Modellbildung in der Entwicklung mit Schwerpunkt Architekturen

Modelle und Qualitätssicherung

Mitverfasser der Folien: Stefan Wagner

Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München
SS 2007

Qualitätssicherung (QS)

- Konstruktiv
  - Prozesse
  - Methoden
  - Werkzeuge
  - Standards
- Analytisch
  - Bewertung
  - Test
  - Verifikation

Qualitätssicherung und Modelle

- Qualitätssicherung von Modellen
  - Richtlinien
  - Reviews
  - Simulation
  - Model-Checking
  - Theorem-Beweisen
  - Test von Modellen
  - Modell-Metriken
- Qualitätssicherung durch Modelle
  - Spezifikationsreview
  - Code-Generierung
  - Modell-basiertes Testen
  - Model-Checking, Verifikation
  - Zuverlässigkeitsmodelle

Qualitätssicherung von Modellen

Konstruktiv: Richtlinien

- Empfehlungen zur „guten“ Modellierung
- Vermeidung von Problemen bestimmter Modellierungssprachen (fehlende Semantik, komplizierte Semantik wie in Semantik in Stateflow oder Statecharts)
- Eingrenzung der Modellierungssprache für bestimmte Zwecke (Code-Generierung, sicherheitskritische Anwendungen)
- Unternehmensstandards zur leichteren Lesbarkeit (Farben, Annotationen)

Beispiele für Richtlinien

- Scott W. Ambler: The Elements of UML 2.0 Style
- MAAB-Richtlinie für Matlab/Simulink (Automotive)
- MISRA für Matlab/Simulink (sicherheitskritisch, Vorversion verfügbar)
- ESA/ESTEC-Richtlinien für VHDL

Modell-Reviews/-Inspektionen

- Analog zu Code-Reviews
- Wahrscheinlich wichtigstes Mittel zur Validierung
- Wiederum Nutzung von Richtlinien und Checklisten
- Grady (1992): 50-75% der Designfehler können durch Modell-Reviews gefunden werden

Simulation

- Großer Vorteil ausführbarer Modellierungstechniken
- Dadurch wird das Modell zu einem abstrakten Prototypen
- Bei graphischer Modellierung können Abläufe und Zusammenhänge anschaulich verfolgt werden
- Generierung von MSCs

Model Checking (Modellprüfung)

- Vollautomatisch
- Nachweis von Eigenschaften, die in Temporallogik formuliert sind
- Gegenbeispiels, falls geforderte Eigenschaft nicht erfüllt wird
- Über „brute-force“-Aufzählung
- Problem: Zustandsexplosion
- Funktioniert nur bei starken Abstraktionen
- Bekannte Werkzeuge: SMV, SPIN

Interaktive Verifikation

- Semiautomatischer Nachweis von Modelleigenschaften
- Nutzung von Theorembeweisern (bspw. Isabelle, Nuprl)
- Darstellung des Modells und der Eigenschaften in einer höherwertigen Logik
- Konstruktion von Beweisausdrücken
- oder auch P |= E

$$\text{Programm}(P) \Rightarrow \text{Eigenschaft}(E)$$

Test von Modellen

- Ähnlich wie Code können Modelle (falls ausführbar) auch getestet werden
- Arten
  - Black-Box: ohne Wissen über die Modellstruktur
  - Glass-Box: Überdeckung bestimmter Modellstrukturen (Zustandsüberdeckung, Transitionsüberdeckung, …)
- SIL-Test (Software in the loop)
- „Strukturierte Simulation“
- Siehe auch Ghosh, France, Braganza, Kawane (2003)

Modell-Metriken

- Analog zu Code-Metriken
- Komplexitätsmetriken
  - Zahl der Schnittstellen zu anderen Komponenten
  - Zahl der Pfade durch Statechart
  - Zahl der Zustände
- Obergrenzen, Voraussage fehleranfälliger Teile
- Praktisch nur bedingt von Nutzen (Begründung oft schwierig)
- Siehe auch
  - Objekt-orientierte Metriken von Chidamber und Kemerer
  - Wagner und Jürjens (2005)

Qualitätssicherung durch Modelle

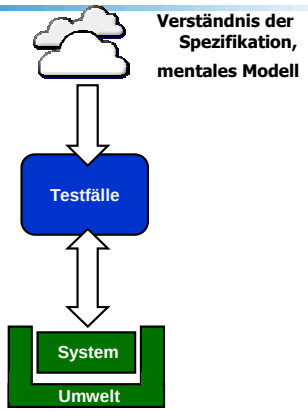
Modellierung

- Modellierung selbst ist konstruktive Qualitätssicherung
- „Spezifikationsreview“
- Präzisierung mentaler und textueller Spezifikationen
- Aufdeckung von Widersprüchen und Auslassungen
- Sehr effektiv (vgl. Pretschner et al. (2005))

