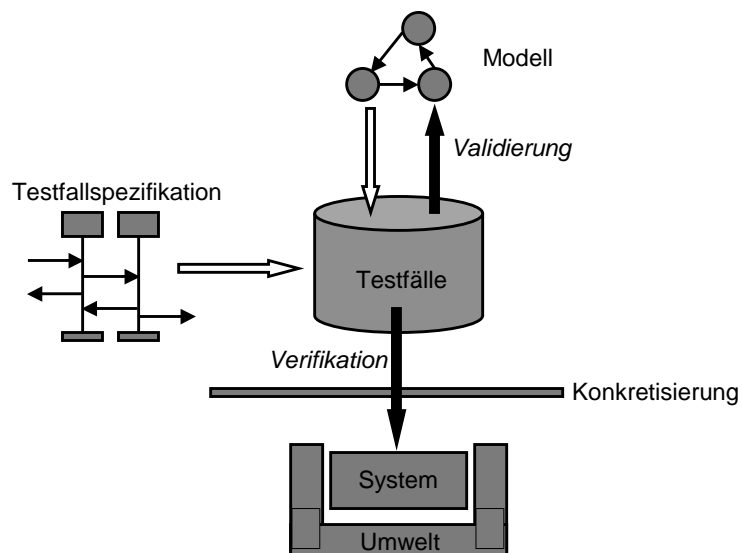


Modellbasiertes Testen

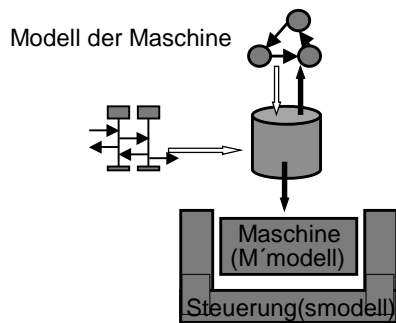
Alexander Pretschner
 Technische Universität München

Modellbasiertes Testen



Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?

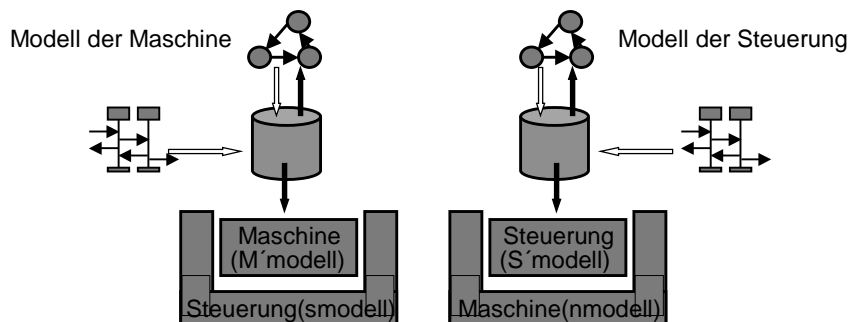


GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 3

Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?

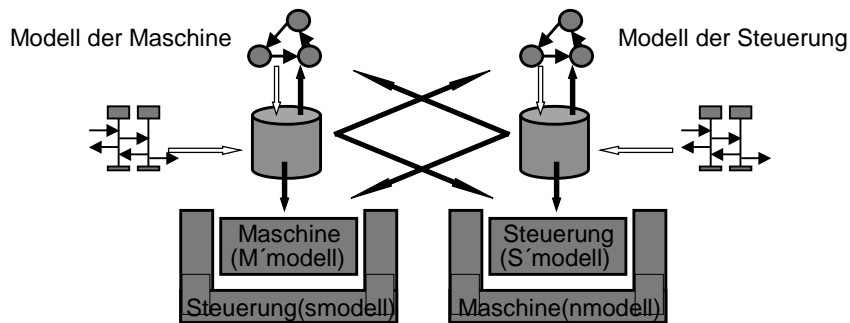


GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 4

Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?

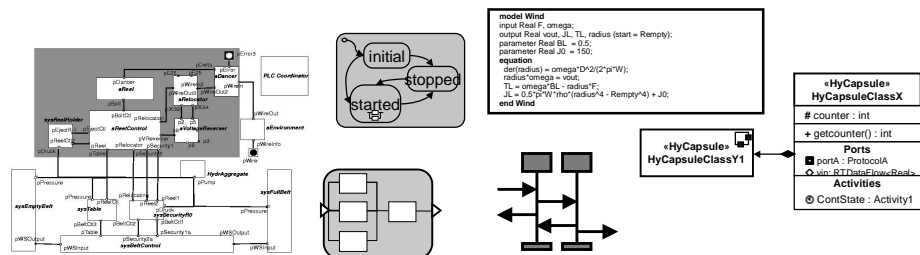


GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 5

Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?



