Refactoring of Acceptance Tests

15.04.2008

Master-Kolloquium

Heiko Ordelt
Extreme Programming (XP)

- Agile Method following the Agile Manifesto
- Software development process which focuses on problem solving rather than formalism
- Two kinds of testing
  - Unit Testing
  - Acceptance Testing
- Core practice Test Driven Development
  - Unit Test Driven Development
  - Executable Acceptance Test Driven Development
Executable Acceptance Test Driven Development (EATDD)

- An acceptance test describes a feature and serves as requirement specification
- Created in collaboration of customer and developers
- Software development is driven by executable acceptance tests
- „Test-First“: Development of a feature starts when the corresponding acceptance test has been created
Tools supporting EATDD

- **FIT**
  - Framework to define and execute acceptance tests
  - Additional fixture support by FitLibrary

- **FitNesse**
  - Multi user wiki system to manage and execute acceptance tests
  - Tests are defined in wiki syntax defined by FitNesse
FIT Fixtures

- Link between test definition and system under test
- Commonly used fixtures
  - ColumnFixture (Testing of parametrized input/output)
  - DoFixture (Testing of workflows)
Acceptance Tests as Requirements Specification

- Acceptance tests specify system behaviour
- Modification of acceptance tests needed when requirements are changed
Effectiveness of Acceptance Tests

- Test definition defines the effectiveness of acceptance tests
- Characteristics by Jennitta Andrea 2007
  - Declarative
    - Tests should be written in the language of the business domain rather than in a language of user interface elements
  - Succinct
    - Short tests are easier and quicker to understand than large test definitions
  - Autonomous
    - Acceptance tests should be self-contained and runnable without dependencies on other tests
  - Sufficient
    - Only testing of the main business processes and the most important test cases
  - Locatable
    - Related tests should be easy to find to serve as requirements specification
Modification of Acceptance Tests

- Modifications are needed
  - to reflect changed requirements
  - to improve the effectiveness
- Manual test modification has disadvantages as follows
  - time-consuming
  - error-prone
  - No regression safety-net like source code
Refactoring von Akzeptanztests

- Modification of the test definition of an acceptance test
- Keeps test definition and fixture code consistent
- Behaviour of the refactored test can be changed by a refactoring
- Execution of a refactored test must result in either red or green
- Automated refactoring tool support can lower the test maintenance effort in the same way as code refactoring tools lower it for source code
Refactoring Catalogue

- Created by analyzing the relationships between test definition and fixture of the FIT framework
- Rename Acceptance Test
- ColumnFixture
  - Add Column
  - Remove Column
- DoFixture
  - Add Action
  - Remove Action
  - Rename Action
FitClipse

- IDE for managing acceptance tests
- Integrated in Eclipse as plug-in with its own perspective
Automated Refactoring Implementation

- Extension of FitClipse for automated refactoring support
- Every refactoring of catalogue implemented
- Used Eclipse plug-ins
  - Java Development Tools (JDT)
  - Language Toolkit (LTK)
- Integrated in the commonly known Eclipse refactoring workflow
- Supports multiple fixtures for multi-modal test execution
- Supports multiple test table in one test definition
Automated Refactoring Components

- **TestDefinitionParser**
  - Analyses test definition and extracts elements like tables, rows and cells

- **FixtureParser**
  - Analyses and modifies Java fixtures, e.g. adding methods or fields etc.

- **RefactoringTest**
  - Encapsulates a test and contains methods needed by the various refactoring types

- **RefactoringTestFactory**
  - Automatically detects the type of a test (Do- or ColumnFixture) and creates the proper RefactoringTest object
Refactoring Process

1. Choose Test and Refactoring

2. Define Input

3. Show Preview and Apply Changes
Developed by following EATDD thus acceptance testing required

Eclipse plug-ins not testable by FIT framework

Extension built with PDE Junit as link between fixture and system
Problems / Contributions

**Problems**
- Less existing academic work
- Understanding of Eclipse framework
- Testing the refactoring extension

**Contributions**
- Academic contribution
- Automated refactoring support of FitClipse
- Integration of the Eclipse refactoring framework making extending refactoring or fixture support easier
Future Work

- Evaluation whether it is beneficial for following the EATDD process
- Support of more fixtures and refactoring types
- Support of HTML as test definition format
- Keeping a history of applied refactoring types
- Refactoring from Java file to test definition
- Improving the test structure of the refactoring extension
- Several test table of different types in one test definition
Thanks
Thanks for your attention

Rocky Mountains  
22.09.2007