Modellbasierte Entwicklung am Beispiel von AutoFocus

Modellbildung in der Entwicklung

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

SS 2007

Inhalte

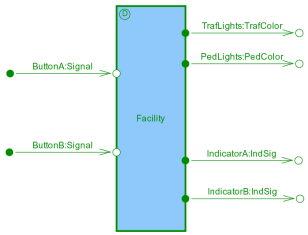
Vorlesungsinhalte:

1. Modellbegriff
2. Grundlagen
3. Anwendung
4. Modellgetriebene Entwicklung
   1. Systemmodell
   2. Sichten
   3. Qualitätssicherung
5. Qualitätsmodelle
6. Domänenspezifische Sprachen

Systemmodell: **Komponente und Schnittstelle**

Komponente:
Verhaltenskapsel

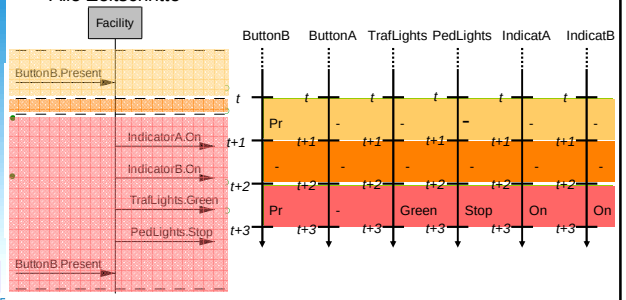
Schnittstelle:
Nachrichtenfluss
– Richtung
– Nachrichtentyp



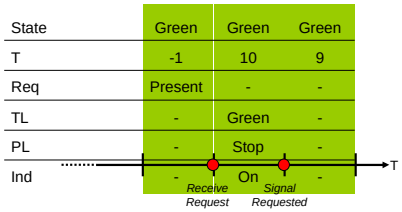
Systemmodell: **Ablauf**

Ablauf: Vollständige Signalgeschichte

- Alle Ports
- Alle Zeitschritte



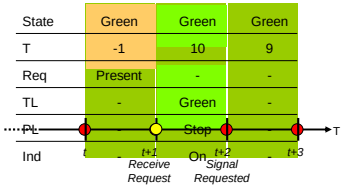
Systemmodell: **Zustand**



Zustand: Beobachtung zwischen zwei Übergängen

- Ports: aktuelle Nachrichten
- Variablen: aktuelle Werte

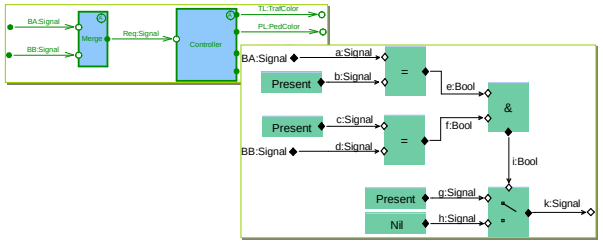
Systemmodell: **Berechnungsschritt**



Berechnungsschritt: Lesen Vorzustand/Schreiben Nachzustand

- Vorzustand: Kontrollzustand, Datenzustand, Eingabe
- Nachzustand: Ausgabe, Datenzustand, Kontrollzustand

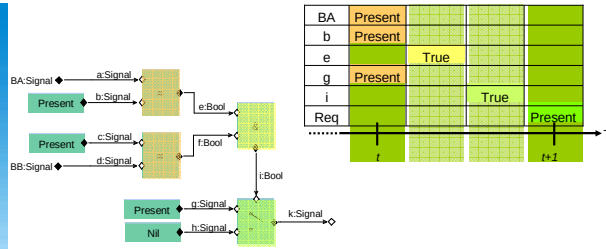
Systemmodell: **Datenfluss**



Funktionsblockdiagramm: Datenflussorientierte Verhaltensbeschreibung

- Signal: Eingabe (z.B. BA), Ausgabe (z.B., Req), interne Signale (z.B. e, f)
- Funktion: Signal-Berechung
- Datenfluss: Schrittweise Berechnung aller Signale

Systemmodell: **Berechnungsfluss**



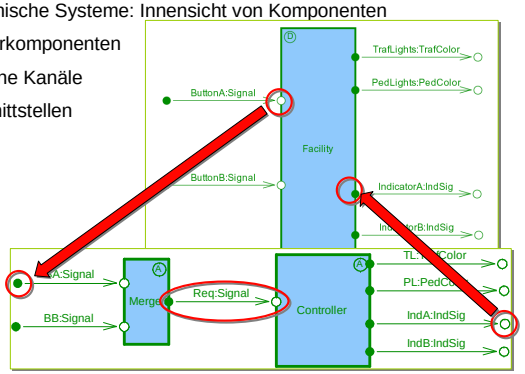
Berechnungsfluss: Rundenweise Berechnung Ausgabe zu Eingabe

