Karlsruher Institut für Technologie

Karlsruhe Institute of Technology

Modellbasiertes Testen in Agiler Softwareentwicklung

17. Juni 2010
David Faragó (farago@kit.edu)

Institut für Theoretische Informatik; Logik und Formale Methoden

30. Treffen der GI-Fachgruppe
Test, Analyse & Verifikation von Software (TAV)

Thema: Testing meets Agility

Übersicht

Einleitung : Agile Softwareentwicklung (ASE)

- iterativ, inkrementell:
  Anforderungen und Lösungen bilden sich heraus
  Sammlung bewährter Ingenieursmethoden
  schnelle Lieferung hochqualitativer Software-Inkremente

$\Rightarrow$ Schlüssel-Anforderungen
  - schnelle Lieferung funktionierender Software-Inkremente
  - flexibel gegenüber Änderungen

- Verbessert Qualität der Software durch Validierung

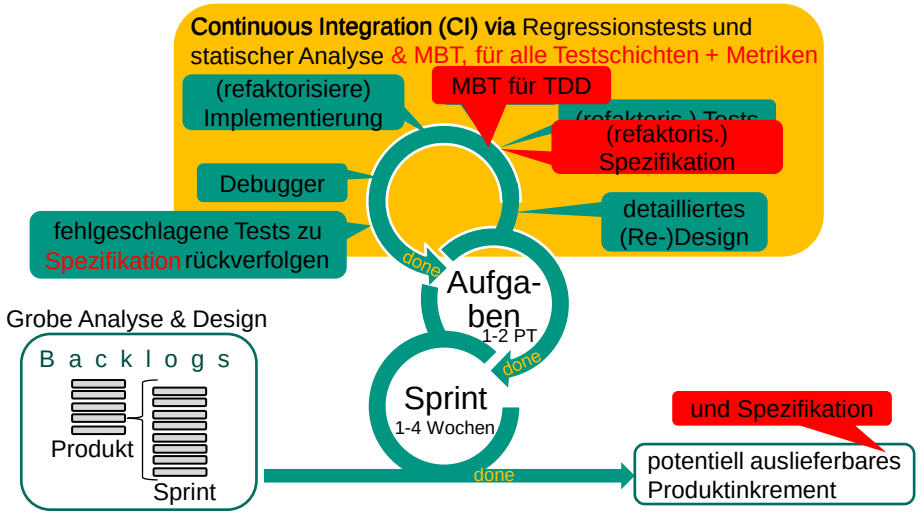
Einleitung : MBT mittels MC

Model Checking (MC)
- "hoch präzise statische Analyse"
- erschöpfende Traversierung des Zustandsraumes
- Anforderungen meist in temporaler Logik

Modellbasiertes Testen (MBT) mittels MC
- praktikabler als < Formale Verifikation, manuelle Testfallgenerierung
- MC generiert Testfälle automatisch aus (funktionalem) Modell
- Gegenbeispiel-Pfade als Testsequenzen < input: Testtreiber, output: Testorakel
- Spezifikation *s*: meist Transitionssystem
  (z.B. in STS, Promela, SDL, Stateflow, Spec#)
- Implementierung *i ioco s* $\Leftrightarrow$ Für alle spezifizierten Pfade:
  $\text{output}(i) \subseteq \text{output}(s)$ ($\wedge$ $\text{input}(i) \supseteq \text{input}(s)$)

- Verbessert Qualität der Software durch Verifikation

MBT in ASE: Prozess (am Beispiel Scrum & XP)



