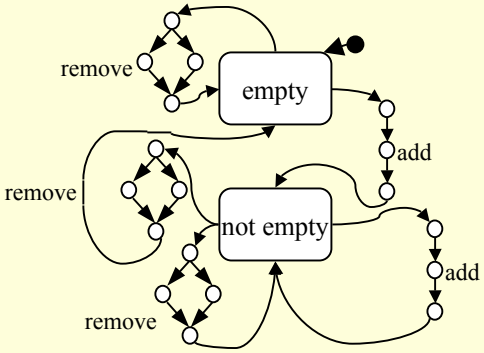
- Integration WhiteBox-Technik - Beispiel

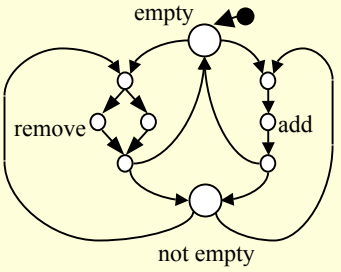




Statecharts

- Integration WhiteBox-Technik - Beispiel



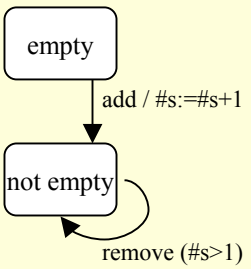


Statecharts

- Datenflußbasierte Technik [Kim+1995]
  - Nutzung der Aktionen zur Datenfluß-analyse
  - Definition und Benutzung von Variablen
  - Definition und Benutzung von Zuständen
  - Generierung von Testfällen durch Defs/Uses-Kriterien

Statecharts

- Datenflußbasierte Technik - Beispiel
  - Zustand 'empty' definiert Zustand 'not empty'
  - Zustand 'not empty' benutzt Zustand 'empty'
  - Transistion 'add' definiert Variable '#s'
  - Transition 'remove' benutzt Variable '#s'

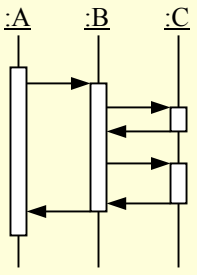


Statecharts

- Modelchecking
  - Verifikationstechnik, die eine Eigenschaft quasi vollständig testet
  - Nutzung der durch Modelchecking erzeugten Pfade als Testfälle zum Testen in Bezug auf eine Eigenschaft [Engels+1997]
  - Nutzung der durch Modelchecking erzeugten Gegenbeispiele als Testfälle

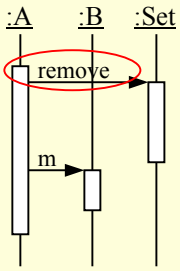
Sequenzdiagramme

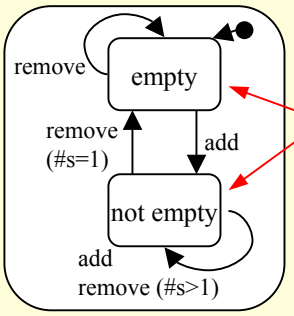
- Nur wenig Informa-tionen für den Test
- Beschreibung des Systemverhaltens nur unvollständig und ausschnittsweise
- Jedes Sequenzdiagramm ein Testfall



Sequenzdiagramme + Statecharts

- Sequenzdiagramme in Kombination mit Statecharts nutzbar für Integrationstests





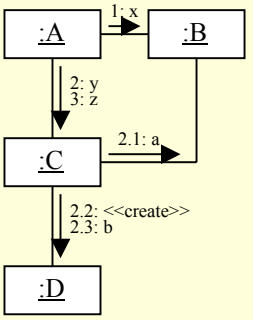
2 mögliche Ausgangszustände = 2 Testfälle

Weitere Diagramme

- Weitere UML-Diagramme nur wenig genutzt
- Weitere UML-Diagramme in Bezug auf Testen wenig erforscht
- Kurzer Überblick der möglichen Nutzung
  - Kollaborationsdiagramme
  - Objektdiagramme
  - Deploymentdiagramme

