Übungen zur Vorlesung
Testautomatisierung
Wintersemester 2002/03

AG BS
Jan Peleska, Aliki Tsiolakis

**Blatt 2**

# Test-Suite zum Testen des *Engine Controllers*

In diesem Aufgabenblatt soll eine komplette Test-Suite zum Testen des bereits aus Blatt 1 bekannten *Engine Controllers* entwickelt werden, die eine komplette Strukturabdeckung des System Under Test (SUT) erzeugt. Den Source-Code des SUT `rttsut.c` finden Sie in dem Archiv `engineController_TEMPLATE.tgz`, der auch die benötigten Interface-Module enthält.

## Aufgabe 1: Erstellung eines Anforderungsdokuments — 35%

Erstellen Sie ein Anforderungsdokument, dass die Requirements des SUT beschreibt. Gliedern Sie die Requirements in

- Requirements
- Derived Requirements
- Design Requirements
- Derived Design Requirements

Die Anforderungsliste soll auf der Beschreibung des SUT aus dem letzten Aufgabenblatt basieren, die noch einmal im Appendix zu finden ist (im Zweifelsfall gilt die Version im Appendix). Untersuchen Sie zusätzlich die Implementierung des SUT (`rttsut.c`), um durch „Reverse Engineering“ Designanforderungen abzuleiten.
Jedes Requirement in der Liste soll durch ein *Requirement Tag* eindeutig identifiziert sein. Abgeleitet Anforderungen müssen zusätzlich einen Link auf das eigentliche Requirement Tag enthalten.

## Aufgabe 2: Erstellung einer Test-Suite — 35%

### Teilaufgabe 1: *Requirements Coverage*

Erstellen Sie anhand des Anforderungsdokuments aus Aufgabe 1 eine Test-Suite, die die volle Anforderungsüberdeckung sicherstellt. Sie können dazu mehrere Testfälle erstellen (beispielsweise gegliedert in `test1`, `test2`,...), die jeweils bestimmte Anforderungen testen sollen. Dokumentieren Sie dies pro Testfall geeignet.
Erstellen Sie ggf. Test-Stubs, die notwendiges Funktionsverhalten entsprechend simulieren.
Zum Testen fügen Sie die Requirements-Tags geeignet in Ihre CSP-Spezifikationen ein, so dass sie mit im Log erscheinen und die Log-Dateien dann nach der Test-Durchführung ausgewertet werden können (z.B. mit dem Skript `rttmkdoc`).
Die von Ihnen in Blatt1 erstellten `USER`- und `CHECKER`-Spezifikationen können bei Bedarf angepasst und weiterverwendet werden.

### Teilaufgabe 2: *Code Coverage*

Erweitern Sie Ihre Test-Suite so, dass Sie volle C0-Code-Coverage erzielen. Erstellen Sie ggf. auch hierfür Test-Stubs, die notwendiges Funktionsverhalten entsprechend simulieren.
Dokumentieren Sie die neuen Testfälle geeigent.

## Aufgabe 3: Durchführung der Tests — 30%

Führen Sie die in Aufgabe 2 erstellten Testfälle aus und dokumentieren Sie die Ergebnisse der Testdurchführung (pro Testcase).

Die Dokumentation für die **schriftliche Abgabe** soll enthalten:

- Anforderungsdokument
- Auflistung der Testfälle
- Dokumentation der Testfälle
- Dokumentation der Ergebnisse pro Testfall

Achten Sie darauf, dass die Testlogs in `testdata` jeweils nur vom letzten Testlauf sind! Zur Erstellung können Sie `rttmkdoc` und `rttdeldoc` verwenden.
Das **Archiv**, das per Email eingereicht wird, soll zusätzlich die Code-Coverage-Datei des Testlings enthalten (`rttsut.c.gcov`).

## Abgabe: Bis Dienstag, 3. Dezember 2002, vor dem Tutorium.

Geben Sie für alle Aufgaben eine **schriftliche Lösung** ab. Bitte schicken Sie zusätlich ein Archiv mit allen Testfällen, Testlogs und Dokumentation an `tsio@informatik.uni-bremen.de`.

## Appendix A: Beschreibung des *Engine Controllers*

Die Firma *TA Mobil* hat ein neues Auto auf den Markt gebracht, das besonders auf Sonntagsfahrer im Stadtverkehr zugeschnitten ist: Die Maximalgeschwindigkeit von 50km/h darf nicht länger als 5sec gefahren werden.

Das Auto kann durch drei spezielle Kontroll-Kommandos vom Fahrer „gesteuert“ werden:

- Mit dem Kontroll-Kommando speedUp (x) kann das Auto so beschleunigt werden, dass die Geschwindigkeit pro Zeitschritt um x km/h erhöht wird.
- Umgekehrt kann die Geschwindigkeit des Autos um x km/h pro Zeitschritt gedrosselt werden, indem das Kontroll-Kommando slowDown (x) an den Motor weitergeleitet wird. (Allerdings soll der *Engine Controller* keine Signale an den Motor weiterleiten, wenn das Auto bereits steht.)
- Mit dem Kontroll-Kommando steady wird die aktuelle Geschwindigkeit beibehalten. (Das steady-Kommando ist eine Art Abbruchkriterium für die Kommandos speedUp und slowDown.

Diese Kontroll-Kommandos werden an den *Engine Controller* gesendet, der daraufhin den Motor beschleunigt (accelerate) oder abbremst (brake). Die Umsetzung der Kontroll-Kommandos muss innerhalb von $t_1 = 1sec$ erfolgen (dies entspricht auch einem Zeitschritt zur Erhöhung oder Senkung der Geschwindigkeit durch die entsprechenden Kontroll-Kommandos).

Sobald das Auto die maximale oder minimale Geschwindigkeit erreicht, gibt der Motor ein kurzes Signal, das je nach persönlicher Einstellung des Fahrers durch ein kurzes akkustisches oder visuelles Signal angezeigt werden könnte:

- Das Signal maxSpeedReached zeigt an, dass die maximale Geschwindigkeit von 50km/h erreicht wurde.
- Das Signal zeroSpeedReached zeigt an, dass die minimale Geschwindigkeit (0km/h) erreicht wurde und das Auto steht.

Als erzieherische Maßnahme für Raser (und zur Schonung des Motors) kann die maximale Geschwindigkeit aber nicht länger 5sec gefahren werden. Deshalb muss der Fahrer innerhalb von 5sec nach Erreichen der maximalen Geschwindigkeit (also innerhalb von 5 sec nach dem Signal maxSpeedReached), wieder eine Abbremsung des Autos auslösen (Kontroll-Kommando slowDown). Falls dies nicht geschieht, leitet der *Engine Controller* eine Notbremsung ein, die dazu führt, dass das Fahrzeug mit maximaler Verzögerung abgebremst wird.

Für die Darstellung der aktuellen Geschwindigkeit $x$ auf dem Tacho sendet der Motor jede Sekunde ($t_0 = 1sec$) das Signal speed (x) an den *Engine Controller*.

Zum Testen der Engine Controller-Software steht die Original-Hardware des Motors leider nicht zur Verfügung. Das Verhalten des Motors wird in dem CSP-Prozess `ENGINE` simuliert (siehe Datei `engine.csp`). Das CSP-Interface für die *Engine Controller*-Software steht in der Datei `ifm_sut.csp` zur Verfügung.