NetObjectives
ESSENTIAL WHITE PAPERS

Introduction to Acceptance Test-Driven Development

by Ken Pugh

Introduction to Acceptance Test-Driven Development

A Net Objectives Essential White Paper

Net Objectives Press, a division of Net Objectives

1037 NE 65th Street Suite #362

Seattle, WA 98115-6655

404-593-8375

Find us on the Web at: *www.netobjectives.com*

To report errors, please send a note to *info@netobjectives.com*

Cover design by Andrea Bain

Published by Net Objectives, Inc.

10 9 8 7 6 5 4 3 2

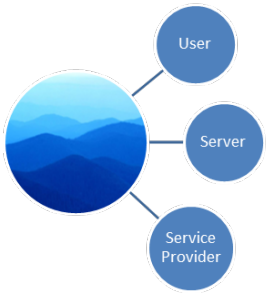
Printed in the United States of America

SUMMARY

In this paper, I show how acceptance test-driven development helps with communication between the business customers, the developers, and the testers. We cover the 5 W's - what are acceptance tests, who creates them, when they should be created, where they are used, and why you should use them. This material is adopted from *Lean-Agile Acceptance Test-Driven Development: Better Software Through Collaboration* by Ken Pugh

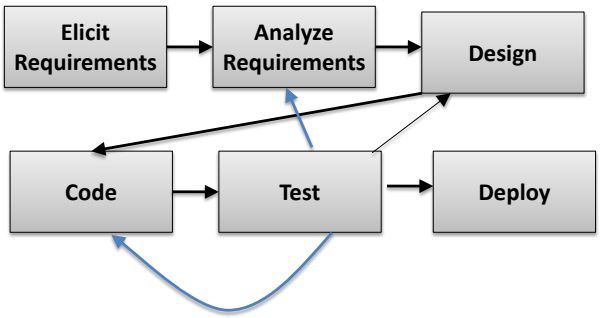
WHAT ARE ACCEPTANCE TESTS

Acceptance tests are from the user's point of view – the external view of the system. They examine externally visible effects, such as specifying the correct output of a system given a particular input. Acceptance tests can show how the state of something changes, such as an order that goes from "paid" to "shipped". They also can specify the interactions with interfaces of other systems. In general, they are implementation independent, although automation of them may not be. In addition, acceptance tests can suggest that operations normally hidden from direct testing (such as business rules) should be exposed for testing.



WHY DO ATDD

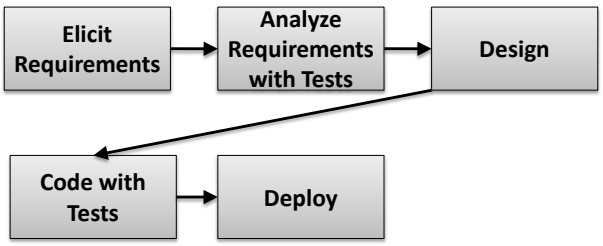
Teams that follow the acceptance test-driven process have experienced efficiency gains. In one case, rework went down from 60% to 20%. This meant that productivity doubled since the time available for developing new features went from 40% to 80%. In another case, the workflows were working the first time the system was brought up. Getting the business rules right, as in an example we shall see, prevents rework. Because the business customer, developer, and tester are involved in acceptance test creation, there is tighter cross-functional team integration. In addition, passing the acceptance tests visibly demonstrates that the story is complete.



WHEN ARE ACCEPTANCE TESTS CREATED

The value stream map for classical development is shown below. After eliciting requirements, they are analyzed. A design is created and code developed. Then the system is tested. You can notice many loops go back from test to analysis, design, and coding. These loop backs cause delay and loss of productivity. Why do these occur?

Frequently the cause is misconstructions. In particular, it is misunderstanding the requirements. The loop backs are really feedback to correct these mistakes. There will always be a need for feedback, but quick feedback is better than slow feedback.



