Gruppe: HK-07-4
Gruppenleiter: Stanley Hillner 29.05.2007

Testkonzept-HK-07-4

1. Junit (V. 3.8.1) – automatisiertes Testkonzept in Java

Testklassen:

Eine Testklasse implementiert immer das Interface TestCase. Variablen werden mit vorgegebenen Assertmethoden auf Richtigkeit, Gleichheit, etc. getestet. Diese Assertmethoden müssen innerhalb von Methoden aufgerufen werden, welche mit Test beginnen. Es gibt noch Suites. Diese sind auch Testklassen, brauchen aber kein Interface sondern nur ein Suitemethode vom Typ Test. In ihr werden mit addSuite (Testklasse) Testklassen hinzugefügt. Um die vorgegebenen Methoden benutzen zu können, muss man Junitpakete importieren. Für Assertmethoden z.B junit.framework.assert. Es sind auch spezielle Methoden vorhanden um Fehler zu simulieren. (meist mit fail(...)) So kann man z.B. testen ob eine Exception geworfen wird oder nicht und ob es die richtige Exception ist.

Durchführung:

Um eine Testklasse ausführen zu können gibt es runner. Diese sind vorgegeben oder auch selber erstellbar und müssen in einer der Mainmethoden der Testklassen der Suite aufgerufen werden. z.B. junit.swingui.TestRunner.run(AllTests.class); Darauf öffnet sich eine Auswertungsoberfläche, welche schiefgelaufene Test anzeigt und mit einem roten bzw. grünen Balken den Teststatus angibt.

2. Manuelles Testen

Da es nicht möglich ist, alle Komponenten und Funktionalitäten einer Software ohne Weiteres automatisch zu testen, sind manuelle Tests unumgänglicher Bestandteil der Prüfung der Funktionalität einer Software.
Zu diesem Zweck werden die einzelnen Funktionen der Software (hier der GMF-Editor) manuell von den Teammitgliedern auf ordnungsgemäßes Verhalten getestet.
Es sollen einzelne Szenarien, wie sie im täglichen Gebrauch des Editors auftreten, erstellt und durchgespielt werden. Die Szenarien sollen eine möglichst große Bandbreite an möglichen Ereignissen des täglichen Gebrauchs abdecken, welche die Software dann erfolgreich bewältigen muss.
Zu den durchlaufenen Tests werden jeweils eigene Testberichte angefertigt, welche

Aufschluss über die Einhaltung des Implementierungsplans geben und gegebenenfalls erforderliche Verbesserungen oder Nacharbeiten aufzeigen.

3. Testschwerpunkte

Da es sich bei unserem Projekt um ein Eclipse-Plugin handelt und somit sowohl sehr starke Abhängigkeiten zwischen dem Editor und GMF, als auch zu Eclipse ergeben, kann man das ordnungsgemäße Funktionieren der Komponenten, auf welche der Editor aufbaut, bereits voraussetzen.
Außerdem ist es relativ schwierig, die einzelnen Bestandteile des Editors einzeln zu testen, also ohne dass diese in den Editor eingebunden sind. Aus diesem Grund wird das Hauptaugenmerk beim Testen auf dem Durchlaufen manueller Testszenarien liegen. Es bietet sich jedoch trotzdem an, einige automatisierte Testklassen zu schreiben, mit welchen das Testen einzelner kleiner Funktionen sehr viel schneller von Statten geht.
Sowohl die Testklassen, als auch die Testberichte werden dann zum jeweiligen Release-Bündel hinzugefügt.