Acceptance Test Refactoring

Heiko Ordelt, Frank Maurer

University of Calgary
Department of Computer Science
2500 University Dr. NW
Calgary, Alberta T2N 1N4 Canada
hordelt@gmail.com, maurer@cpsc.ucalgary.ca

Abstract. In Executable Acceptance Test Driven Development, acceptance tests represent the requirements of a software system. As requirements change over time, the acceptance tests have to be updated and maintained. This process can be time-consuming and risky as acceptance tests lack the regression safety net that production code has. Refactoring of acceptance tests is used to keep the fixtures and the acceptance test definitions consistent.

Keywords: Refactoring, Executable Acceptance Test Driven Development (EATDD), Story Test, Acceptance Test, Fit

1 Refactoring of Acceptance Tests

As Andrea discusses [1], whenever production code is refactored, unit tests can be used to check whether the system's behavior is still unchanged. Acceptance tests lack such an important regression safety net [1]. Changes to acceptance tests have to be made safely to minimize the risk of an unwanted behavior change. Additionally, the fixture code has to be kept consistent with the test definition which is time-consuming and error-prone. Our goal is to allow the user to carry out changes safely and to keep the test definition and the corresponding fixture aligned. We distinguish between two different kinds of acceptance test refactorings:

- Behavior preserving: The behavior specified by the acceptance test is not changed by the refactoring and there is no user interaction needed to make the refactored test pass.
- Behavior changing: The behavior specified by the acceptance test is changed by the refactoring and the user is required to manually update the fixture and/or system-under-test code to make the refactored test pass.

However, the refactoring has to result in a successful compilation of the fixture code and in an executable acceptance test.

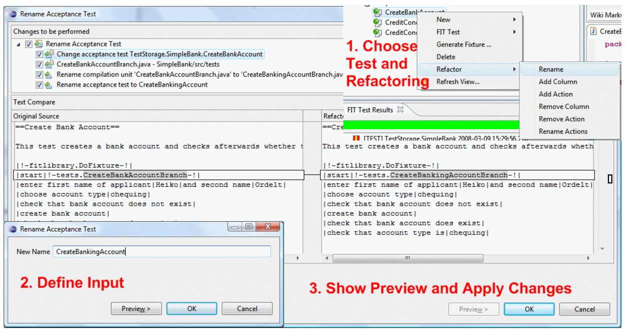
2 Related Work and Existing Tools

Acceptance test refactoring has been discussed by Andrea [6] who refactored an acceptance test to simplify its structure and improved the readability. Furthermore [1], she mentioned that tool support for acceptance test refactoring is an important feature that the next generation of functional testing tools must support. There are several Functional Testing Development tools available that support Executable Acceptance Test Driven Development like FitNesse [8], AutAT [2], ConFIT [3] and FITpro [4], GreenPepper [7]. These tools and to our best knowledge no other tool supports acceptance test refactoring yet.

3 Implementation

We extended the open-source acceptance test IDE FitClipse [5] to support acceptance test refactoring by using the Eclipse refactoring framework (see Figure 1). FitClipse currently supports the following refactorings:

- Rename acceptance test (ColumnFixture, DoFixture) – behavior preserving
- Add/Remove column (ColumnFixture) – behavior changing
- Add/Remove/Rename action (DoFixture) – behavior preserving



Rename Acceptance Test
- Change acceptance test TestStorage.SimpleBank.CreateBankAccount
- CreateBankAccountBranch.java - SimpleBank/src/tests
- Rename compilation unit 'CreateBankAccountBranch.java' to 'CreatingBankingAccountBranch.jav
- Rename acceptance test to CreateBankingAccount

```
==Create Bank Account==

This test creates a bank account and checks afterwards whether t

|!-fitlibrary.DoFixture-!|
|start|!-tests.CreateBankAccountBranch-!|
|enter first name of applicant|Heiko|and second name|Ordelt|
|choose account type|chequing|
|check that bank account does not exist|
|create bank account|
```

```
This test creates a bank account and checks afterwards wheth

|!-fitlibrary.DoFixture-!|
|start|!-tests.CreateBankingAccountBranch-!|
|enter first name of applicant|Heiko|and second name|Ordelt|
|choose account type|chequing|
|check that bank account does not exist|
|create bank account|
|check that bank account does exist|
|check that account type is|chequing|
```

[TEST] TestStorage.SimpleBank 2008-03-09 15:29:56.

New Name: CreateBankingAccount

Fig. 1. FitClipse Refactoring Support

References

1. Andrea, J.: Envisioning the Next Generation of Functional Testing Tools, IEEECS, 2007
2. AutAT, http://boss.bekk.no/boss/autat
3. ConFIT, http://bandxi.com/fitnesse/confit.html
4. FITpro, http://www.luxoft.com/fit
5. FitClipse, http://ase.cpsc.ucalgary.ca/index.php/FitClipse/FitClipse
6. Andrea, J.: Brushing Up On Functional Test Effectiveness, http://www.stickyminds.com/s.asp?F=S9937_ART_2 (last accessed: 03/06/2008)
7. GreenPepper, http://greenpeppersoftware.com/en/products
8. FitNesse, http://fitnesse.org