Akzeptanztesten mit Integrity und FitNesse - Ein Vergleich

Dehla Sokenou
GEBIT Solutions
Koenigsallee 75b, 14193 Berlin, Germany
dehla.sokenou[at]gebit.de

1 Einführung

Als letzte Phase im Softwaretest steht in der Regel der im Idealfall durch den Kunden durchgeführte Akzeptanztest. Während es für die frühen Phasen des Testens, die oft von den Entwicklern selbst oder von einer Testgruppe im Projekt durchgeführt werden, sehr viel Werkzeugunterstützung (Werkzeuge für Unittests, Lasttests, Überdeckungsmessung, Mutationstest, etc) gibt, ist das Feld beim Akzeptanztest überschaubar. Hier gibt es im Wesentlichen zwei Vertreter, die scriptorientierten Testwerkzeuge oder Capture&Replay-Werkzeuge. Während die ersteren eine mehr oder weniger an natürliche Sprache angelehnte Syntax verenden, um Testfälle zu definieren, nutzen die letzteren einen eigenen Rekorder, mit dem der Tester die Software bedient, um so alle Testschritte später abspielen zu können. Zu den scriptbasierten Werkzeugen gehören FitNesse [2], Integrity [3] und Behavior-Driven-Testwerkzeuge wie JBehave [4] oder Cucumber [1], einer der bekanntesten Vertreter der Capture&Replay-Werkzeuge ist Selenium [6].

Scriptbasierte Testwerkzeuge haben den entscheidenden Vorteil, dass sie flexibel einsetzbar sind. So kann mit ihnen nicht nur GUI-zentriert getestet werden, wie mit den Capture&Replay-Werkzeugen, sondern es können Tests von APIs oder Datenbanken erfolgen. So kann bspw. auch einfach das Zusammenspiel verschiedener Applikationen getestet werden.

In unseren Projekten benutzen wir beide Varianten, allerdings mit einem starken Fokus auf scriptbasierte Testwerkzeuge. Für Webtests wird überwiegend Selenium eingesetzt, auch gekoppelt mit FitNesse oder Integrity als reine Ausführungsumgebung, so dass auch hier die Vorteile von scriptbasierten Testwerkzeugen zum Tragen kommen. Akzeptanztests für Rich-Client-Anwendungen nutzen dagegen überwiegend FitNesse oder Integrity, wir haben aber auch in einigen Projekten Erfahrungen mit Capture&Replay-Werkzeugen (bspw. Abbot [7], QF-Test [5]) gesammelt.

Im folgenden wird das Testframework Integrity vorgestellt und dem etablierten Testframework FitNesse gegenübergestellt.

2 Integrity

Integrity ist ein in den letzten Jahren bei GEBIT intern entwickeltes Testframework, das neben FitNesse seit etwa zwei Jahren im produktiven Einsatz ist. Inzwischen ist es als Open-Source-Testwerkzeug unter der Eclipse Public License (EPL) veröffentlicht.

Integrity stellt im Prinzip nur die Basis für die Definition und Ausführung von Tests zur Verfügung. Diese muss vom jeweiligen Testteam noch an die eigenen Bedürfnisse angepasst werden. Dazu werden so genannte Fixtures implementiert, die die zu testende Anwendung an das Integrity-Framework binden. Als vordefinierte Fixtures wird bisher ein Paket mit Fixtures zum Testen von Swing-Anwendungen bereitgestellt. Um Fixtures auch nutzbar zu machen, muss eine Fixture-Deklaration erfolgen. Somit können mehr Fixtures implementiert sein, als für das aktuelle Projekt zur Verfügung gestellt werden.

Integrity ist in Eclipse integriert. Es gibt einen XText-basierten Editor, mit dem einerseits die Testfälle erstellt werden können, der aber andererseits auch dazu dient, das Integrity-Binding an die implementierten Fixtures (die Fixture-Deklaration) vorzunehmen. Der Editor unterstützt Fehler- und Syntaxhighlighting sowie Autovervollständigung, was die Testfallerstellung erleichtert. Tests werden in einer eigenen Syntax definiert, die einfache Aufrufe und Testtabellen (ähnlich wie FitNesse) unterstützt.