MBT in ASE: Nutzen der Integration

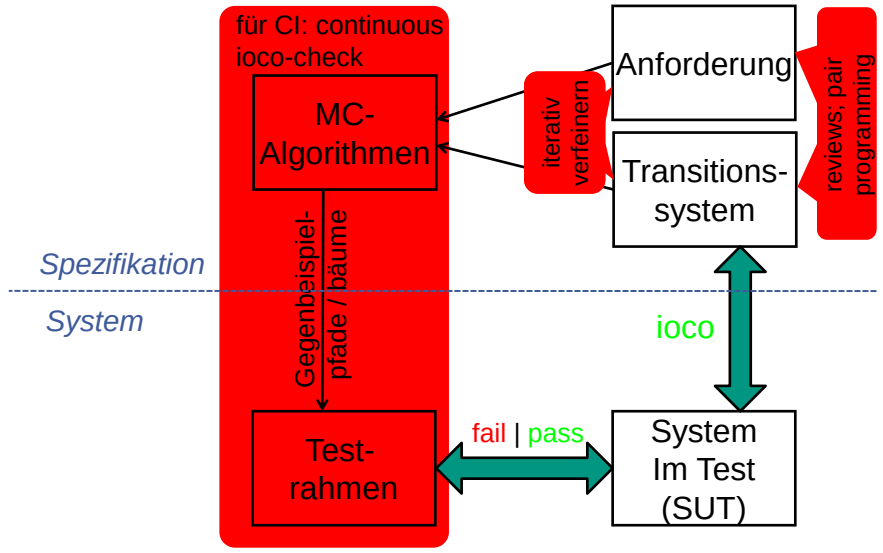
- Bessere Testabdeckung
- Änderungen, Konfigurationsmanagement & Rückverfolgung einfacher
- Liefert mehr Spezifikationen
  (z.B. zum Verteilen und Wiederverwenden von Komponenten)

! Aufwand für zusätzliche MBT-Spezifikation

! Harte Anforderungen an MBT-Technik

- Flexibles Spezifizieren, kein Big-Design-Up-Front (BDUF)
- schnelles MBT: innerhalb der Sprints; „10-minute build“

ASE in MBT: Prozess



ASE in MBT: Nutzen der Integration

- ASE für die MBT-Spezifikation selbst
  - Techniken wie Pair-Programming, Reviews
  - iterative, inkrementelle Prozesse

⇓

- Effizienteres MBT und kein BDUF
- ! MBT muss flexible Unterspezifikation bieten und effizient handhaben

