Developing Acceptance Tests from Existing Documentation Using Annotations: An Experiment

AST '09

David Connolly, Frank Keenan and Fergal Mc Caffery

This research is supported by the Institutes of Technology Ireland's Technological Sector Research: STRAND I Post-Graduate R&D Skills Programme.

Software Technology Research Centre
Dundalk Institute of Technology, Ireland

Overview

- Introduction:
  - Agile
  - Acceptance Test Driven Development (ATDD)
  - The Framework for Integrated Test (FIT)
- Problem Background
- Research Question
- Experiment Description
- Experiment Results
- Progress to Date
- Concluding Remarks

Agile

- Agile Manifesto (K. Beck et. al. / www.agilemanifesto.org)

  Individuals and interactions over processes and tools
  Working software over comprehensive documentation
  Customer collaboration over contract negotiation
  Responding to change over following a plan

- Agile Methodologies
  - eXtreme Programming (XP)
  - Scrum

eXtreme Programming

- 12 inter-linked practices (K. Beck, 2005)
- Three interact with this project:
  - Testing
    Unit & Acceptance Tests are written before coding
  - Continuous Integration
    System is built regularly with tests executing automatically.
  - Whole Team
    Customer or Customer Proxy deeply involved with team

Acceptance Test Driven Development (ATDD)

- Acceptance Tests are black-box system tests
- Customer involvement through whole team practice
- ATs are linked to User Stories.
- ATDD extends parts of Unit Testing/TDD increasing customer focus but not conflicting with testing practice
- Executable, Automated- allows Continuous Integration Variants include:
- Mugridge- Story Test Driven Development
- Maurer et.al. Executable Acceptance Driven Development

Framework for Integrated Test (FIT)

- A leading open source ATDD Tool in common use on agile projects with multiple programming languages.
- FIT uses data represented in tables to drive fixture code which runs against system code.
- Also a Wiki user interface available called FitNesse.
- FitNesse works with traditional non-web projects.

Example FIT Table

WebShop.ValidCard		
Card Number	**Expiry Date**	**Valid?**
5500 0000 0000 0004	09 2009	TRUE
4111 1111 1111 1111	09 2009	TRUE
4111 1111 1111 1112	09 2009	FALSE
5500 0000 0000 0004	03 2002	FALSE

Problem Background

- Calls for tools which support multiple formats "textual, tabular, storyboard, graphical, or multi-modal" (J. Andrea, 2007)
- ATDD not well supported by tools from Customers' point of view.
- FIT Tests are nearly always written by developers from customer descriptions or documentation.

Research Question

- *To what extent can Acceptance Test Driven Development be improved by supporting the elicitation of executable acceptance tests from existing text?*

These sub-questions are related to this question:

- Is current acceptance test authoring process hampered by existing practices, which often mean the customer is limited to writing descriptions?
- Is there a loss of fidelity when developers translate customer descriptions to FIT tables?

Experiment: Participants

- Six participants
- Participants:
  - Randomly assigned to Group A or B
  - Worked independently
- Given:
  - Four Questions (ATs for a package management system)
  - Alternating Annotated & Non-Annotated
- Task:
  - Write FIT Tables from descriptions given in Questions.

Experiment: Task

- Group A – Started with non-annotated version
- Group B – Started with annotated version

PackageManagement. PackageManagementDescriptions.

BootstrapBaseSystem

To Bootstrap the Base System on first installation, a bootstrap package manager? installs a list of packages with out calculating there dependencies.

The list of packages includes libc$^{\Sigma}$ and baselayout$^{\Sigma}$ for the basic system, sh$^{\Sigma}$ for users to interact with the system and tar$^{\Sigma}$ and gz$^{\Sigma}$ to support the package-manager$^{\Sigma}$. Each installation must succeed$^{\bullet}$. These packages taken together form a package collection named "system".

Any attempt to install a package outside of the "system" collection will fail$^{\bullet}$. Such as firefox$^{\Sigma}$, from the web collection.

Verification of the system collection$^{\Sigma}$ of the packages with the package manager? will fail$^{\bullet}$ before any packages have been installed and will continue to fail$^{\bullet}$ until all packages from the system collection are installed^{0}.

After the completion of the bootstrap, the package-manager verification of the system collection$^{\Sigma}$ of packages must pass$^{\bullet}$.

Annotated

PackageManagement. PackageManagementDescriptions.

BootstrapBaseSystem

To Bootstrap the Base System on first installation, a bootstrap package manager installs a list of packages with out calculating there dependencies.

The list of packages includes libc and baselayout for the basic system, sh for users to interact with the system and tar and gz to support the package-manager. Each installation must succeed. These packages taken together form a package collection named "system".

Any attempt to install a package outside of the "system" collection will fail. Such as firefox, from the web collection.

Verification of the system collection of the packages with the package manager will fail before any packages have been installed and will continue to fail until all packages from the system collection are installed.

After the completion of the bootstrap, the package-manager verification of the system collection of packages must pass.

Non-Annotated

Experiment: Annotations in Use

- These annotations are based on elements of an acceptance test description recommended by N. Jain

Annotation Legend

Given Precondition[0]
When Actor + Action[?]
Then Observable Result[⊗]

Example[Σ]

Experiment: Design

- Metrics:
  - Errors
  - Correct Elements
  - Missing Elements
  - Time
- Used to Measure:
  - Over-Specification (Errors)
  - Under-Specification (Missing Elements)

Experiment: Results Overview

- In all cases *Errors* were reduced with Annotations
- On average had more *Correct Elements* and fewer *Missing Elements*.
- Impact of *Time* taken need further study
- New type of annotation needed to visualise groupings

Progress to Date: Tool Implementation

- Tool implementation in progress:
  - Google Web Toolkit for User Interface, DOM Manipulation and RPC/Web Services.
  - Previously investigated using ATLAS Transformation Language to transform between ECore models of Annotations and FIT Tables (also generate fixture code).
  - Currently proposing to have new Fixture type work directly with Annotated document. Generating FIT tables for discussion only.

Progress to Date: Other

- Investigation into authoring of User Stories from existing documentation. Poster, appeared at ECBS '08 (Belfast, Northern Ireland)
- Initial outline of prototype tool. Appeared at *Research in Progress* session of Agile '08 (Toronto, Canada)
- Initial outline of a modeled approach to translating from annotations to fixture code & FIT Tables. Appeared at Doctoral Symposium of EuroSPI² (Dublin, Ireland).

Concluding Remarks

- Tool implementation and planning of experiments continuing this summer.
- Further experiments continuing into early 2010 With larger groups including undergraduates
- Planning a Case Study involving prototype with a company/team adopting ATDD & FIT.

Thanks for Listening!

- Questions?

- For more information please contact:

David Connolly
- david.connolly@dkit.ie

Supervisors:

Frank Keenan
- frank.keenan@dkit.ie

Fergal Mc Caffery
- fergal.mccaffery@dkit.ie

This research is supported by the Institutes of Technology Ireland's Technological Sector Research: STRAND I Post-Graduate R&D Skills Programme.