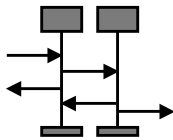
GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 6

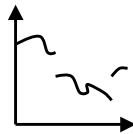
Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?

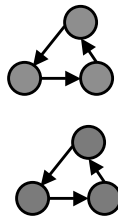
funktional



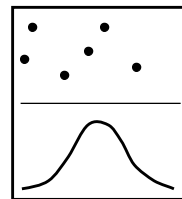
Ad-hoc



strukturell

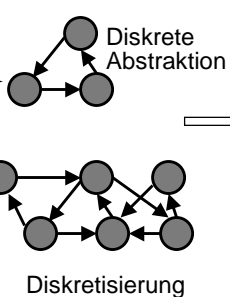
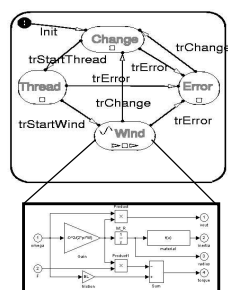


stochastisch

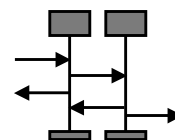


Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?

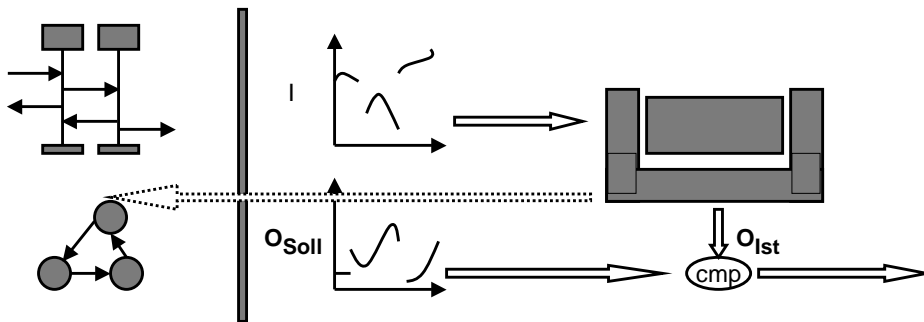


Testfall



Fragestellungen

- Was wird modelliert?
- Wie wird es modelliert?
- Was sind gute Testfallspezifikationen?
- Wie werden daraus Testfälle erzeugt?
- Wie werden Abstraktionsniveaus abgeglichen?

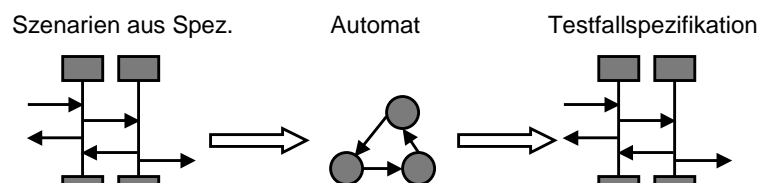


GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 9

Testfallspezifikationen: Beschreibung

- Informelle Sequenzdiagramme aus Spezifikation erster Schritt
 - Aber vermutlich sind diese Szenarien ohnehin richtig implementiert
 - Interessant: (il-)legale Permutationen, Ausbleiben von Signalen, verspätetes Eintreffen
 - Also eine *Menge* von Szenarien
- Funktionale Testfallspezifikationen als (nichtdeterministische) Zustandsmaschinen, auf denen Coverage definiert wird



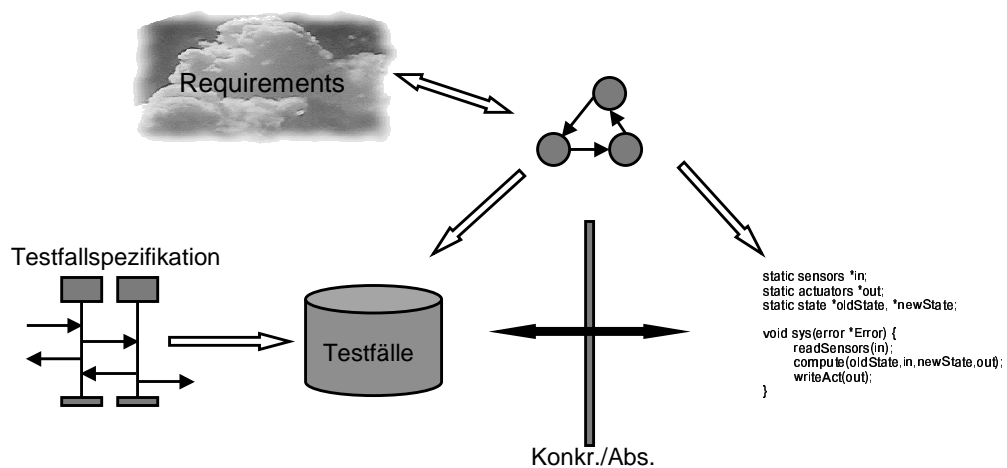
GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 10