Ken Pugh is a fellow consultant with Net Objectives (www.netobjectives.com). He helps companies transform into lean-agility through training and coaching. His particular interests are in communication (particularly effectively communicating requirements), delivering business value, and using lean principles to deliver high quality quickly. He also trains, mentors, and testifies on technology topics ranging from object-oriented design to Linux/Unix. He has written several programming books, including the 2006 Jolt Award winner, Prefactoring. His latest books are *Lean-Agile Acceptance Test Driven Development: Better Software Through Collaboration.* and *Essential Skills for The Agile Developer.* He has helped clients from London to Boston to Sydney to Beijing to Hyderabad. When not computing, he enjoys snowboarding, windsurfing, biking, and hiking the Appalachian Trail.

As you can notice in the revised value stream map below, the acceptance tests are created when the requirements are analyzed. The developers then code using the acceptance tests. A failing test provides quick feedback that the implementation is not meeting the requirements.

WHO AUTHORS THE TESTS

The tests are authored by the triad – the customer, tester, and developer. At least one example used in the tests should be created by the customer working with the tester and developer. The tester and developer can then create more and have them reviewed by the customer.

The developers connect the test to the system by writing short bits of code. Anyone – customers, developers or testers can run the tests. Acceptance tests can be manual. However, automating acceptance tests allows them to run as regression tests to ensure that new features do not interfere with previously developed features.

Acceptance tests are not a substitute for interactive communication between the members of the triad. However, they provide focus for that communication. The tests are specified in business domain terms. The terms then form a common language that is shared between the customers, developers, and testers.

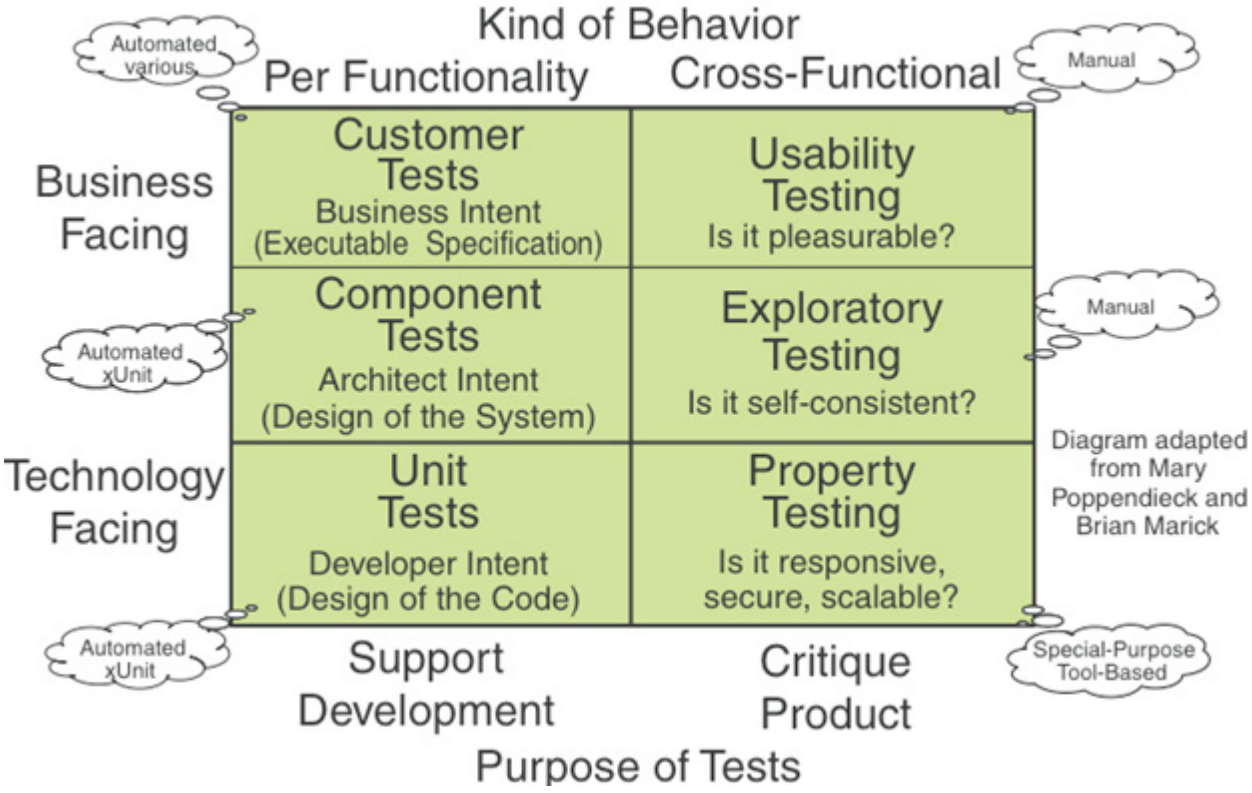
HOW DO ACCEPTANCE TESTS FIT INTO THE OVERALL TESTING STRATEGY?