- Rundenstart: Lesen Eingabe Berechnung
- Zwischenschritte: Iterative Bestimmung Zwischenberechnungen
- Rundenende: Schreiben Ausgabe Berechnung

Systemmodell: **Hierarchie**

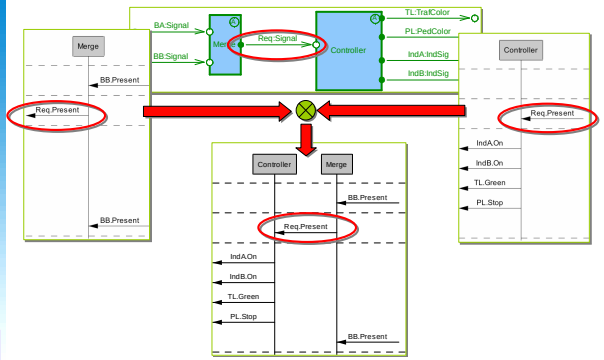
Hierarchische Systeme: Innensicht von Komponenten

- Unterkomponenten
- Interne Kanäle
- Schnittstellen



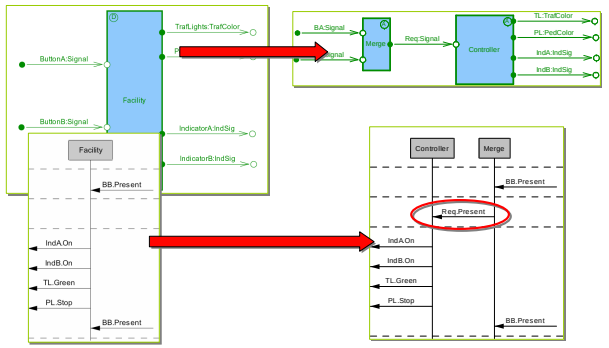
Systemmodell: **Komposition**

Hierarchische Systeme: Komposition mittels Kommunikation

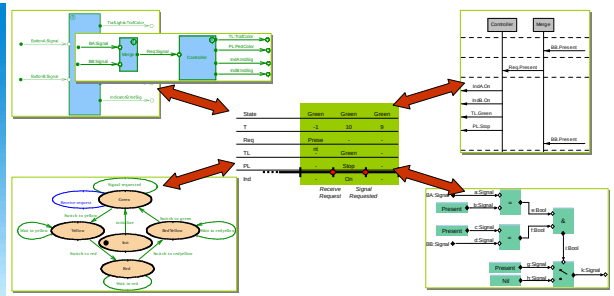


Systemmodell: **Verfeinerung**

Hierarchische System: Verfeinerung durch interne Kommunikation



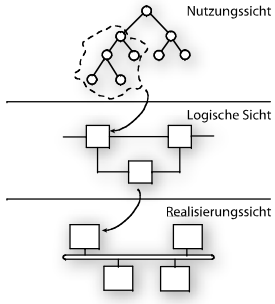
Systemmodell: **Integration**



Integration aller Sichten und Beschreibungstechniken im Systemmodell erlaubt

- Vergleich von Sichten
- Transformation zwischen Sichten

Sichten im Entwicklungprozess

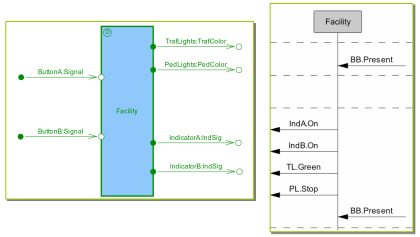


Nutzungssicht: Funktionsarchitektur
- Szenarien (Reaktive Funktionen)
- Datenflüsse (Steuerungsfunktionen)

Logische Sicht: Anwendungsarchitektur
- Komponentenstrukturen
- Zustandsmaschinen
- Datenflüsse

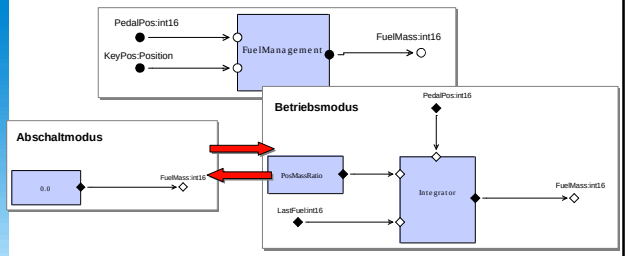
Realisierungssicht: SW/HW-Implementierungsarchitektur
- Softwarestrukturen
- Hardwarestrukturen