Neben dem Editor bringt Integrity eine Testausführungsumgebung mit, die es erlaubt, sich zur Laufzeit der Tests zu attachen. Dies nutzt das ebenfalls in Eclipse integrierte Testcontrol, um die laufenden Tests und die Testergebnisse anzuzeigen. Die Laufzeitumgebung lässt sich auch ohne Eclipse nutzen, z.B. im automatischen Build. Ein Plugin für den Jenkins-Buildserver ist verfügbar.

3 FitNesse

Bis zur Entwicklung des Integrity-Testframeworks war FitNesse das Mittel der Wahl bei GEBIT, wenn es um Akzeptanztests ging. Diese wurden von eigenen Testern oder auch von Kundenseite durchgeführt.

FitNesse ist ein Open-Source-Werkzeug, das ursprünglich auf dem FIT-Framework [8] aufbaut.

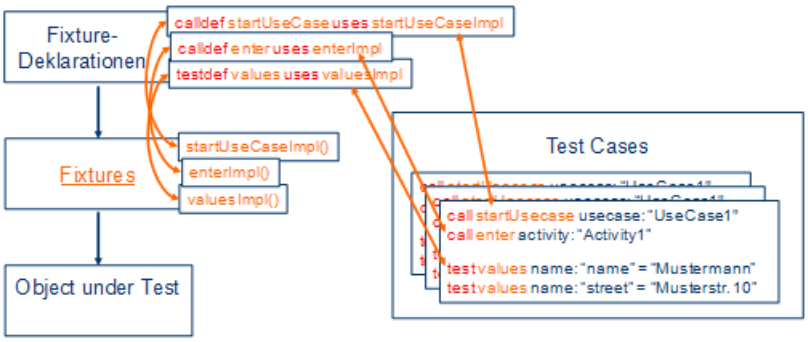

Abbildung 1: Integrity-Architektur

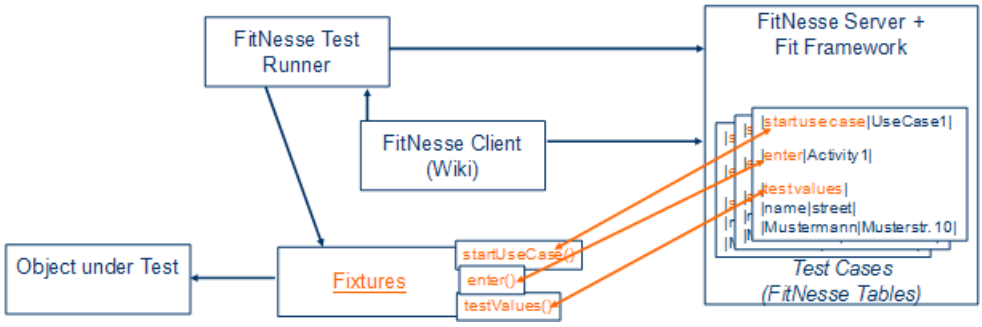

Abbildung 2: FitNesse-Architektur

Inzwischen sind weitere Ausführungsumgebungen verfügbar, die entweder das FIT-Protokoll oder das Slim-Protokoll nutzen. Slim vermeidet einen der großen Nachteile von FIT, das die Testfälle in der ursprünglichen HTML-Tabellenform an die zu implementierenden Fixtures übergibt und HTML-Tabellen als Resultat erwartet - Slim dagegen erhält nur die Testbefehle, so dass sich der Entwickler der Fixtures sich um das Parsen der Testtabellen und die Gestaltung des Ergebnisses nicht selbst kümmern muss. In unseren Projekten nutzen wir das neuere Slim allerdings nicht, da die Implementierung unserer eigenen Fixtures andernfalls komplett portiert hätte werden müssen, ohne dass es einen Mehrwert gegeben hätte.

Analog Integrity stellt auch FitNesse nur die Basis für die Definition und Ausführung von Tests zur Verfügung. Auch hier müssen Fixtures implementiert werden, die die zu testende Anwendung an das FitNesse-Framework binden. Diese sind allerdings global verfügbar, sobald sie sich im Klassenpfad befinden. Es gibt eine Reihe bereits vorhandener Pakete mit Fixtures, die die Arbeit mit FitNesse erleichtern, z.B. für den Zugriff auf Datenbanken, den Test von Swing-Anwendungen oder Webapplikationen, letzteres mit Hilfe von Selenium. Zudem wird Behavior-Driven-Development durch ein Paket mit entsprechenden Fixtures unterstützt [9].