Acceptance tests are only a part of the overall testing strategy. They are the customer tests that demonstrate the business intent of a system, as shown in the upper left box the figure below. The component tests which are beneath them, are technical acceptance tests developed by the architect that help specify the behavior of large modules. Unit tests, the lowest left box, are partially derived from acceptance and component tests, and help the developer to create easy-to-maintain code. The tests on the right – usability, exploratory, and property – examine what are often termed the non-functional aspects. They also need to pass to have a high quality system.

ACCEPTANCE TEST EXAMPLE

Suppose you had a requirement which states:

> ***As the marketing manager, I want to give discounts to repeat customers so that I can increase repeat business.***



There is one detail for this story. The customer discount is computed according to business rule Customer Discount Rule. The details of this rule are:

If Customer Rating is Good and the Order Total is less than or equal $10.00,

Then do not give a discount,

Otherwise give a 1% discount.

If Customer Rating is Excellent,

Then give a discount of 1% for any order.

If the Order Total is greater than $50.00,

Then give a discount of 5%.

Now read the rule again and answer one question. For a customer who is Good and has an order of $50.01, what should the discount be?

Depending on how you read the rule, you may come up with one, five, or six percent. The rule is ambiguous. How do we make it clearer? The customer, developer, and tester come up with some examples, which turn into tests.

Suppose they come up with a table of examples, as shown below. In third set of values, the discount for the case in question should be 1 percent. That's what the business customer wanted. Imagine if the customer had not been consulted and if both the tester and developer had thought it should be 6 percent.

Now these examples are used as acceptance tests. These tests and the requirement are tied together – the tests help clarify the requirement and the requirement forms the context for the tests.

Discount		
Order Total	**Customer Rating**	**Discount Pct**
10.00	Good	0
10.01	Good	1
50.01	Good	1
0.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

WHERE ACCEPTANCE TESTS ARE IMPLEMENTED

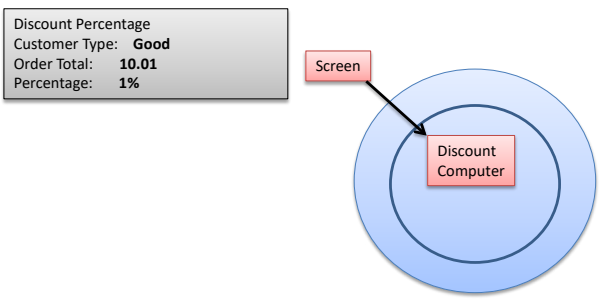
There are at least four ways to implement the tests. They are a testing script, which uses the user interface; a graphical or command line interface; a unit testing framework; and an acceptance testing framework. Let's take a brief look at each.

In the first case, the tester creates a testing script. For example, they logon as a Good customer, start up an order, and put items into it. When the order total is $10.01, they complete the order and make sure that it shows a $.10 discount. Now they repeat the process for the five other cases, increasing the possibility of carpal tunnel syndrome.