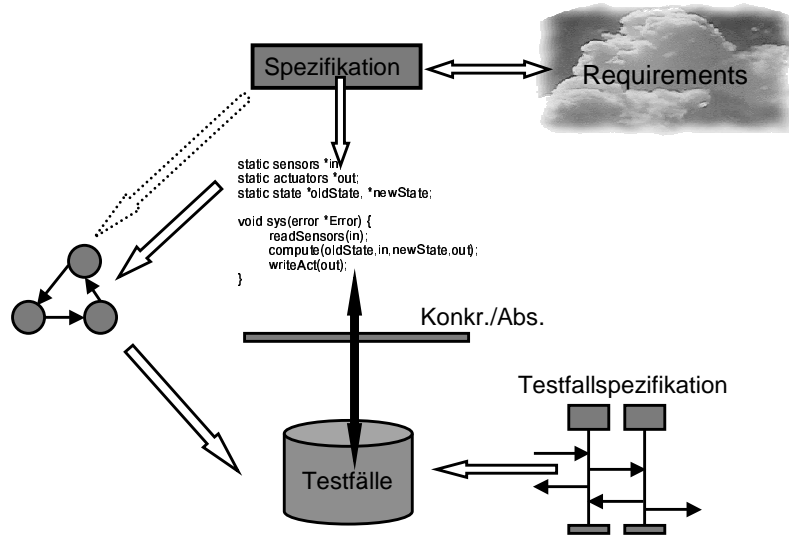
Überblick

- Szenarien modellbasierten Testens
- Technologie
- Diskrete und kontinuierliche Systeme
- Zusammenfassung