FitNesse teilt sich auf in die Testausführungsumgebung und den FitNesse-Server (Webserver), der ein Wiki für die Testfallerstellung und Dokumentation zu Verfügung stellt.

4 Erfahrungen im Projektalltag

Sowohl FitNesse als auch Integrity werden in unserem Projekten regelmäßig zum Testen eingesetzt. Da beide auch durch eine Softwareschicht mit dem GEBIT-Entwicklungsframework TREND kommunizieren und damit jede mit TREND entwickelte Software testen können, ist der Einsatz wahlweise möglich und so eine gute Vergleichbarkeit gegeben.

4.1 Dokumentation und Erweiterbarkeit

Während eines der großen Mankos von FitNesse die schlechte, zum größten Teil nicht vorhandene Dokumentation des Quellcodes ist, ist der Integrity-Source-Code ausreichend dokumentiert.

Die Erweiterbarkeit ist bei beiden Frameworks gegeben, allerdings ist dies auch essentielle Notwendigkeit, da beide Frameworks nur einfache Testaufgaben out-of-the-Box erledigen können und ansonsten an die zu testende Software gebunden werden müssen. Allerdings ist es bei Integrity im Gegensatz zu FitNesse nicht notwendig, auf die interne Datenstruktur zuzugreifen. FitNesse (unter Verwendung von FIT) gibt für die Implementierung von Testbefehlen Fixtures vor, die auf der internen Repräsentation der HTML-Seiten basieren.

FitNesse erlaubt, diverse Anpassungen auch am Framework, der Testausführung, der Bearbeitung von Tests im Web etc vorzunehmen. In unseren Projekten benutzen wir u.a. eigene Wiki-Widgets (z.B. für Datumseingaben relativ zu aktuellen Datum) und eigene Edit-Responder (z.B. für die einfache Eingabe

von unterstützten Testbefehlen). Integrity beschränkt sich hier auf Anpassungen an die Testausführung, z.B. durch die Implementierung eigener Auswertungsmethoden.

4.2 Testdefinition

Vorteil der Testdefinition unter FitNesse ist, dass auf dem Client keine zusätzliche Software installiert werden muss, da Tests in einem beliebigen Webbrowser erstellt (und ausgeführt) werden können. Dabei muss der FitNesse-Server nicht notwendigerweise auf dem gleichen Rechner laufen. Zudem unterstützt FitNesse eine verteilte Architektur, so dass Tests sowohl lokal als auch auf verschiedenen entfernten Servern gepflegt werden können. Dies haben wir bspw. in Kundenprojekten ausgenutzt, wo Testfälle für den nächtlichen Build auf einem zentralen FitNesse-Server gepflegt wurden, gleichzeitig der Kunde auch eigene Testfälle auf einem lokalen Server erzeugen konnte. Die zentralen Tests wurden dann einerseits in den Build eingebunden, andererseits durch einen automatische Checkin regelmäßig in das Versionskontrollsystem übernommen. So wurde dem Kunden jegliche Verwaltung der Tests abgenommen, er konnte sich also auf die reine Testerstellung konzentrieren. Zudem unterstützt FitNesse prinzipiell weitere (von uns aber wenig bis gar nicht genutzte) Arten der Testdefinition. So können die Tests mit einem einfachen Texteditor oder unter Eclipse editiert werden. Zudem gibt es die Möglichkeit, Testfälle in Excel zu definieren. Syntaxhighlighting und Autovervollständigung werden allerdings nicht out-of-the-Box unterstützt. Um das Editieren zu erleichtern, haben wir die Erweiterbarkeit ausgenutzt und einen eigenen Edit-Responder implementiert, der Templates für unsere eigenen Testbefehle in FitNesse-Testseiten einfügt.

Für Integrity gibt es bisher nur die Möglichkeit, Testfälle in Eclipse zu erstellen und zu editieren (bzw. einen normalen Texteditor zu benutzen). Integrity bringt einen eigenen XText-basierten Editor mit, der Fehler- und Syntaxhighlighting und Autovervollständigung unterstützt, was die Testfallerstellung deutlich erleichtert. Verteiltes Arbeiten wird nur durch die Verwendung eines Versionskontrollsystems unterstützt, was aber aus Eclipse heraus unproblematisch ist.