This script needs to be run at least once in order to ensure the discount is computed properly as part of the workflow. However there are three other ways to check for the other cases.

A graphical or command line interface could be created that accessed the module that computed the discount. The tester need only enter the customer rating and order total to determine if the discount is correctly computed.

The developer could create an Xunit test, as shown below. This automates the testing process. However, since the test is in the language of the developer, it can be more difficult to use as a communication vehicle and to ensure that changes in the business rule have been incorporated into the test.



```
class TestCase {
    testDiscountPercentageForCustomer() {
        SomeClass o = new SomeClass();
        assertEquals(0,
            o.computeDiscount(10.0, Good));
        assertEquals(1,
            o.computeDiscount(10.01, Good));
        assertEquals(1,
            o.computeDiscount(50.01, Good));
        assertEquals(1,
            o.computeDiscount(.01,
            Excellent));
        assertEquals(1,
            o.computeDiscount(50.0,
            Excellent));
        assertEquals(5,
            o.computeDiscount(50.01,
            Excellent));
        }
 }
```

One could use an acceptance test framework, which allows the tests to be readable by the customer. The table shown here is from Fit – Framework for Integrated Testing -- but other frameworks, such as Cucumber and Robot Framework have similar tables.

Discount		
Order Total	**Customer Rating**	**Discount Pct**
10.00	Good	0
10.01	Good	1
50.01	Good	1
0.01	Excellent	1
50.00	Excellent	1
50.01	Excellent	5

The table becomes a test and is tied to the underlying system through glue code called a fixture. When the test is run, the results appear in the table – green is a pass, red is a fail.

ATDD FROM THE START

The acceptance test process actually begins at the definition of a feature or capability. For example, the user story about offering discounts is part of marketing initiative. There is a purpose in offering discounts – to increase repeat business. How do you measure the effectiveness of the discount? You need to create an acceptance test, such as "During the next year, 80% of customers will have placed an additional order over their average orders of the past three years." Often acceptance tests as this one are termed project objectives. Objectives should be SMART – specific, measurable, achievable, relevant, and time-boxed.

If this acceptance test passes, then the feature is a success. That is as long as there are no additional business features being added that might affect the outcome, such as providing a personal shopper for every Excellent customer. If the acceptance test fails, it may be due to a variety of reasons such as an insufficient discount or a competitor's discount. Or it may be that the objective is not achievable –the economy is such that customers are not buying. In either case, you have a definitive measurement that suggests a course of action such as increasing the discount or abandoning the feature.

REVIEW

Acceptance tests represent the detailed requirements for a system. They are developed by the triad--customer, developer, and tester-- as part of requirement definition. They are used by developer and testers during implementation to verify the system. Using acceptance tests can double the efficiency in producing new features.

Drive from Business Value

BUSINESS-DRIVEN SOFTWARE DEVELOPMENT

Business-Driven Software Development is Net Objectives' proprietary integration of Lean-Thinking with Agile methods across the business, management and development teams to maximize the value delivered from a software development organization. This approach has a consistent track record of delivering higher quality products faster and with lower cost than other methods.

Business-Driven Software Development goes beyond the first generation of Agile methods such as Scrum and XP by viewing the entire value stream of development. Lean-Thinking enables product portfolio management, release planning and critical metrics to create a top-down vision while still promoting a bottom-up implementation.

Our approach integrates business, management and teams. Popular Agile methods, such as Scrum, tend to isolate teams from the business side and seem to have forgotten management's role altogether. These are critical aspects of all successful organizations. Here are some key elements:

- Business provides the vision and direction; properly selecting, sizing and prioritizing those products and enhancements that will maximize your investment.
- Teams self-organize and do the work; consistently delivering value quickly while reducing the risk of developing what is not needed.
- Management bridges the two; providing the right environment for successful development by creating an organizational structure that removes impediments to the production of value. This increases productivity, lowers cost and improves quality.

