Testkonzept
Gruppe geo10

Thomas Pröger - Verantwortlicher für Test

9. Juni 2003

1 Allgemein

1.1 Was soll getestet werden...

Zur Unterstützung des laufenden Entwicklungsprozesses ist es notwendig die einzelnen Teilprodukte sowie das gesamte System auf Korrektheit bezüglich der, vom Kunden geforderten, Leistungsparameter zu testen. Zu diesem Zwecke werden regelmäßige Komponenten- und Systemtests durchgeführt. Dabei wird während der Implementation jede Komponente(Klasse), mit Hilfe von automatischen Tests, auf Einhaltung der Produktspezifikationen(siehe Abschnitt 2) geprüft. Besonderes Augenmerk wird dabei auf der Model-Komponente liegen, da hier die meisten Testansätze zu finden sind. Für die eher ereignissorientierten Prozesse, der View- und Controller-Komponenten ist das verwendete Unit-Framework nur bedingt geeignet. Dafür stehen weitere Softwareprodukte zur Verfügung, die aber im Rahmen dieses Projektes nicht verwendet werden.

1.2 Wer muß welche Tests liefern...

Im Allgemeinen muß jede Gruppe alle geforderten Tests für die, von ihr bearbeiteten, Storie(s) liefern. Das beinhaltet alle Tests für die Klassen, die in der jeweiligen Storie enthalten sind. Sollten den Gruppenmitgliedern Spezifikationen auffallen, für die in der Planung bisher keine Tests vorgesehen waren, können die neuen Tests, unter Absprache mit dem Testleiter, in das Konzept aufgenommen und implementiert werden.
Einmal erstellte Tests werden stets zusammen mit ihren jeweiligen Testobjekten(Klassen) weitergegeben, sodass selbst bei Ausführung einer fertig implementierten Klasse alle abhängigen Komponententests mit durchgeführt werden. Damit wird gewährleistet, dass neu erstellte Komponenten keinen Einfluß auf bestehnde funktionierende Komponenten haben. Die Arbeit einer Gruppe gilt am Ende einer Zeitscheibe erst als "abgenommen", wenn sie den erfolgreichen Durchlauf aller Tests entsprechend der Spezifikationsliste vorweisen kann.

1.3 Wie sind die Tests zu organisieren...

Zur Integration in das laufende Projekt wird ein Package 'Tests' angelegt, welches alle Testklassen und Testsuites kapselt. Pro Storie wird von der jeweiligen Gruppe eine Testsuite implementiert, die alle erforderlichen Tests beinhaltet. Dabei muß darauf geachtet werden, dass diese Suite nicht nur die neu erstellten Testcases sondern auch die bisherigen Tests der verwendeten Klassen umfasst.
Zu jeder Klasse wird eine entsprechende Testklasse erstellt, die alle zu testenden Methoden berücksichtigt. Dabei ist auf folgende Namenskonventionen zu achten:

Testklassen: "Klassenname+Test", z.B.: Line → LineTest

Testsuites: "Storytitel+Tests", z.B.: Story02 → Story02Tests

Das Testpackage wird bei jedem Gruppenwechsel zusammen mit den restlichen Code-Produkten aktualisiert und auf den neuesten Stand gebracht, sodass es zu Beginn einer Zeitscheibe alle bisher erstellten Testklassen und -suites umfasst.
Zum Abschluß einer Zeitscheibe wird noch einmal eine Testsuite erstellt, die alle Anforderungen an das bestehende System untersucht. Erst nach erfolgreichem Durchlauf dieser Tests kann das bestehende System als neue Release ausgeliefert werden.

1.4 Was passiert wenn einzelne Tests nicht laufen...

Falls ein Test eines Modules(Story) nicht erfolgreich durchgeführt werden kann, wird das betreffende Modul als "nicht implementiert" betrachtet.
Solche Module können aus Zeitgründen(Releasetermine) vorübergehend in das bisherige System

aufgenommen werden. Im Allgemeinen ist aber davon abzusehen, da dieses Vorgehen den Grundgedanken des XP-Paradigmas, in dem eine Komponente erst als implementiert gilt wenn sie alle erforderlichen Tests erfolgreich durchläuft, widerspricht.

2 Spezifikationen

Die folgende Liste umfasst ein Sammlung aller Spezifikationen an die zu erstellenden Klassen. Dabei wurde sich bewußt auf die Klassen als Ausgangspunkt konzentriert, um die Unabhängigkeit von der Gestaltung der Stories zu gewährleisten.

2.1 Model

2.1.1 GeoModel

- neu erstellte GeoElement-Objekte werden in die Liste der Elemente aufgenommen

2.1.2 Point

- der Abstand zwischen zwei Punkten wird korrekt berechnet

2.1.3 Line

- der Anstieg m wird korrekt bestimmt
- der Schnittpunkt mit der y-Achse wird korrekt ermittelt

2.1.4 FixedPoint

- der FP wird als FixedPoint initialisiert
- der FP liegt auf allen Line-Objekten des upperDependences-Array
- der FP wird als Schnittpunkt markiert, wenn das upperDependences-Array mehr als ein Element enthält
- ein Schnittpunkt ist kein Slider, da nicht klar ist auf welcher Geraden er sich bewegt

2.1.5 Slider

- verschieben: neuer Punkt (wenn Maus bewegt) liegt wieder auf der Geraden
- verschieben: es wird das Maximum von ΔX und ΔY zur Berechnung der neuen Position des Slider verwendet

2.2 View

2.2.1 GeoPanel

- Änderungen des Skalierungsfaktors werden korrekt berechnet