Nichtdeterminismus

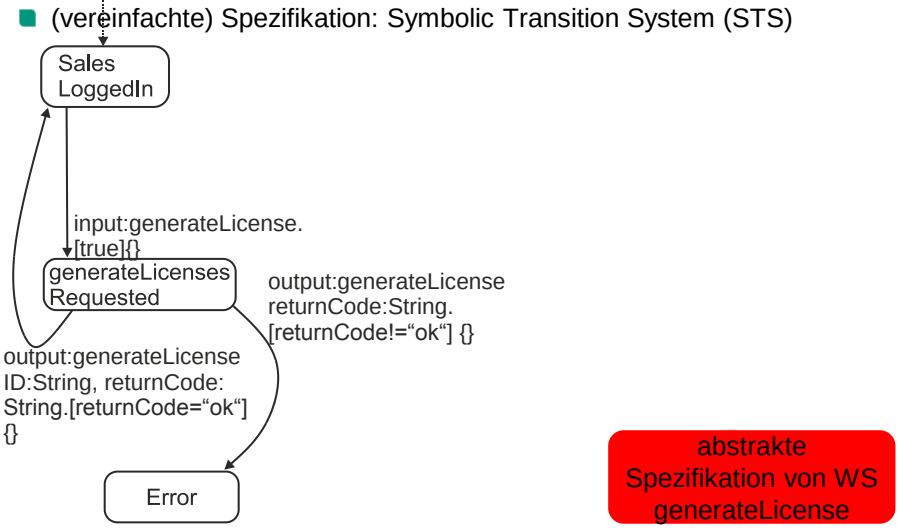
Zielsuche für Testselektion

Eng verzahnte Integration

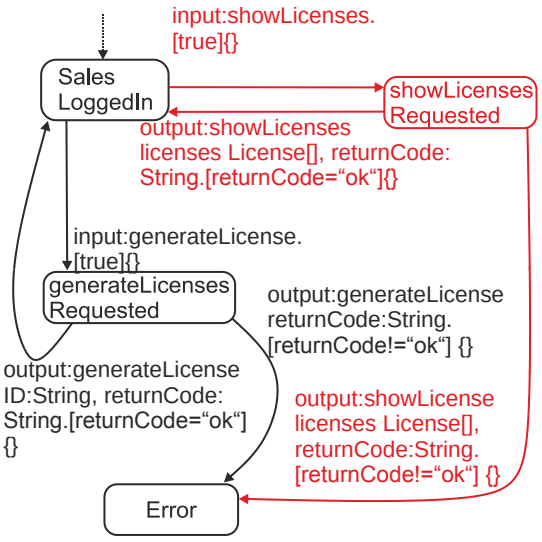
- Vereinigter Prozess
- vereinigte funktionale Spezifikation
  - weniger Aufwand für MBT-Spezifikation
  - keine Fehler durch Redundanz
  - abstrakten ersten Spezifikationen zur Übersicht und Kommunikation (z.B. statt user stories und use cases)
  - Verfeinerungen für MBT und Entwicklung
- Effektive CI mit vernünftigen Endekriterien (Definition of Done), Synergieeffekte durch continuous ioco-check

Fallbeispiel (Spezifikation)

- Aus Webservices (License Central von WIBU-SYSTEMS AG)
- (vereinfachte) Spezifikation: Symbolic Transition System (STS)

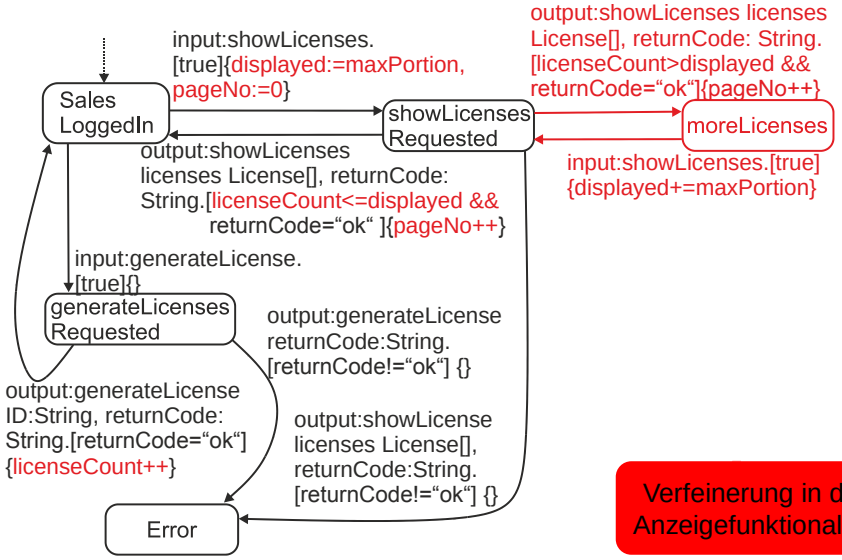


Fallbeispiel (Spezifikation)