Nutzungssicht: Reaktive Funktionen



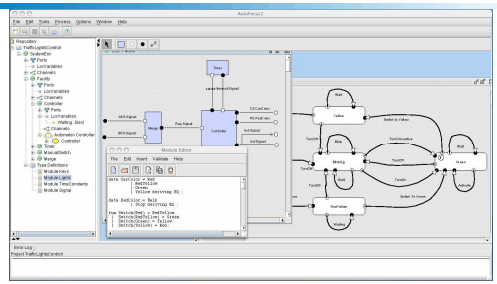
- Ziel: Strukturierte Beschreibung der reaktiven Funktionalität aus Systemnutzungssicht
- Konzepte: Ein-/Ausgabesignale, Funktionshierarchie
- Notationen: Datenbeschreibungen, Schnittstellenbeschreibung, Szenarienbeschreibungen

Nutzungssicht: Steuerungsfunktionen



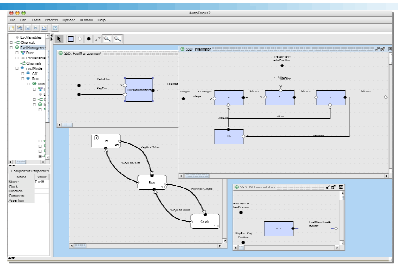
- Ziel: Strukturierte Beschreibung der Steuerungsfunktionalität aus Systemnutzungssicht
- Konzepte: Ein-/Ausgabesignal, Funktionshierarchie
- Notationen: Datenbeschreibungen, Schnittstellenbeschreibungen, Datenflussbeschreibungen

Logische Sicht: Reaktive Systeme



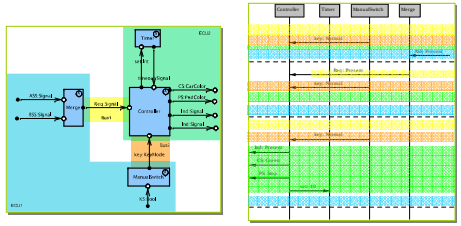
- Ziel: Strukturierte Beschreibung der logischen Struktur und des Verhaltens von reaktiven Systemen
- Konzepte: Komponentenhierarchie, Ein-/Ausgabeereignis, Zustand
- Notationen: Datenbeschreibungen, Komponentenbeschreibungen, Zustandsmaschinenbeschreibungen

Logische Sicht: Steuerungssysteme



- Ziel: Strukturierte Beschreibung der logischen Struktur und des Verhaltens von Steuerungssystemen
- Konzepte: Datenflusshierarchie, Ein-/Ausgabesignal, Modus
- Notationen: Datenbeschreibungen, Datenflussbeschreibungen, Modusbeschreibungen

Technische Sicht



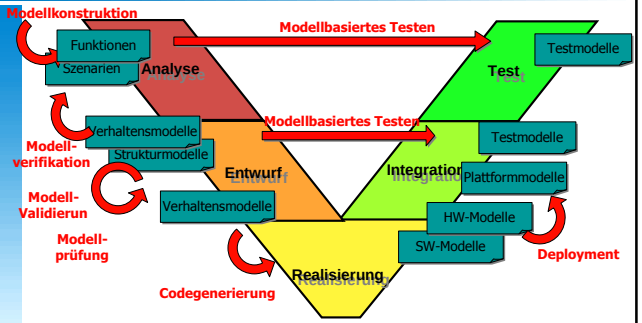
- Ziel: Strukturierte Beschreibung der technischen Realisierung des System
- Konzepte: Prozess, Nachricht, Steuergerät, Kommunikationskanal
- Notationen: SW-Beschreibungen, HW-Beschreibungen, Allokations-/Schedulingsbeschreibungen

Modellbasierung und Qualitätssicherung

Modellbasierung verbessert die Produktqualität durch

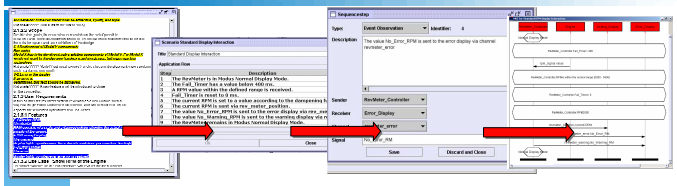
- Aufgabenorientierte Darstellung von Systemperspektiven
  - Zielgerichtete Abbildung von Anwender-/Entwicklerwissen
  - Gezielte Vorgaben für Abfolge von Entwicklungsprodukten
- Abstrahierte Sichten des Gesamtsystems
  - Verständliche/überprüfbare Darstellung Gesamt/Teilsystem
  - Konstruktive Vermeidung von fehlerhaften Produkten
- Umfassende Integration und Werkzeugtechnische Unterstützung
  - Sichtenkonsistenz und Sicherung von Systemeigenschaften
  - Weiterverwendung/Transformation von Produkten inkl. Codegenerierung