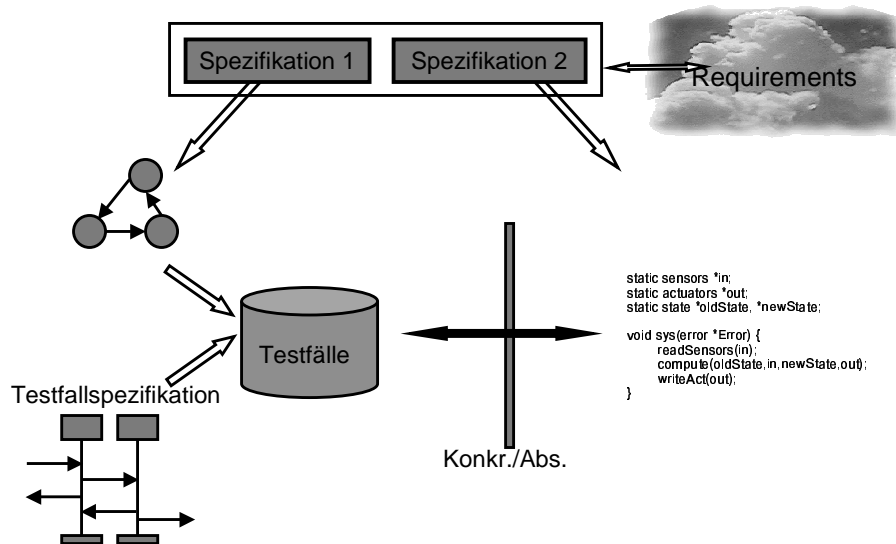
Code aus Modell



Modell aus bestehendem System



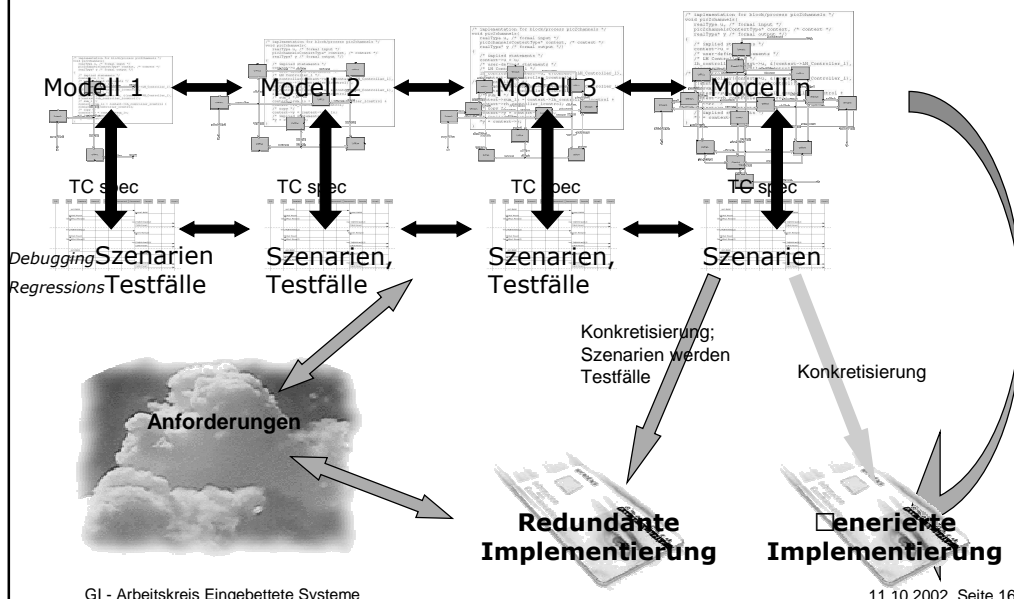
Simultane Entwicklung Code und Modell



Zusammenfassung Szenarien

- Code und Testfälle aus Modell
 - Plus: Test von Generatoren und Compilern
 - Minus: Keine Redundanz
- Modell aus bestehendem System
 - Plus: kein „moving target“ mehr
 - Minus: Eventuell zu spät
- Simultane, unabhängige Entwicklung System und Modell
 - Plus: vollautomatische Verdichtbildung
 - Minus: Abgleich iterativer Prozesse

Modellbasiertes Testen: Inkrementalität



Prinzip Testfallerzeugung

Rekursive Übersetzung von Systemen in Prädikate

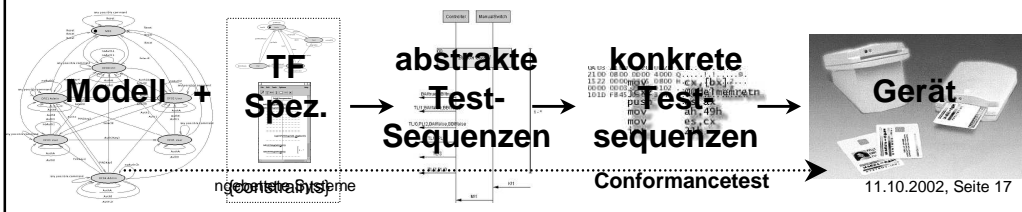
Plus TF-Spez. als Automat/MSCs/Constraints

- Symbolische Ausführung; Rechnen mit Mengen
- Ungebundene Variablen werden bei Ausführung (partiell) gebunden

Plus Umgebungs- und Effizienzconstraints

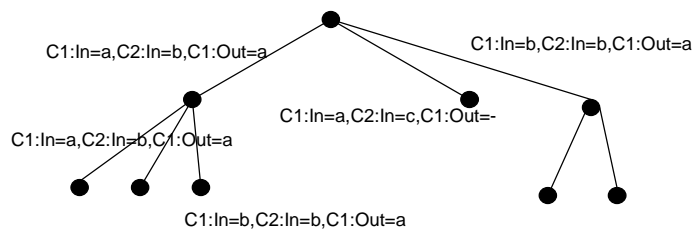
- Interaktion für graceful degradation

Instantiierung und Konkretisierung der Testfälle

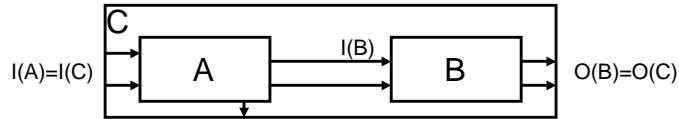


TF-Generierung als Suchproblem

- Testfälle aus TF-Spezifikationen (aus Testzwecken)?
 - Strukturell, funktional, stochastisch
- Abdeckung des Modells
 - funktionaler Test; strukturell auf Codeebene
- Aufzählen (einer Menge) von I/O-Traces (beschränkte Tiefe) mit verschiedenen Such- und Speicherstrategien