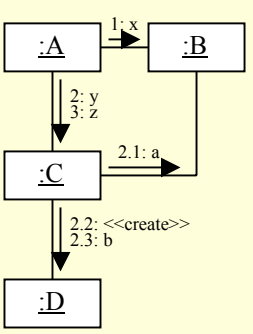
Kollaborationsdiagramme

- Bieten ähnliche Informationen wie Sequenzdiagramme
- Darstellung von Iteration und Nebenläufigkeit möglich
- Jedes Kollaborations-diagramm ein oder mehrere Testfälle



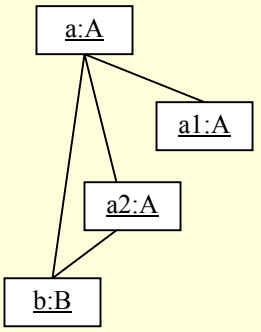
Kollaborationsdiagramme

- Kombination mit Statecharts analog zu Sequenzdiagrammen möglich
- Objektbeziehungen nutzbar zur Testreihen-folgeermittlung
- Verbindung mit Klassen-diagramm möglich



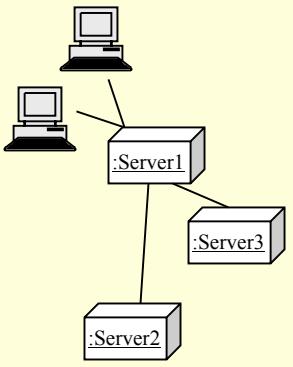
Objektdiagramme

- Modellierung statischer Momentaufnahme des Systems
- Nutzung zum Aufbau von Objektstrukturen zum Integrations- und Systemtest



Deploymentdiagramme

- Modellierung insbesondere von verteilten Systemen
- Nutzbar zur Vorbereitung des Systemtests



Zusammenfassung

- Bisher einzelne Techniken für jedes Diagramm
- Statecharts am besten untersuchte Art der Spezifikation, nutzbar insbesondere für Klassentest
- Techniken zum Teil mit implementations-basierten Techniken kombinierbar
- Techniken zum Teil bereits automatisiert als Tools verfügbar

Zusammenfassung

- Integrierte Ansätze für
  - Statecharts und Sequenzdiagramme
  - Klassendiagramm und Kollaborations-diagramme
- Integrierter Ansatz zur Nutzung aller vorhandenen UML-Diagramme fehlt

Ausblick

- Weitere vorhandene Techniken auf Anwendbarkeit und Automatisierung überprüfen
- Integrierte Techniken entwickeln bzw. erweitern
  - Berücksichtigung aller Modelle zur Testunterstützung
  - Gewinnung von Testfällen aus einem Modell, Test gegen alle Modelle

Ausblick

- Möglichkeiten der Erweiterung von UML für den Test untersuchen
  - Laufende Arbeit zur formalen Erweiterung mit Hilfe von OCL und zur Erweiterung mit Hilfe von Stereotypen und Profilen
- Entwicklung eines Frameworks zur Unterstützung des Tests anhand von UML-Spezifikationen
- Integration in methodischen Ansatz

Literatur

- **[Binder1996]** - R. Binder. The Free Approach to Testing Object-Oriented Software: An Overview, www.rbsc.com/pages/FREE.html, 1996.
- **[Chow1978]** - T. Chow. Testing Software Design Modeled by Finite-State Machines, IEEE Transactions on Software Engeneering SE-4(3), 1978.
- **[Engels+1997]** - A. Engels, L. Feijs, S. Mauw. Test Generation for Intelligent Networks Using Model Checking, LNCS 1217, 1997.
- **[Kim+1995]** - Y. Kim, H. Hong, S. Cho, D. Bae, S. Cha. Test Cases Generation from UML State Diagrams, IEEE Proceedings Software, v. 146, 1995.
- **[Kung+1994]** - D. Kung, J. Gao, P. Hsia, Y. Toyoshima, C. Chen. On Object State Testing, COMPSAC'94, 1994.
- **[Kung+1997]** - D. Kung, N. Suchak, J. Gao, P. Hsia, Y. Toyoshima, C. Chen. Constructing Functional Test Cases Using Incrementally Derived State Machines, 11th ICTCS, 1994.