Are you ready for Agile?

Going Agile, Staying Agile
You burn the toast, I’ll scrape it
Remember when coding looked like this?
What Happened to Waterfall?

- It must be fully reliable and resilient, fully tested
- Must have **total ownership** by management & staff
- Staff must have confidence in the systems reliability

- Must allow fully for consultation, quality assurance, testing, and training
- Introduce the new system in a **step-wise approach**
- Steps given **maximum benefit to be implemented first**
1. Lack of clear links between the project and the organisation's key strategic priorities, including agreed measures of success.

2. Lack of clear senior management and Ministerial ownership and leadership.

3. Lack of effective engagement with stakeholders.

4. Lack of skills and proven approach to project management and risk management.

5. Too little attention to breaking development and implementation into manageable steps.


7. Lack of understanding of, and contact with the supply industry at senior levels in the organisation.

8. Lack of effective project team integration between clients, the supplier team and the supply chain.
Iterative is normal
Find the feedback

- Requirements
  - Design
    - Implementation
      - Verification
        - Maintenance
  - Requirements Analysis
    - High Level Design
      - Detailed Specifications
        - Unit Testing
          - Coding
            - Integration Testing
              - Operational Testing
                - Review/Test
                  - Ongoing Support
Agile is about Quality Feedback

Creating **quality, working deliverables** of the **highest possible business value**

Whilst **reducing the risk** of failure
Agile is an umbrella term for a variety of adaptive, iterative and incremental techniques and methods.
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.
# Agile Principles

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Satisfy customer through early and continuous delivery</td>
</tr>
<tr>
<td>2.</td>
<td>Welcome changing requirements, even late in development</td>
</tr>
<tr>
<td>3.</td>
<td>Deliver working software frequently</td>
</tr>
<tr>
<td>4.</td>
<td>Business and developers must work together daily</td>
</tr>
<tr>
<td>5.</td>
<td>Build projects around motivated individuals</td>
</tr>
<tr>
<td>6.</td>
<td>Face-to-face conversation is the most efficient and effective method of collaboration</td>
</tr>
<tr>
<td>7.</td>
<td>Working software is the primary measure of progress</td>
</tr>
<tr>
<td>8.</td>
<td>Agile processes promote sustainable development</td>
</tr>
<tr>
<td>9.</td>
<td>Continuous attention to technical excellence and good design enhances agility</td>
</tr>
<tr>
<td>10.</td>
<td>Maximising the amount of work not done</td>
</tr>
<tr>
<td>11.</td>
<td>The best results emerge from self-organising teams</td>
</tr>
<tr>
<td>12.</td>
<td>At regular intervals, the team reflects and adjusts its behaviour accordingly</td>
</tr>
</tbody>
</table>
User Stories

User
Who is going to use the product?
How will they use the product?
What will they do with the product?

Story
A description of how a user interacts with the product:
   As a <user>,
   I want <feature>,
   so that <benefit>
User Stories are an established method of clear communication between the team and the business.

- **Card**
  - Stories are usually written on cards
  - Annotated with estimates, notes, etc

- **Conversation**
  - The details of the story emerge in conversations with the Product Owner and business

- **Confirmation**
  - Acceptance tests confirm that the story has been implemented correctly
• Details are essentially conditions of satisfaction and translate to tests
• These can be added as Acceptance Criteria
• Typically written on the back of the index card

User Story:
As an advisor I want to be able to view all my clients so that I can see a summary of all clients in one place….

Acceptance Criteria:
• Search returns 200 clients
• Select next 200 option available
• Extract to excel spreadsheet
• 200 results returned in < 5 seconds
• All results returned in <10 seconds
• Returns active + inactive customers
• Includes scale + heritage products
• Adviser can filter by agency code from a pre-populated drop down list of agency codes
Scrum Process

**Vision**
- Aim of the project
- With a business owner

**Product Backlog**

**Sprint Backlog**

**Sprint Planning Meeting**
- Review Product Backlog
- Build Sprint Backlog
- Commit to selected scope

**Daily Stand-Up Meeting**
- Done since yesterday?
- Plan for today?
- Barriers?

**24hrs**

**1 – 4 weeks**

**Release Planning Meeting**
- Product Backlog
- Prioritised features desired by customer

**Sprint Review Meeting**
- Demo completed features to all stakeholders

**Potentially Deployable Increment**

**Sprint Retrospective Meeting**
- How did we do?
- What can we improve?

*Ken Schwaber and Mike Beedle*

"Agile Software Development with Scrum" Prentice-Hall, 2001
Test Driven Development

