“In this presentation, Marie will share her experiences working in agile teams across multiple projects and companies over the years. Marie started working in iterative development teams before agile methodologies were accepted at the companies she was working in. Throughout this presentation, Marie will share the challenges and learnings from her journey as an agile tester, and will dispel the common myths and misconceptions around agile (when done properly).”
Who am I?
Marie Walsh
http://www.linkedin.com/in/mariewalsh

• Independent Contractor

• Solutions Delivery Manager for Health Industry Exchange

• ANZTB Board Member & SIGiST Coordinator for Aust & NZ

• Have been in IT / Testing since 1998

• Contact: mariewalsh@anztb.org
OVERVIEW OF AGILE TESTING JOURNEY

Additional slides displayed during presentation only
So...What is Agile.... The Agile Manifesto

- **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*.

http://agilemanifesto.org/
Comparison of models

- **Traditional Software Development**
  - Relies on the unrealistic premise that requirements won’t change
  - Doesn't involve testing early
  - Test late in the process
  - Naturally contributes to a wedge between Business / Development / Testing by nature of baton passing

- **Agile Software development**
  - Encourages change to deliver what the customer wants
  - Testing is involved early and occurs throughout
  - Bridges gaps by collaboratively designing, doing walkthroughs and working to **prevent** defects - not find them a moment before shipping to Production.
Flavours of Agile

Well known “flavours” of Agile include:

- Agile Modelling
- Agile Unified Process (AUP)
- Crystal Clear
- Dynamic Systems Development Method (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Graphical System Design (GSD)
- Kanban (development)
- Lean software development
- Scrum
- Velocity tracking
- etc....
Where does testing fit into Agile?
## Testing activities across iterations

<table>
<thead>
<tr>
<th>Iteration Planning</th>
<th>Next Iteration</th>
<th>Current Iteration</th>
<th>Previous Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Strategy/Plan</td>
<td>Story Reviews</td>
<td>System Testing</td>
<td>Clean up activities</td>
</tr>
<tr>
<td>Identify &amp; Prepare:</td>
<td>• Identify Acceptance Criteria</td>
<td>• Accept frequent builds into Test (eg: daily, twice daily, etc)</td>
<td>• Finalise test case execution</td>
</tr>
<tr>
<td>• Environments</td>
<td>• Identify high level test scenarios</td>
<td>• Document and perform tests for dev completed tasks / stories</td>
<td>• Update test cases with any changes from iteration</td>
</tr>
<tr>
<td>• Data</td>
<td>• Prepare test management tool</td>
<td>• Automate passing test scripts flagged for automation</td>
<td>• Test automation maintenance</td>
</tr>
<tr>
<td>• Resources</td>
<td>• Identify any potential impacts to automation</td>
<td>• Perform Exploratory Testing</td>
<td>• Performance testing activities</td>
</tr>
<tr>
<td>Perform test automation maintenance</td>
<td>• Review:</td>
<td>(refer to next slide)</td>
<td>• Showcase</td>
</tr>
<tr>
<td>• Update invalid scripts</td>
<td>• Environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Retire redundant scripts</td>
<td>• Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• General script &amp; code tidy up</td>
<td>• Resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Agile Testing Phases

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Pre-Check in Testing</th>
<th>Build Verification Testing</th>
<th>Daily Functional Testing</th>
<th>Iteration Stabilisation</th>
<th>Project Stabilisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process verification</td>
<td>Verify processes have been followed, and feature is complete.</td>
<td>To determine if the build is worth testing.</td>
<td>To ensure application stability has not regressed and new features are working as expected.</td>
<td>To ensure the combined new features with existing functionality are production quality before progressing to the next feature set.</td>
<td>To ensure the application can be released into production.</td>
</tr>
<tr>
<td>Frequency</td>
<td>Every check in</td>
<td>Every daily build</td>
<td>Daily</td>
<td>Once per iteration</td>
<td>Once per release</td>
</tr>
<tr>
<td>Time Required</td>
<td>½ hour per check in</td>
<td>½ hour</td>
<td>A few hours</td>
<td>Multiple days</td>
<td>Multiple weeks</td>
</tr>
</tbody>
</table>
Example: day in the life of an agile tester