Beispiel: Modelle und QS-Techniken



Prozessintegrierte Qualitätsmaßnahmen durch definierte Produkte und konstruktive/analytische Verfahren

Modellkonstruktion

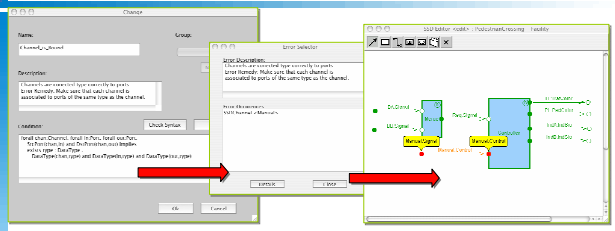


Strukturierung Klassifizierung Formalisierung

Modellkonstruktion: Konstruktive Qualitätssicherung

- Hohe Effektivität (vgl. Pretschner et al. (2005))
- Schritte: Strukturierung, Klassifizierung, Formalisierung
- Aspekte: Präzisierung
  - Sicherung Eindeutigkeit
  - Entdeckung Unvollständigkeit
  - Sicherung Prüfbarkeit

Modellprüfung



Definition Prüfung

Modellprüfung: Analytische Qualitätssicherung

- Schritte: Richtliniendefinition, -prüfung (Review/Inspektion)
- Aspekte: „Gute Modellierung“
  - Vermeidung von Modellierungsfehlern
  - Lesbarkeit, Wartbarkeit
- Grady: 50-75% der Designfehler durch Modell-Reviews entdeckt

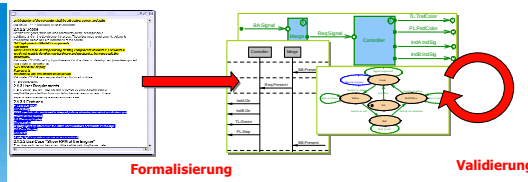
Modellprüfung: Prüfungsaspekte

- Beispiele Prüfungsaspekte
  - Vermeidung von Problemen bestimmter Modellierungssprachen (12 o‘clock-Semantik in Stateflow)
  - 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

Modellprüfung: 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)

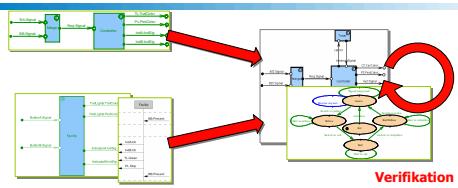
Modellvalidierung



Modellvalidierung: Analytische Qualitätssicherung

- Schritte: Modellkonstruktion, Modellsimulation, Ablaufgenerierung
- Aspekte: Validierung Angemessenheit
  - Ausführbare Spezifikation (Blackbox-/Glassboxsimulation)
  - Prototypenkonstruktion (In-the-Loop-Simulation)

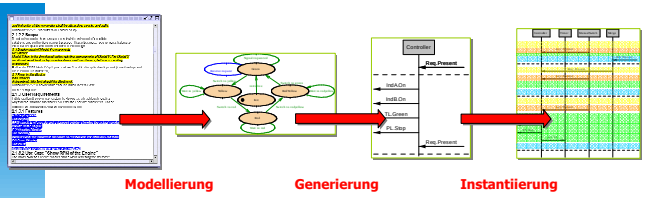
Modelltest



Modelltest: Analytische Qualitätssicherung

- Schritte: Modellkonstruktion, Modellverifikation
- Aspekte: Frühe Qualitätssicherung
  - Konsistenz deskriptive/konstruktive Modelle
  - Modellsimulation

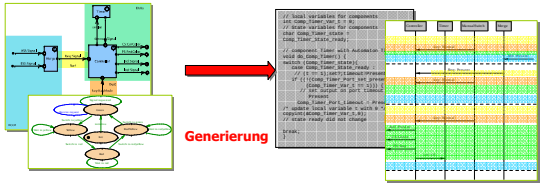
Modellbasiertes Testen



Modellbasiertes Testen: Analytische Qualitätssicherung

- Schritte: Modellbildung, Testfallgenerierung, Testfallinstantiierung
- Aspekte: Ableitung von Modellausführungen
  - Analysephase: Validierung Spezifikationsadäquanz
  - Testphase: Konformanz Spezifikation/Realisierung

Codegenerierung/Deployment



Codegenerierung/Deployment: Konstruktive Qualitätssicherung

- Schritte: Modellkonstruktion, Implementierungsgenerierung
- Aspekte: Automatisierung Implementierung
  - Wiederverwendbarkeit
  - Konstruktive Korrektheit (Zertifizierte Codegenerierung)

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?

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.