- Developer Behaviour
  - Develop Test Cases First
  - Develop code to satisfy the test cases
  - Refactor the code as new code is written
- Brings requirement clarity before coding
- Ensures good Quality
Continuous Integration

- All developers commit at least every day
- Keep the build fast
- Development environment mirrors production environment as closely as possible
- Make it easy for anyone to get the latest executable
- The team has a real time view of the build status
Strategy

1. Identify business pain points
2. Derive from these currencies for the agile enablement
3. Derive feature stories with acceptance criteria
4. Define practices that align with the business values identified
## Pain Points / Enablement currencies

<table>
<thead>
<tr>
<th>The Organisation Pain Points</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Takes too long to deliver software</td>
<td>Reduce cycle time</td>
</tr>
<tr>
<td>2 - Business wants more involvement</td>
<td>Increase value delivered</td>
</tr>
<tr>
<td>3 - Fail to get good requirements</td>
<td>Increase value delivered</td>
</tr>
<tr>
<td>4 - Only one release per year</td>
<td>Increase quality</td>
</tr>
<tr>
<td>5 - Long lead time into production</td>
<td>Reduce Cycle Time</td>
</tr>
<tr>
<td>Pain Point</td>
<td>Currency</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| 1, 4       | Reduce Cycle Time, Increase quality | As the organisation I want to have our software delivered sooner so that the Return on Investment starts earlier | 1. Are we practicing 'iteration' and its supporting practices?  
2. Are we practicing 'continuous integration' and its supporting practices?  
3. Are we practicing 'test driven development' and its supporting practices?  
4. Are we practicing 'test driven requirements' and its supporting practices? |
| 2          | Increase Value Delivered | As a business sponsor I want to be involved with the decision behind the software I sponsor so that I get the right things delivered. | 1. Are we practicing 'customer as part of the team' and it's supporting practices? |
| 3          | Increase Value Delivered | As a Director of Information Management I want to understand the business needs better so that we deliver what the business really requires. | 1. Are we practicing 'customer as part of the team' and it's supporting practices? |
| 5          | Reduce Cycle Time | As the organisation I want to reduce the lead time delivery from ITIS to user availability so that the return of investment is starts earlier. | 1. Are we practicing 'iteration' and its supporting practices?  
2. Are we practicing 'continuous integration' and its supporting practices? |
## Currency / Practice mapping

<table>
<thead>
<tr>
<th>Practice</th>
<th>Reduce Cycle Time</th>
<th>Increase Value Delivered</th>
<th>Increase Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Product Backlog</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Done State</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Customer involvement</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Cross Functional Team</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Driven Requirements</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Functional Tests</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Test-Driven Development</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Release Often</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Evolutionary Design</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>
## Currency / Practice mapping

<table>
<thead>
<tr>
<th>Practice</th>
<th>Reduce Cycle Time</th>
<th>Increase Value Delivered</th>
<th>Increase Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refactoring</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Simple Design</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Show &amp; Tell</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>User Stories</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Stand-Ups</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Collective Code Ownership</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Pair Programming</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Automated Testing</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Restrospective</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Release Planning</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Iteration Planning</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Information Radiators</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>
In software engineering, an **anti-pattern** is a pattern that may be commonly used but is ineffective and/or counterproductive in practice.
Anti-Pattern:

Title

Problem

Solution

Compromised Values and Principles

Refactored Solution
Anti-Pattern:

No User Acceptance Tests
The software engineering group know that quality is at the heart of agile and introduced a testing policy as part of their software engineering process.
Anti-Pattern Solution:
The software engineering process has testing policy, unit testing and sets a high test coverage. All new projects follow the unit testing rigorously.

Why does my Show and Tell turn into an embarrassment of failed features?
Compromised Values and Principles

The Value:
**Working software** over comprehensive documentation

The Principle:
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

The Principle
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

The Principle
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

The Principle
Continuous attention to technical excellence and good design enhances agility.
Integrate a member of the test team into each development team to develop User Acceptance tests that support the developer’s work.

