Improving Agile with Acceptance Test Driven Development

Course Summary

Description

A fundamental axiom of the Agile Manifesto is that “business people and developers must work together daily throughout the project.” The framers of the Agile Manifesto were all seasoned developers who understood how to structure these regular interactions with the client to produce software that met the clients’ requirements in the most efficient way possible, but these patterns of interaction – the descriptions of what happens in these daily collaborations – was left up to the various flavors of Agile to manage in their own unique ways.

Over the years various Agile gurus tried to tried to document and describe the best practices used by the experts to manage these business and developer interactions in the most productive, efficient and productive ways possible. A variety of approaches have been developed in the Agile community including Dan North’s Behavior Driven Design (BDD), Gojko Adzic’s Specification to try and capture these best practices. ATDD synthesizes this work with the automation concepts from Ken beck’s Test Driven Development which are supported with open source tools like Gherkin and Cucumber.

The course starts with a critical analysis the Agile processes with a focus on identifying the root causes of inefficiencies and the various types of problems that occur during an Agile development project. This is followed by an overview of the ATDD process and how it specifically targets the root causes of these problems.

Students are then led on a detailed walkthrough of the ATDD process emphasizing its iterative nature, the development of high quality acceptance tests as a driver of collaborative work and how the four phases of the ATDD cycle – Discuss, Distill, Develop and Demo – integrate with the other practices and disciplines common to Agile development, such as TDD, model based design, and iterative development.

Using a worked example and corresponding lab exercises, students work through the details of each of the phases, exploring the techniques and developing the artifacts appropriate at that point such examples with acceptance criteria and a domain specific language in the Discuss phase, executable acceptance tests in the Distill phase and derived component unit tests in the development phase.

During the examples and the hands on lab exercises, the Gherkin specification language is used to create acceptance tests for automated execution. Students are taught how to use all of the language constructs and how to follow the best practices involved in writing Gherkin tests such as when and how to use scenario outlines, data tables, tags and other features.

The course concludes with an examination of how the ATDD process integrates with various standard Agile process models and other current trends like microservices and devops. There is also a discussion of metrics that can be used to evaluate and improve an Agile team’s implementation of ATDD.

All of the theory presented is reinforced with examples, real world case studies, hands-on exercises, and projects.

Improving Agile with Acceptance Test Driven Development

Course Summary (cont'd)

Objectives

After taking this course, students will be able to:

- Identify root causes of failures and inefficiencies in Agile projects.
- Describe the ATDD process phases and how ATDD activities address the root causes of failure.
- Describe how the ATDD process fits into the standard Agile iterative module.
- Develop robust specifications by example with acceptance criteria during the Discuss phase.
- Turn examples in to efficient and high quality acceptance tests during the Distill phase.
- Describe and use the metrics to evaluate the quality of the acceptance tests.
- Write acceptance tests in Gherkin using the full set of features offered by the language.
- Develop a domain specific language appropriate for the project.
- Write acceptance in a declarative form and describe why we use declarative instead of imperative forms for the tests.
- Use Gherkin features, backgrounds scenarios, scenario outlines, tags and data tables correctly.
- Describe Gherkin best practices.
- Describe how to Integrate ATDD with their Agile process
- Describe how ATDD supporting Agile artifacts like backlogs and spikes
- How to integrating Agile deliverables and ATDD artifacts
- How to support development by having ATDD tests feed into the Test Driven Development process.

Topics

- Why ATDD?
- Tests as Drivers of Development
- The ATDD Process
- The Discussion Phase
- The Distill Phase
- Gherkin Basics
- More Gherkin
- The Development Phase
- The Demo Phase
- Best Practices
- Summary

Audience

The course is appropriate anyone who is involved in or responsible for an Agile development project or process.

Prerequisites

The course has no prerequisites other than a familiarity with Agile processes.

Duration

Two days

Improving Agile with Acceptance Test Driven Development

Course Outline

I. Why ATDD?
- A. Root causes of software project failures
- B. Errors, faults, and failures
- C. How ATDD addresses the root causes
- D. Filling the gaps in Agile with ATDD
- E. How ATDD supports efficient development
- F. Effective software, efficient development, validation, and verification

II. Tests as Drivers of Development
- A. Unit testing and integration testing
- B. The importance of good tests
- C. The Agile testing quadrants
- D. Test automation in the four quadrants
- E. Supporting Test Driven Development with ATDD
- F. Acceptance tests as a system specification
- G. Interfaces, implementations, and functional testing

III. The ATDD Process
- A. Iterative and incremental: Red, Green, Go
- B. The four phases: Discuss, Distill, Develop, Demo
- C. Deliverables for each phase
- D. The deliverable: Gherkin acceptance tests
- E. Integrating the ATDD phases with Agile iterations and sprints
- F. Backlogs, grooming and ATDD artifacts
- G. Incorporating ATDD from the project management perspective

IV. The Discussion Phase
- A. Collaborating to develop examples
- B. Identifying acceptance criteria for examples
- C. Iterative development of the examples
- D. Examples, user stories, requirements, and features
- E. Moving from examples to acceptance tests for features
- F. Developing a Domain Specific Language (DSL)

V. The Distill Phase
- A. Robustness and correctness – drilling down into the examples
- B. Coverage and correctness
- C. Developing optimal sets of acceptance tests
- D. Exploring invalid cases, edge cases and outliers
- E. Using the DSL to write the cases
- F. Declarative versus imperative acceptance tests
- G. Organizing the tests by features
- H. Feeding back into the discussion phase

VI. Gherkin Basics
- A. Gherkin as a structured acceptance test language
- B. Features and feature files
- C. Gherkin scenarios as test cases
- D. The Given, When, Then syntax for scenario steps
- E. The Background section
- F. Using And and But for readability
- G. How Cucumber executes a feature file
- H. Comments and feature file documentation

VII. More Gherkin
- A. Using Scenario Outlines and Example Tables
- B. Organizing complex test data with Data Tables
- C. Well-structured Gherkin Scenarios
- D. Best practices in organizing Scenarios and Steps
- E. Best practices for using data in a scenario step

Improving Agile with Acceptance Test Driven Development

Course Outline (cont'd)

F. When to use Scenarios, Backgrounds, and Outlines
G. Using Tagging for selective test execution

VIII. The Development Phase
A. Planning the incremental addition of acceptance tests
B. Supporting the coding and build processes
C. Selective execution of Scenarios with tags
D. What failing tests tell us
E. Using ATDD for continuous regression testing
F. Maintaining the acceptance tests as system specification

IX. The Demo Phase
A. Validation of the system functionality
B. Acceptance tests as a basis for interface development
C. Understanding and using the results of the Demo phase

X. Best Practices
A. Review of best practices by ATDD phase
B. Introducing ATDD into an Agile process
C. Recommended reviews and quality checklists
D. Using ATDD to refactor the development process
E. Using ATDD to improve efficiency of project communications
F. Typical pitfalls, problems, roadblocks, and strategies for resolution

XI. Summary
A. Review of the course
B. How to move forward and apply what has been learned
C. Other topics requested by students
D. Assessment of knowledge and skills gained