Sowohl FitNesse als auch Integrity unterstützen Referenzierung und Einbindung von Testfällen und Testsuiten, Testfalltemplates und Testsuite-Hierarchien.

4.3 Testausführung

Die Testausführung in FitNesse erfolgt aus dem Browser heraus. Die Testergebnisse entsprechen dem Testfall inklusive aller Kommentare, der nur um die Testausführungsinformation angereichert wurde.

Integrity kann sowohl ausschließlich in der Konsole laufen als auch in der Testausführungsumgebung. Testergebnisse in Integrity enthalten im Gegensatz zu FitNesse nicht den Test selbst, sondern nur die Beschreibung des jeweilige Tests sowie dessen Ergebnis. Dadurch sind die Testergebnisse deutlich kürzer und auf den ersten Blick besser lesbar. Für die Fehleranalyse ist eine direkte Markierung der genauen Stelle, die im Test fehlgeschlagen ist, wie in FitNesse, jedoch manchmal hilfreich. Von der Testausführungsumgebung in Integrity kann direkt in den Testfall gesprungen werden.

Tests können in FitNesse und in Integrity jederzeit abgebrochen werden, Teilergebnisse des Tests werden auch in diesem Fall angezeigt.

4.4 Debugging

Um FitNesse-Tests zu debuggen, kann der Testrunner verwendet werden, der z.B. aus Eclipse heraus im Debug-Modus gestartet werden kann. Für den Fall, dass man keinen Zugriff auf den Testrunner hat, weil FitNesse auf einem entfernten Server ausgeführt wird, haben wir einen eigenen Debugger in FitNesse integriert, der durch den Testbefehl `debug break` im Laufe eines Tests gestartet werden kann. Dieser bietet die üblichen Möglichkeiten für das Debuggen (schrittweise Test ausführen, zum nächsten Breakpoint springen) sowie eine Inspektion der Anwendung auf dem GUI-Level, allerdings weder Zugriff auf den Code der zu testenden Anwendung noch den Code der Fixtures.

Integrity bietet die Möglichkeit, einen Debugger an den Testlauf zu attachen. Auch hier sind die üblichen Möglichkeiten des Debuggens inklusive des Setzens eines Breakpoints im Testfall verfügbar. Da alles in Eclipse ausgeführt wird, ist der Zugriff auf den Code sowohl der zu testenden Anwendung als auch die Fixtures möglich, wobei auch im Code gesetzte Breakpoints berücksichtigt werden.

4.5 Integrationstests zwischen verschiedenen Systemen

FitNesse unterstützt erst einmal prinzipiell den Test nur einer Anwendung zu einer Zeit. Durch eigene Erweiterungen ist allerdings auch der Test von mehreren Anwendungen möglich.

Integrity ist von Haus aus dazu ausgelegt, mehr als eine Anwendung gleichzeitig zu testen, da eines der ersten Anwendungsgebiete der Test einer Kassenanwendung inklusive Backoffice und Peripherie war.

4.6 Plattformunabhängigkeit

Da beide Framework in Java implementiert sind, ist eine Betriebsystemunabhängigkeit gegeben.

Integrity kann im Augenblick nur Tests für die Java-Plattform ausführen. Erste Experimente haben eine prinzipielle Machbarkeit des Tests von .net-Anwendungen gezeigt, es gibt jedoch noch keine umfängliche Lösung. Andere Plattformen sind im Augenblick nicht geplant.

FitNesse kann durch die Verwendung eines anderen Testrunners auf verschiedenen Plattformen laufen. Dabei werden die Fixtures in der jeweiligen Sprache geschrieben. Verwendet haben wir bereits den mitgelieferten FitNesse-dotnet-Testrunner für den Test von .net-Anwendungen, darüber hinaus sind Testrunner für diverse weitere Plattformen (Ruby, PHP, Python, Smalltalk), meist als Slim-Implementierungen, verfügbar.

Durch die Anbindung der Selenium-Ausführungsengine an FitNesse oder Integrity sind beide Frameworks in der Lage, Webanwendungen zu testen.

