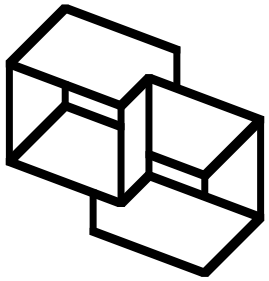
Quality Assurance and Testing in Agile Projects



Pete McBreen, McBreen.Consulting
pete@mcbreen.ab.ca

QA and Testing in Agile Projects

The Manifesto for Agile Software Development

Quality Assurance in Agile Projects - what does it mean?

Agile Testing??? - not just another buzzword

What is so special about Test Driven Development?

Test Driven Development in non-XP projects

Helping users with acceptance testing

Back to basics - what are the real requirements?

Safeguarding the user experience - the role of QA in usability

Ensuring quality is designed into the application

Assisting in the creation of Working Software

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions **over processes and tools**

Working software **over comprehensive documentation**

Customer collaboration **over contract negotiation**

Responding to change **over following a plan**

That is, while there is value in the items on the right, we value the items on the left more.

Quality Assurance in Agile Projects - what does it mean?

Traditional style Quality Assurance seems at odds with the Agile manifesto

- Process and tools are a key part of QA and Testing
- QA people seem to love documentation
- QA people want to see the written specification
- And where is testing without a plan?

So, is there a role for QA in Agile projects?

- Maybe, but the role is different, the tasks are different

So do we need *Agile Quality Assurance*?

The Agile approaches are changing the conversation about software development

Agile shifted our attention to small teams incrementally delivering quality software

The old ideas about testing at the end of the coding phase no longer applicable

We need to think about the role of Quality Assurance in Agile Projects

Definitely NOT business-as-usual

But before we go any further, what exactly is quality in software development?

Conformance to requirements is not it

Nor is passing a bunch of tests

Software Quality is a lot more than that

Quality means something quite different in Agile projects

Working Software

It just barely works?

Software that the users prefer to use?

Software that does what the customer wants, not just what the customer asked for

Software that is responsive to change, on the customer's timescale

What is Agile Quality Assurance?

How long should the customer have to wait once the developers have finished coding?

The project is delivering working software every 2 weeks, QA and Test must keep up!

QA and Test must be in conversation with the customer and users

So do Agile Methods require changes in the way we do Quality Assurance?

More questions than answers right now

At one level, Quality Assurance is still looking at the same deliverables

But the process used to create the deliverables does affect how Quality Assurance works

Many things about Quality Assurance will change drastically with Agile Methods

Unfortunately my crystal ball is refusing to tell me exactly what will change...

But there are some strong hints...

The Changing Face of Quality Assurance

Programmers have become test infected thanks to Beck, Gamma and JUnit

Test Driven Development is now supported by most development environments

Incremental development changes things

Many small production releases are fundamentally different than a single planned release

`Fit.c2.com` may be the next big thing

It could be even bigger than JUnit ☺

Agile Testing - another buzzword or a step in the right direction?

Agile Testing abandons the old notion about how Testers communicate

Requirements and design docs are insufficient, as are test plans and bug reports

Agile Testing sees docs as interesting texts, partly fictional, often useful

Documents are as good as they are going to get

Testers need to join in the conversations with developers and users

Agile Testing is still evolving and being actively discussed on mailing lists

Marick suggests we need new models for Test Development

Testing may be degraded by poor or late docs, but it should not be blocked entirely

Testers should use sources of information other than project docs when designing tests

Test design must take into account what is learned from running tests

The tester must take explicit, accountable action in response to dropped handoffs, new handoffs and changes to the contents of handoffs

Brian Marick has identified two imperatives that affect testing in Agile Projects

Extending the programmers Hands-On Imperative to users

Abstractions simply *mean differently* than working software, so the requirements never represent the application adequately

The imperative toward human contact, face-to-face conversation

Agile methods foster collaboration to obtain communication that documents cannot replace

A Testing Team Motto

We are a service organization whose job it is to reduce damaging uncertainty about the perceived state of the product

http://www.testing.com/writings/purpose-of-testing.htm

Metaphors shape how we think

Testing as Flashlight ...

- A desperate need for a flashlight is a sign of poor preparation
- Over reliance on a flashlight spoils night vision
- Some bugs avoid light
- If the bugs have reflectors, they are easier to spot
- Big heavy flashlights make good clubs (not sure how that fits)

What is so special about Test Driven Development?

Nothing really, good developers have always written unit tests

Test Driven Development has popularized this practice, at least in a few enlightened teams

The *Test First* style has a big impact on quality

But very few people have this habit so it is kind of a moot point (sort of like exercise and healthy eating)

It is also great not to have to use a debugger

But that just means you are not an elite programmer ☺

Test Driven Development in non-XP projects

Yes, XP is just one way to be Agile

The Test Driven style of development is (slowly) catching on where people value working code

So XP has been good for something at least

Realistically, I expect it to take 5-10 years before most code ships with unit tests

Amazingly, Open Source projects are leading the way

Enabling Customer Collaboration

I'm not seeing much of this yet

Software developers just do not get it

Neither unfortunately do the users

Plenty of opportunity for people who do get it

FIT Allows Users to define their own tests using simple HTML documents

FIT could lead to *specification by example*

Especially for standard business applications

FIT derives from a paper Ward Cunningham wrote for XP/Agile Universe Conference

Acceptance Testing as Document Authoring and Annotation

Ward was heavily influenced by the work on *Context Driven Testing*

The choice of process depends on context: what learning will have value right now

Back to basics what are the real requirements?

Understanding the requirements is the key task of Quality Assurance

Kind of obvious huh?

Quality as conformance to requirements?

Next joke please

Requirements are really a myth that arose from the idea of "change control"

We delivered what you asked for, so do not complain

Safeguarding the user experience - the role of Quality Assurance in usability

QA is an advocate for the user experience

Yes, technically it runs, but would you want to use it every day?

Most software is not nice to use

Agile Software Development is trying to change this

Getting the users as an integral part of the team is an essential step to better software

Ensuring quality is designed into the application

How to do this is an open question

Assisting in the creation of Working Software

An extremist view of traditional software development is *Crummy Software Late*

Duff O'Melia of RoleModel Software asks some hard questions

- How do you know your code works?
- Have you ever been afraid to change code?
- How do you know your design is right?
- What happens if your star programmer is hit by a bus?

Agile is one attempt to address this

Quality Assurance on Agile Projects is Different ...

But we do not know how different ...

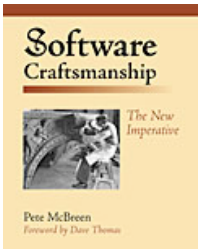
Learning the implications of the Agile Methods is going to be an interesting challenge

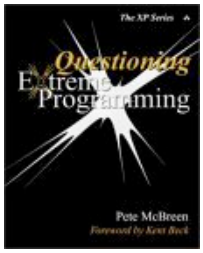
The Quality Assurance community is not renowned for process innovation ...

But *Individuals and interactions*,*Working software, Customer collaboration* and *Responding to change* hold out some hope

Software development is meant to be fun, if it isn't the process is wrong

Some light reading on other topics





Is anyone fluent in Korean? **How about Japanese?**