12:00 pm

Maintenance

9:00 am

Analysis of test failures

Automation develops new automated scripts

Manual testers continue verification testing & test case writing

2:00 pm

New Build deployed by Release Team

Build verification testing

3:00 pm

Daily Testing Workflow

3:30 pm

Daily Stand up

4:30 pm

Automated tests execute

5:00 pm
Lessons Learnt

Along my journey working with dysfunctional and highly functional teams, I learnt the following, which I now apply to every new project where possible:

• Communication within the team is key – testers and developers should be co-located to enable free flowing conversations and discussions

• Tools – not just automated tools – other tools to be used are:
  • Wiki – central location for all written communication
    • Use for recording design discussions and decisions
    • Collaborative sharing and viewing of important information
    • Keeps history automatically – easy to see changes made
    • Searchable – able to find information quickly
    • Can subscribe to pages to know when they are updated
  • Agile Boards – either physical on online
  • Team Calendars

• Experienced staff – as with all development, working with experienced staff is vital. If a new less-experienced staff member joins the team, pair them with the experienced staff for a few weeks – have a mentor programme in place.

• One size does NOT fit all!!! Find what’s right for your team and use it – discard the things that slow you down.

• Continuous Integration is your friend!!
Agile Boards
## Dispelling Agile Myths and Misconceptions

<table>
<thead>
<tr>
<th>Agile Myth / Misconception</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agile means no documentation</strong></td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Agile means you never have to write documentation&quot;</td>
<td>Agile values working software over comprehensive documentation. Documentation should be “just enough” and “just in time”. Documentation should not be rigid, it needs to cater for changing requirements.</td>
</tr>
<tr>
<td><strong>Agile means no discipline</strong></td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Agile is a sloppy, ill disciplined way of developing.&quot;</td>
<td>Agile teams must be more disciplined for success. Agile engineering practices such as test driven development, automated builds and continuous integration create a very focused, transparent and efficient way of building a product.</td>
</tr>
<tr>
<td><strong>Agile means no plan</strong></td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Agile projects don’t plan. It’s very chaotic and direction-less.&quot;</td>
<td>Agile continuously plans and estimates, it is not a one time up front activity. Continuous planning, adapting and refining allows for flexibility and changing requirements (which happens on ALL projects).</td>
</tr>
<tr>
<td><strong>Agile is a silver bullet solution</strong></td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;We are constantly over schedule – Agile will save us&quot;</td>
<td>Bad disciplines will be more evident in an agile environment. Switching to agile needs to be for the right reasons – not as a reaction to poor performance. Agile provides the ability to deliver smaller working deliverables sooner, rather than “big bang” deliverables.</td>
</tr>
<tr>
<td><strong>Agile only works for trivial projects</strong></td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Agile isn’t scalable, therefore can only be used on small projects&quot;</td>
<td>Agile has been used on many large programmes of work successfully. Large corporations are adopting agile practices as they mature their software delivery.</td>
</tr>
<tr>
<td><strong>Agile projects always fail</strong></td>
<td>POSSIBLY TRUE – BUT NOT IN ALL CASES</td>
</tr>
<tr>
<td>&quot;We’ve tried agile, and it failed......&quot;</td>
<td>Agile isn’t something you just “do”. Rationalisation for changing from traditional practices to agile practices needs to be accepted by the teams and actively adopted. Changing practices isn’t trivial, therefore needs buy-in from all team members. Any team member who doesn’t understand or doesn’t want to understand agile practices can be the demise of the whole team.</td>
</tr>
</tbody>
</table>
Why agile fails….

- We do stand-ups – therefore we are agile…..
- We have an agile board, we update it every week – therefore we are agile…..
- We’re agile – we don’t need testers or analysts

Cartoon: We're going to try something called Agile Programming. That means no more planning and no more documentation. Start writing code and complaining. It was your training.
Thank you

Marie Walsh
mariewalsh@anztb.org