4.7 Lizenz

Eines der großen Probleme beim Einsatz von FitNesse ist die Veröffentlichung unter der GNU GPL. Diese erlaubt die Weitergabe von Erweiterungen und Ergänzungen des Codes nur unter der Voraussetzung, dass dies ebenfalls wieder unter der GNU GPL veröffentlicht werden (*Copy Left*). Dies kann für die Einsatz in Unternehmen problematisch sein, sofern nicht entsprechende Maßnahmen ergriffen werden, z.B. das der selbst implementierte Code (Fixtures) komplett getrennt von der zu testenden Anwendung ist oder dass eine Weitergabe von FitNesse an andere (also auch Kunden) nicht erfolgt.

Integrity dagegen ist unter der EPL veröffentlicht. Dabei müssen zwar Modifikation des ursprünglichen Codes ebenfalls unter EPL gestellt werden, Erweiterungen wie die implementierten Fixtures können jedoch auch unter einer anderen Lizenz (nicht notwendigerweise Open-Source-Lizenz) vertrieben werden.

4.8 Akzeptanz von Entwicklerseite

FitNesse wird zwar in unseren Projekten eingesetzt, allerdings war die Akzeptanz von Entwicklerseite schon immer gering. Gerade elementare Funktionen wie das Syntaxhighlighting oder die Autovervollständigung waren die schwerwiegendsten Kritikpunkte neben der schlechten Dokumentation des FitNesse-Codes, der eigene Erweiterungen schwierig machte.

Integrity wurde u.a. auch entwickelt, um die Nachteile von FitNesse auszugleichen. Es wurde ein besonders Augenmerk auf Erweiterbarkeit, Verständlichkeit und Flexibilität gelegt. Obwohl Integrity erst seit etwa zwei Jahren überhaupt in Projekten eingesetzt wurde und in größerem Umfang erst seit dem letzten Jahr Verwendung findet, ist die Akzeptanz der Entwickler gut. In Projekten, in denen sowohl FitNesse als auch Integrity eingesetzt werden, wird für die Neudefinition von Testfällen meist Integrity verwendet.

4.9 Akzeptanz von Kundenseite

Für die Kunden ist der Vorteil des Einsatz von FitNesse, dass keine neue Software installiert werden muss, wenn der FitNesse-Server von uns gehostet wird. Bei Integrity muss sowohl Eclipse als auch das Integrity-Plugin installiert werden.

Von der Testfallerstellung unterscheiden sich die beiden wenig, da in beiden Fälle eine neue Testsprache gelernt werden muss. Durch die Fixture-Deklaration in Integrity statt des direkten Zugriffs auf die implementierten Fixtures aus dem Test heraus ist es aber deutlich einfacher, Kundenwünsche zur Benennung dieser Testbefehle umzusetzen, da nur die Deklaration geändert oder ergänzt werden muss. So ist bspw. Mehrsprachigkeit kein Problem.

5 Zusammenfassung und Ausblick

Integrity ist ein neues Testframework, das eine gute Alternative zu FitNesse bildet. Beide Frameworks sind als Open-Source verfügbar und damit erweiterbar und an eigene Bedürfnisse anpassbar. Ob ein Einsatz des einen oder anderen Frameworks lohnt, ist auch eine Frage des jeweilige Kontexts. Ist eher Plattformunabhängigkeit, die zentrale Erfassung der Testfälle oder die einfache Verteilung der Testumgebung an die Tester die Priorität, dann sollte FitNesse verwendet werden. Wird dagegen ein Testframework gebraucht, das Java-Anwendungen testen soll, sich dabei möglichst gut in die vorhandene Entwicklungsumgebung einfügt und zudem den Integrationstest verschiedener Systeme ermöglichen soll, ist Integrity die bessere Wahl.

Literatur

[1] Cucumber.
http://cukes.info/.

[2] FitNesse Test Framework.
http://www.fitnesse.org/.

[3] GEBIT Solutions GmbH. Integrity Test Framework.
http://www.integrity-tf.org/.

[4] JBehave.
http://jbehave.org/.

[5] Quality First Software. QF-Test.
http://www.qfs.de/.

[6] Selenium - Web Browser Automation.
http://docs.seleniumhq.org/.

[7] Timothy Wall. Abbot Java GUI Test Framework.
http://abbot.sourceforge.net/.

[8] Ward Cunningham. Fit: Framework for Integrated Test.
http://fit.c2.com/.

[9] Wes Williams. GivWenZen.
https://github.com/weswilliams/GivWenZen.