GEO01 - SOFTWARE SOLUTIONS

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Daniel Seifarth, Anne Güpner
Am: 09.06.03

Das Testkonzept

1. Einführung

Es ist ein Geometrieprogramm zu testen, welches nicht sehr umfangreich gestaltet ist und das nur die nötigsten Funktionen beinhaltet. So müssen verschiedene einfache geometrische Berechnungen auf ihre Richtigkeit geprüft werden. (z.B. Mittelpunkt Berechnung)
Alle Funktionen müssen die komplette Funktionalität aufweisen können. Die Testreihen werden verschiedene Stufen zu durchlaufen haben. Es fängt mit einem einfachen Komponententest an, den die einzelnen Klassen und Methoden durchlaufen, es geht dann weiter mit einem Integrationstest der verschiedenen Klassen und es wird geprüft wie sie untereinander und miteinander agieren und es wird mit einem Systemtest abgeschlossen. Alle vorhandenen Funktionen müssen dabei überprüft werden. Als Werkzeug benutzt man hier JUnit, sowie einfache und komplexe Testbeispiele, welche man zum Verarbeiten benutzt.

2. Zu testende Komponenten

Es liegen verschiedene Klassen vor, z.B. HauptController, BasisController, PunktReferenzController, KonstruktionController, BasisView, KonstruktionView, PunktreferenzView und Konstruktion, um die Hauptklassen zu nennen. Die einzelnen Klassen sind, jede für sich, während der Programmierung zu testen. Einzuarbeitende Methoden und Parameter sind auf ihre Funktionalität zu überprüfen. Es werden hierfür, genau wie bei den Funktionen, Testsuites erstellt. Mit Hilfe dieser Testsuites kann man die korrekte und vollständige Funktionalität der vorliegenden Komponenten überprüfen. Da es eine Dreiteilung der Programmierung geben wird, werden die einzelnen PP´s ihre jeweilige Komponente selbstständig testen müssen.
Es liegen vor: Controller Klassen , View Klassen und Konstruktion Klassen. Controller und View erstellen eine Sicht des Programms. Hier muss überprüft werden ob alle Buttons angelegt werden, sowie ob alle Frames richtig eingebunden sind. Dies macht man zum einen durch eine einfache Hauptklasse, welche je einen Controller oder View anlegt, oder man programmiert eine Testsuite für JUnit. Ersteres ist aber für dieses Problem eine trivialere Lösung und sollte hier verwendet werden.
Für die Konstruktion hingegen wird eine Testsuite benötigt. Diese muss testen, ob sich die einzelnen Objekte anlegen lassen und ob die Attribute, die sie enthalten, auch das enthalten können, was sie sollen (und auch nicht mehr). Sowie, dass die Funktionen, die eingearbeitet werden, in dem Rückgabewertebereich liegen der zu erwarten ist. (Mittelpunkt Berechnung würde hier zwei Integerwerte zurück liefern) Diese Funktionen werden aber für sich noch einmal bei den Funktionstest geprüft.
Die Testsuites werden mit jeder Story erweitert. Es wird also vier verschiedene Testsuites geben. Für jede Woche eine. Es kann aber sein das sich die Testsuite nicht von der der vorherigen Woche unterscheidet da nur an den Funktionen gearbeitet wird und diese sollten in der Testsuite enthalten sein.

3. Zu testende Funktionen

Zeichenoberfläche anlegen, Punkt per Maus anlegen, (x,y)-Punkt anlegen, Gerade anlegen I, Strecke anlegen I, Halbgerade anlegen I, Gerade anlegen II, Strecke anlegen II, Halbgerade anlegen II, Mittelpunkt anlegen, Punkt löschen, Punkt verschieben, Abstand zwischen zwei Punkten berechnen, Koordinatenachsen einblenden (ausblenden), Gitternetz ein- (und ausblenden), Punktbeschriftung einblenden (ausblenden), Punkt färben, Zoomen
Die Funktionen sind aus dem Pflichtenheft entnommen.