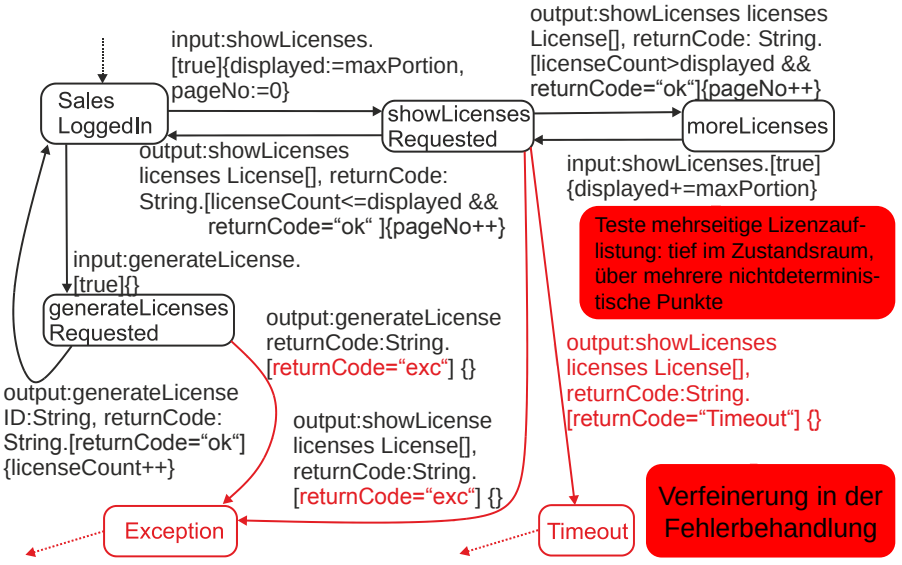
Verfeinerung: zusätzlicher WS showLicenses

Fallbeispiel (Spezifikation)



Verfeinerung in der Anzeigefunktionalität

Fallbeispiel (Spezifikation)



MBT-Verbesserung

Off-the-fly MBT: generiert —Test-suite→ Ausführung & Evaluierung

z.B. TGV, Spec Explorer

ineffizient, insbesondere bei Nichtdet. & Daten; keine dynamische Information

On-the-fly MBT: generiert —Test-schritt→ Ausführung & Evaluierung

z.B. TorX, UPPAAL Tron, Spec Explorer

schwache Zielsuche für Testselektion

Lazy on-the-fly MBT: generiert —Teil-Test-sequenz→ Ausführung & Evaluierung

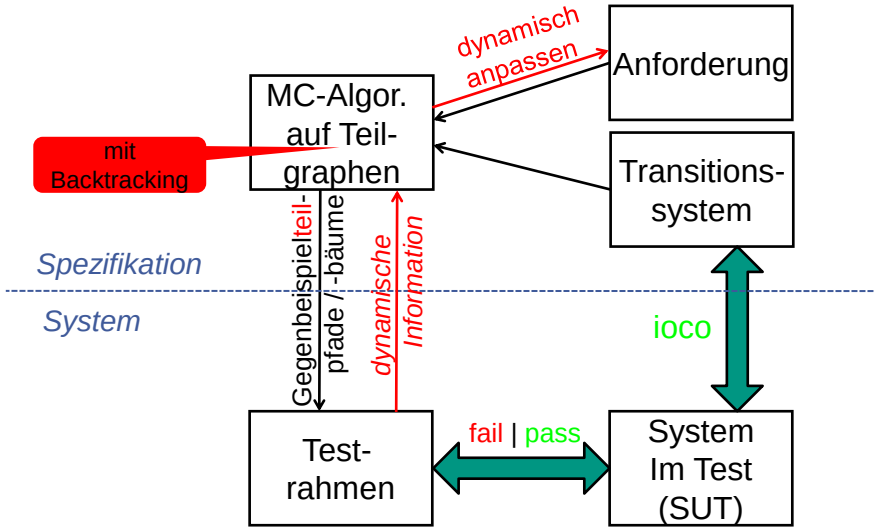
Initiiere Ausführung, z.B. beim Erreichen von

- Testzielpunkte
- einer bestimmten Tiefe
- Nichtdeterm. des SUTs

+aussagekräftigere Testfälle
+flexibler durch Reproduzierbarkeit und Wiederverwendbarkeit
+Nichtdeterminismus des SUTs effizient behandelbar

DFG-Projekt MOCHA, seit Oktober 2009

MBT-Verbesserung (2)