Introduce an automated User Acceptance testing tool e.g. FitNesse, Cucumber, easyb.

Introduce a consolidation cycle, review the software towards the end of the iteration and manually test the software.
Anti-Pattern:

No design
Problem:
The software engineering group has now dropped the big upfront design approach now adopts emergent design as part of their design approach.
Anti-Pattern Solution:
The development team has no coherent vision and an understanding of the design and how to progress on the solution to the business problem.
<table>
<thead>
<tr>
<th>The Value:</th>
<th>Working software <em>over</em> comprehensive documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Principle:</td>
<td>Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.</td>
</tr>
<tr>
<td>The Principle:</td>
<td>Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.</td>
</tr>
<tr>
<td>The Principle:</td>
<td>Continuous attention to technical excellence and good design enhances agility.</td>
</tr>
</tbody>
</table>
The development team needs to collaborate and come to a solution together so all have a clear vision of the issues and the solutions. Hold design meetings after the iteration planning meeting and on a as needed thereafter where all team members come to a collaborative solution.

Document the solution on white boards in the team area so the solution is displayed to all and easily discussed as the solution is being developed.
We need resources that are limited and therefore not permanent members of the team
Problem:
The software engineering group has now evolved to a cross-functional project team.
Anti-Pattern Solution

The DBAs are the only people who have access to the databases and the only resource that writes SQL queries. We normally have to request DBA resource a few weeks in advance to minimise the bottleneck.
Compromised Values and Principles

The Value:
*Individuals and interactions over processes and tools*

The Principle:
Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

The Principle
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Refactored Solution

Negotiate with the DBA team so that the project team has access to a DBA on a regular basis e.g. Two half days a week.

Negotiate with DBAs to attend iteration planning meeting. From results negotiate resource to ensure bottleneck does not occur.
How do we know when we are “done” with a Story
Problem:

To support the agile approach to project development the software engineering group has now adopted User Stories as a way of capturing requirements.
Anti-Pattern Solution:
The User Stories adopted the template “As a... I want... So that...”.

How do I know when the User Story has been satisfied?

There doesn’t seem to be much detail behind each requirement!
Compromised Values and Principles

The Value:
**Individuals and interactions** over processes and tools

The Principle:
Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

The Principle
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
Refactored Solution

User Stories are a token for a conversation. This means developers have to collaborate with the business more than they have ever done before.

As an output from the iteration planning meeting ensure that the business requirement is well understood and that as many questions to understand what is being asked for have been asked. It is the responsibility of each project member to understand what is being asked for in each User Story.

Define the acceptance criteria fully with the business so there are no misunderstandings expressed during the Show and Tell. The business add their initial acceptance criteria to the User Story during creation this is then enhanced by the development team during the collaboration to understand the exact detail behind the User Story.
Agile change is complex and not always achieved in a single journey. It requires not only focusing on the visible elements of Agile – process management, tools and techniques – but more importantly, on the invisible elements; strategy and alignment, leadership and employee behaviour and engagement.
Anti-Pattern:

Static development process
Problem:
The business can see benefits from implementing agile methods. Agile is therefore implemented as the standard software engineering process.
Anti-Pattern Solution:

The software engineering process is rewritten to be agile (Scrum/XP/FDD …). All new projects follow this new process. The process is static and never revised.

*Agile processes/practices have been prescribed why change?*
Compromised Values and Principles

The Value:
Individuals and interactions *over* processes and tools

The Value:
Responding to change *over* following a plan

The Principle:
At regular intervals the team reflects and adjusts its behaviour accordingly
Always ask yourself “How can we work in a more agile way.”

Agile is all pragmatism not dogmatism and remember - There isn’t inherent value in the practices…it’s the consequences of using the practice.

A pillar of agile is continuous improvement always strive for perfection. Continually retrospect.
More anti-patterns

- Cross Functional teams
- Testing
- User Acceptance
- Collaboration (inter-developer)
- Collaboration (PO/developer)
- Team autonomy
- Executive support
- No Product Owner
- Test automation
- Operations Team engagement
- How do we know when we are “done” with a Story
- Iteration Vision
Thank You

John McNeill
IT Process & Service Management, Global Consulting Practice
john.mcneill@tcs.com
+44 7921 936 744