4. Nicht zu testende Funktionen

Benutzen der Hilfe
Konstruktionsschritt ausführen

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Daniel Seifarth, Anne Güpner
Am: 09.06.03

5. Vorgehen

Es werden für jede Funktion einzelne Testsuites geschrieben. Manche Funktionen werden, wenn möglich, im laufenden Programm getestet, so z.B. Gitternetz ein(aus)blenden, Koordinatenachsen ein(aus)blenden, Punktbeschriftung ein(aus)blenden, zoomen. Für die restlichen Funktionen werden die sogenannten Testsuites geschrieben. Für jede Funktion, welche programmiert werden soll, wird eine Testsuite durch das jeweilige PP-Team angelegt. Hierbei wird der zu erwartende Rückgabewert überprüft, mittels der von JUnit vorgegebenen assert Methoden. Sollte kein Rückgabewert vorhanden sein, werden die Parameter der jeweiligen Methoden getestet. Hierbei werden die zu erwartenden Werte in die Testsuite eingetragen. Da die Funktionen nur im Grundgerüst vorhanden bzw. noch gar nicht existent sind, wird JUnit erstmal Fehler bringen. Es werden nun die Funktionen implementiert und gleichzeitig auf ihre Funktion überprüft. Die Funktionen werden in Gruppen eingeteilt.

Gruppe - Anlegen:
(x,y)-Punkt anlegen, Gerade anlegen I, Strecke anlegen I, Halbgerade anlegen I, Gerade anlegen II, Strecke anlegen II, Halbgerade anlegen II, Mittelpunkt anlegen

Hierbei wird überprüft ob die Objekte angelegt wurden, also ob sie existent sind. Und ob sie auf dem View auch angezeigt, sowie in der Referenzliste angelegt werden. Bei Mittelpunkt anlegen muss ein besonderes Augenmerk darauf gelegt werden, das hierbei eine Berechnung durchgeführt werden muss. Diese muss auf ihre Korrektheit überprüft werden.

Die zweite Gruppe sind die Funktionen, welche einen Parameter verändern und damit die Ansicht der View:
Koordinatenachsen einblenden (ausblenden), Gitternetz ein- (und ausblenden), Punktbeschriftung, einblenden (ausblenden), Punkt färben, Zoomen

Zu überprüfen sind hierbei die Parameter und ihre korrekt veränderten Eigenschaften.
(Hierbei wird recht häufig assertboolean verwendet werden müssen.)

6. Kriterien für erfolgreiche bzw. fehlgeschlagene Testläufe

JUnit gibt an ob ein Test erfolgreich war. Sollte ein Test fehlschlagen, so ist zu überprüfen, ob ein herabsetzen der Anforderungen an den Test immer noch die Funktionalität und die Anforderungen an das Programm genügt.

7. Testtätigkeiten

Vor jedem Test sind TestSuites anzulegen, bzw. zu ergänzen. Es sind keine Spezialkenntnisse erforderlich um die Tests durchzuführen. Als einziges wird Java Wissen und Umgang mit JUnit vorausgesetzt. Wünschenswert ist auch die Installation von Eclipse, welches CVS und JUnit unterstützt.

8. Umgebung

Als Umgebung sollte Eclipse mit integriertem JUnit verwendet werden. Eine Java Umgebung ist sowieso grundsätzlich vorgesehen. Das Programm sollte auf jedem Betriebssystem mit einer Java Umgebung laufen, es ist also kein bestimmtes Betriebssystem vorgeschrieben.

04109 Leipzig • Augustusplatz 10
mail: softwaresolutions@everyday.com

Erstellt von: Daniel Seifarth, Anne Güpner
Am: 09.06.03

9. Verantwortlichkeiten

Verantwortlich für die Tests ist Daniel Seifarth als Koordinator der Tests, aber es wird eine Unterstützung der einzelnen PP-Teams vorausgesetzt. Der Verantwortliche für Tests steht hierbei nur als Ansprechpartner zur Verfügung. Grossteile der Tests sollten von den PP-Teams selber geschrieben und verwaltet werden.

10. Zeitplan

Ein Konkreter Zeitplan wird nicht vorausgesetzt. Nur am Ende jeder Woche, sollten die vorhandenen Testsuites keine Fehler bringen. Also zum Abgabetermin dürfen keine Fehler auftreten.