Technische Universität München **SS 2007**
Institut für Informatik
Prof. Dr. Dr. h.c. Manfred Broy, Dr. Bernhard Schätz
Dr. Martin Rappl, Doris Wild **Musterlösung zu Blatt 11**

Übungen zu „Softwarearchitekturen und modellgetriebene Softwareentwicklung“

Aufgabe 1 Richtlinien zur Modellierung [LÖSUNG]

Machen Sie sich mit einer typischen Modellierungsrichtlinie, wie z.B. der Richtlinie des MAAB (The MathWorks Automotive Advisory Board) für Matlab/Simulink/Stateflow (siehe http://www.mathworks.com/industries/auto/maab.html), vertraut.

(a) Was sind typische Vorschriften für Modelle, wie sie auch bei Code zu finden sind?
Antwort:
Beispielsweise Vorschriften zur Namensgebung (sprechende Namen, Konsistenz, ...) oder unerreichbare Modellteile (= toter Code). Interessanterweise gibt es in Simulink auch ein Goto-Konstrukt (das natürlich auch vermieden werden sollte). Darüber hinaus gibt es analog zu den MISRA-Richtlinen für C inzwischen auch einen MISRA-Standard für Simulink, der „gefährliche“ Modellelemente verbietet.

(b) Welche Vorschriften sind speziell für (graphische) Modelle vorhanden?
Antwort:
Die graphische Anordnung auf dem Bildschirm enthält viele neue Möglichkeiten das Lesen der Modelle zu erschweren. Es geht nicht mehr nur um Einrückung und Abstand, sondern auch um die zweidimensionale Anordnung. Oft kommt sogar noch eine dritte Dimension hinzu, da sich Modellelemente evtl. überdecken können. Weiterhin sind Farbgebung, Schriftart und -größe speziell für graphische Modelle interessant. Oft kommen außerdem noch weitere Einschränkungen durch den Code-Generator hinzu, der evtl. nicht alle Modellelemente unterstützt oder zusätzliche reservierte Wörter einführt.

Aufgabe 2 Modellbasiertes Testen [LÖSUNG]

Beim modellbasierten Test werden explizite Modelle zur Generierung von Testfällen für das SUT (*S*ystem *u*nder *T*est) verwendet. Üblicherweise werden dazu Verhaltensmodelle (Automaten, Sequenzdiagramme) verwendet.

(a) Definieren Sie die Begriffe „Testfall“ und „Testsuite“.
Antwort:
Ein Testfall ist eine endliche Sequenz von Ein- und erwarteten Ausgaben an das System. Eine Testsuite ist eine Sammlung zusammengehöriger Testfälle.

(b) Analysieren Sie unter diesem Gesichtspunkt nochmals das Beispiel des Instrument Clusters vom Übungsblatt 6 und 9. Welche Informationen müssten Sie mindestens

noch hinzufügen, um aus dem Automaten Testfälle generieren zu können?
Es müssten evtl. Zeitanforderungen (time-outs, ...) mit angegeben werden. Außerdem muss klar sein, wie die logischen Aktionen in der realen Implementierung dargestellt werden. Diese Übersetzung kann im Testtreiber stattfinden.

(c) Bewerten Sie die Qualitätssicherungsmaßnahme „Modellbasiertes Testen“.
Der modellbasierte Test stellt ein strukturiertes Vorgehen da, um eine große Zahl an Testfällen unabhängig vom Programmcode (also Black-Box) zu generieren. Einer der Hauptvorteile liegt in der expliziten Modellierung des SUT, das bereits zu einer Aufdeckung von Fehlern in (textuellen) Spezifikationen führt. Weiterhin kann dieses Modell auch als Testorakel dienen, d.h. es liefert die erwarteten Ausgaben für die generierten Eingaben. Als negativ zu sehen ist der hohe Aufwand für die Modellierung, dass nur das getestet wurde, was auch modelliert wurde und, dass noch keine systematischen Vorgehen zur Generierung (Was ist ein „interessanter“ Testfall?) existieren.