BECOME A LEAN-AGILE ENTERPRISE

Involve all levels. All levels of your organization will experience impacts and require change management. We help prepare executive, mid-management and the front-line with the competencies required to successfully change the culture to a Lean-Agile enterprise.

Prioritization is only half the problem. Learn how to both prioritize and size your initiatives to enable your teams to implement them quickly.

Learn to come from business need not just system capability. There is a disconnect between the business side and development side in many organizations. Learn how BDSD can bridge this gap by providing the practices for managing the flow of work.

WHY NET OBJECTIVES

While many organizations are having success with Agile methods, many more are not. Much of this is due to organizations either starting in the wrong place, such as focusing on the team when that is not the main problem, or using the wrong method, such as using Scrum or kanban because they are popular.

Net Objectives is experienced in all of the Agile team methods (Scrum, XP, Kanban) and integrates business, management and teams. This lets us help you select the right method for you.

LEARN TO DRIVE DEVELOPMENT FROM THE DELIVERY OF BUSINESS VALUE

What really matters to any organization? The delivery of value to customers. Most development organizations, both large and small, are not organized to optimize the delivery of value. By focusing the system within which your people are working and by aligning your people by giving them clear visibility into the value they are creating, any development organization can deliver far more value, lower friction, and do it with fewer acts of self-destructive heroism on the part of the teams.

THE NET OBJECTIVES TRANSFORMATION MODEL

Our approach is to start where you are and then set out a roadmap to get you to where you want to be, with concrete actionable steps to make immediate progress at a rate your people and organization can absorb. We do this by guiding executive leadership, middle management, and the teams at the working surface. The coordination of all three is required to make change that will stick.

OUR EXPERTS

Net Objectives' consultants are actually a team. Some are well known thought leaders. Most of them are authors. All of them are contributors to our approach.


Al Shalloway


Alan Chedalawada


Guy Beaver


Scott Bain


Max Guernsey


Luniel de Beer

SELECTED COURSES

Executive Leadership and Management

Lean-Agile Executive Briefing

Preparing Leadership for a Lean-Agile/SAFe Transformation

Product Manager & Product Owner

Lean-Agile Product Roadmaps

PM/PO Essentials

Lean-Agile at the Team

Acceptance Test-Driven Development

Implementing Team Agility

Team Agility Coaching Certification

Lean-Agile Story Writing with Tests

Technical Agility

Advanced Software Design

Design Patterns Lab

Effective Object-Oriented Analysis and Design

Emergent Design

Sustainable Test-Driven Development

DevOps

DevOps for Leaders and Managers

DevOps Roadmap Overview

SAFe®-Related

Implementing SAFe with SPC4 Certification

Leading SAFe® 4.0

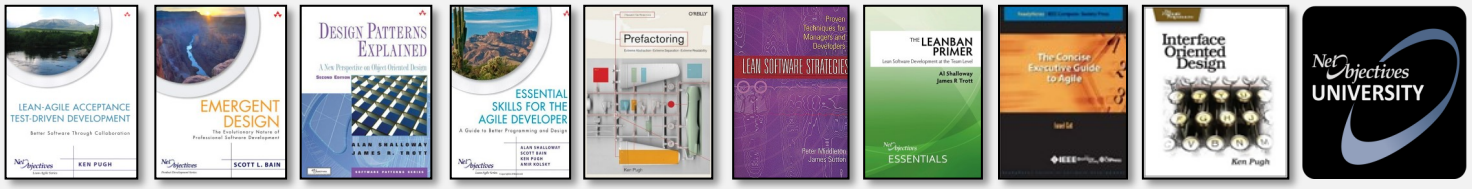
Using ATDD/BDD in the Agile Release Train (workshop)

Architecting in a SAFe Environment

Implement the Built-in Quality of SAFe

Taking Agile at Scale to the Next Level

OUR BOOKS AND RESOURCES



CONTACT US
info@netobjectives.com
1.888.LEAN-244 (1.888.532.6244)

LEARN MORE
www.NetObjectives.com
portal.NetObjectives.com

Net Objectives