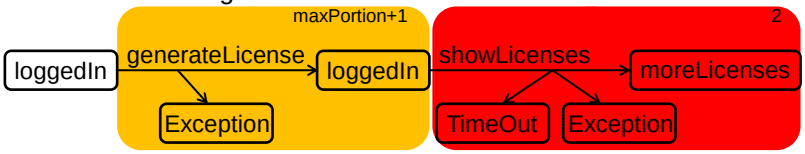


Fallbeispiel (MBT)

- Testziel: Lizenzauflistung über mehrere Seiten, (◊ pageNo>1)

⇒ Testsequenz: z.B. generateLicense$^{maxPortion+1}$ · showLicenses^{2}

- Off-the-fly MBT: potentiell unendlicher Zustandsraum durch Variablen, ggf. riesen Testbäume durch Nichtdet. (z.B. bei Fehlerbehandlung)
- On-the-fly MBT: besser, aber Testziel wird sehr ineffizient gefunden viele showLicenses (oder weitere WS) bis generateLicense$^{maxPortion+1}$
- Lazy on-the-fly MBT
  initiiere Ausführung in Fehlerzuständen und Testziel



Zusammenfassung

- Eng verzahnte ASE und MBT am effizientesten

⇒ effizienterer und flexiblerer Nichtdeterminismus in MBT-Techniken!

- Lazy on-the-fly MBT unterstützt dies

Vielen Dank für Ihre Aufmerksamkeit

Literaturverzeichnis (ASE)

- Scott Ambler (2002). Agile Modeling: Eective Practices for eXtreme Programming and the Unied Process. John Wiley & Sons, 2002.
- Scott Ambler (2008). *Has agile peaked?* Dr. Dobb's, May 07, 2008 http://www.ddj.com/architecture-and-design/207600615
- C3 Team (1998). *Chrysler goes to „Extremes“*, Distributed Computing, October 1998, 24—26.
- Forrester (2009). *Agile Development Method Growing in Popularity.* http://www.internetnews.com/dev-news/print.php/3841571
- Mika Katara and Antti Kervinen (2006). Making model-based testing more agile: A use case driven approach. In Haifa Verication Conference, volume 4383 of Lecture Notes in Computer Science. Springer, 2006.
- Martin Fowler et al. (2001). Agile Manifesto. http://agilemanifesto.org
- Olli-Pekka Puolitaival (2008). Adapting model-based testing to agile context. ESPOO2008.
- Ron Jeffries et al. (2000). Extreme Programming Installed. Addison-Wesley.
- J.B. Rainsberger (2009). *Integration Tests are a Scam*. Agile 2009 Conference.
- Hirotaka Takeuchi and Ikujiro Nonaka (1986). *The New Product Development Game.* Harvard Business Review.

Literaturverzeichnis (MBT)

- C. Jard and T. Jéron. TGV: Theory, principles and algorithms, a tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems. Software Tools for Technology Transfer (STTT), 6, October 2004.
- Kim G. Larsen, Marius Mikucionis, and Brian Nielsen. Online testing of real-time systems using UPPAAL. In In Proc. of the 4th Intl. Workshop on Formal Approaches to Testing of Software, pages 79–94. Springer, 2004.
- J. Tretmans. Model based testing with labelled transition systems. In Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers, volume 4949 of LNCS, pages 1–38. Springer, 2008.
- J. Tretmans and E. Brinksma. Torx: Automated model-based testing. In First European Conference on Model-Driven Software Engineering, Nuremberg, Germany, pages 31–43, December 2003.
- Mark Utting and Bruno Legeard (2007). Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann, 1 edition, 2007
- Margus Veanes, Colin Campbell,Wolfgang Grieskamp,Wolfram Schulte, Nikolai Tillmann, and Lev Nachmanson. Model-based testing of object-oriented reactive systems with Spec Explorer. In Formal Methods and Testing, volume 4949 of LNCS, pages 39–76. Springer, 2008.