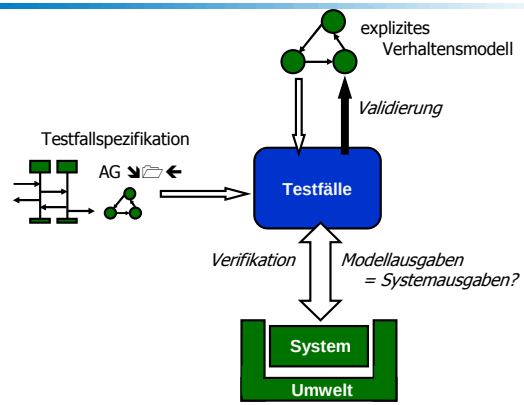
Correct-by-construction

- Konstruktive QS: Code-Generierung
- Korrektes Modell und Code-Generator werden vorausgesetzt
- Direkte Generierung von Programmcode aus dem Modell
- -> Programm korrekt
- Beispiele: MathWorks Real-Time Workshop, dSpace TargetLink, IBM Rational Rose-RT
- Wird erschwert durch verschiedene Abstraktionsebenen, deren Überbrückung Fehler einführen kann

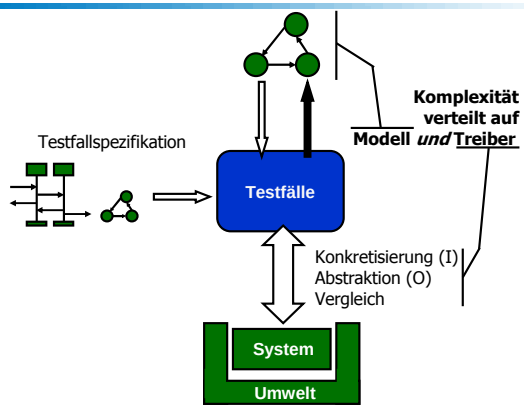
Modellbasiertes Testen



Vorgehen modellbasiertes Testen



Modellbasiertes Testen: Abstraktionsniveaus



Bewertung modellbasiertes Testen

- Vorteile
  - Effektiv, findet teilweise andere Fehler als manuelle Tests
  - Automatische Generierung großer Testsuiten
  - Schnelle Neugenerierung bei Änderungen an Spezifikation und Modell
  - Explizite Modellierung schärft die Spezifikation
- Nachteile
  - Modellierung aufwändig
  - Es kann nur getestet werden, was modelliert wurde
  - Auswahl der Testfälle: Was ist ein sinnvolles Vorgehen zur Generierung?

Model-Checking, Theorembeweisen

- Formale Analysen des Modells
- Auch Analyse des Programms, bei
  - automatischer Code-Generierung
  - Nachweis der Übereinstimmung zwischen Code und Modell
- Analyse der Modelle einfacher als Code-Analyse, da die Modelle abstrakter sind

Zuverlässigkeitsmodelle

- Ein stochastisches Modell des Fehlverhaltens
- Dient zur
  - Abschätzung der aktuellen Zuverlässigkeit
  - Planung der weiteren Qualitätssicherung
- Klassische Fragestellung: „When to stop testing?“
- Übliches Vorgehen
  - Sammeln von Fehlerdaten im Test
  - Anpassung eines stochastischen Modells
  - Voraussagen für weiteren Test und Feld
- Voraussagen für Feld meist sehr ungenau
- Problem: Test muss Nutzung wiederspiegeln
- Kann auch mit Kosten/Nutzen-Betrachtungen verbunden werden

Literatur

- Broy, Jonsson, Katoen, Leucker, Pretschner: Model-Based Testing of Reactive Systems. LNCS 3472, Springer-Verlag, 2005.
- Clarke, Grumberg, Peled. *Model Checking*. MIT Press, 2000.
- Pretschner, Prenninger, Wagner, Kühnel, Baumgartner, Sostawa, Zölch, Stauner: One Evaluation of Model-Based Testing and its Automation. Proc. 27th International Conference on Software Engineering (ICSE'05). ACM Press, 2005.
- Wagner, Jürjens: Model-Based Identification of Fault-Prone Components. Proc. 5th European Conference on Dependable Computing (EDCC-5). LNCS 3463, Springer-Verlag, 2005.
- Scott W. Ambler. The Elements of UML 2.0 Style. Cambridge University Press, 2005.
- MathWorks Automotive Advisory Board. Controller Style Guidelines For Production Intent Using Matlab, Simulink and Stateflow, 2001.
- Ghosh, France, Braganza, Kawane. Test Adequacy Assessment for UML Design Model Testing. Proc. 14th International Symposium on Software Reliability Engineering (ISSRE'03). IEEE CS Press, 2003.

Literatur II

- Nipkow, Paulson, Wenzel. Isabelle/HOL. A Proof Assistant for Higher-Order Logic. LNCS 2283, Springer-Verlag, 2005.
- Musa. Software Reliability Engineering. 2nd ed., AuthorHouse, 2004.
- Wagner. A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques. Proc. International Symposium on Software Testing and Analysis (ISSTA'06). ACM Press, 2006.