Kompositionale Testfallerzeugung



Idee: Benutze existierende Sequenzen als Einschränkung des Suchraums

Erzeuge TF für A, B (units), C: $\{TA\}$, $\{TB\}$, $\{TC\}$

Verwende $\{TA\}$ -Ausgaben als (putative) B-Eingaben $\Rightarrow \{TB2\}$

- Nicht immer möglich (z.B. B erfordert bestimmtes Timing)

Verwende $\{TB\}$ -Eingaben als (putative) A-Ausgaben $\Rightarrow \{TA2\}$

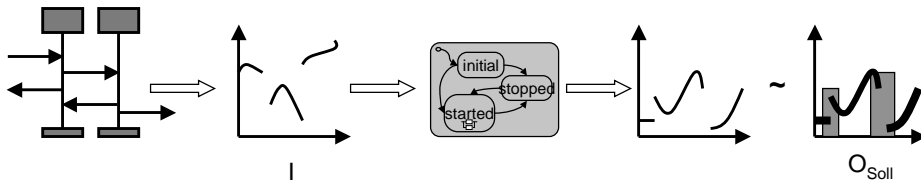
Dann: $I(\{TA\})$, $I(\{TA2\})$, $O(\{TB\})$, $O(\{TB2\}) \Rightarrow \{TC2\}$

GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 19

Testfallgenerierung für hybride open-loop-Systeme

- Bei grober diskreter Abstraktion: Konkretisierung ad hoc
 - Ausgaben können i.a. nicht rekonstruiert werden
 - Also in nicht-abstrahiertes Modell füttern
- Wert- und zeitbezogener Nichtdeterminismus
- Bei Diskretisierung: Modelle zu genau
 - Abweichungen von „Schläuche um“ Sollverhalten zulassen



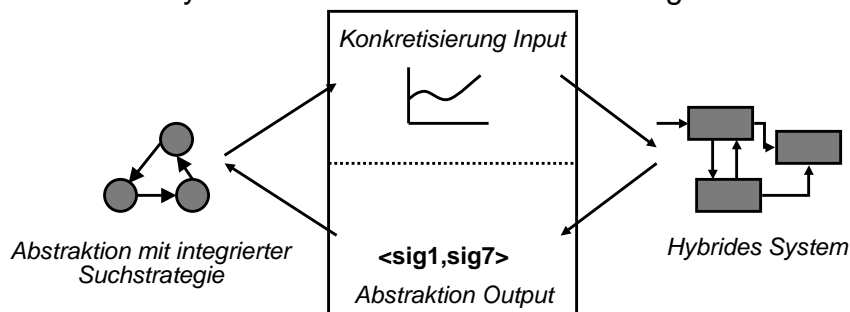
$R(O_{Ist}, O_{Soll})?$

GI - Arbeitskreis Eingebettete Systeme

11.10.2002, Seite 20

Closed-loop

- Test von Regler und/oder Strecke
- Problem: diskret abstrahierter Teil muß direkt mit kontinuierlichem System interagieren
- Idee: diskrete Abstraktion der qualitativen Zustände des Gesamtsystems und on-line Konkretisierung/Abstraktion



Keine eierlegende Wollmilchsau - Klärungsbedarf

- Wie wird modellbasiertes Testen eingesetzt?
- Wie präzise ist das Modell; wie können Abstraktion und Konkretisierung implementiert werden?
- Welche Eigenschaften sind interessant, wie erfaßt man sie systematisch?
- Lohnt der Overhead?

- Wie kann der Suchraum sinnvoll eingeschränkt werden?
- Sprache für (nicht-funktionale) Testfallspezifikationen?

Zusammenfassung

- Systematische modellbasierte Testfallgenerierung machbar
- Domänen-, prozeß- und kundenspezifische Ausrichtung
- Zentrales Problem nicht V&V-Technologie, sondern Festlegen der zu überprüfenden Eigenschaften
 - Modellierung als solche identifiziert viele Probleme
 - Automatisierte Testfallgenerierung: Refactoring, Codegeneratoren
 - Fehlerklassen, Testmuster
- Durchsetzung
 - primär politisches Problem
 - „Wenn Modell, dann nicht nur Testfälle, sondern auch Code“ – Notwendigkeit der Redundanz verkannt