**Quality people, quality solutions.
Planit Software Testing.**

Abstract

Agile software development is a subject that can polarise IT professionals, with strong views on both sides. This paper looks at some of the Agile principles and how they fit into current work practices regardless of the methodology being followed. These are practices that can help the testing team meet the business need to deliver projects faster, while not compromising quality.

Agile—Why the fear?

Introduction

The technology & software development industries are very good at adopting the latest technology trends and development practices but these trends and practices often polarise opinion on how good or bad the new technology or software development practices are. Agile software development processes and methods is probably the most potent example of where the industries practitioners range from supportive and embracing of the concepts, through to those who simply see another example of an excuse for bad practice or make statements to their peers "that it will never work in our organisation".

It is our observation that much of this polarisation appears to be driven by a lack of understanding of what it actually means to be "agile" or more importantly, what "agile" means within the context of the individual, i.e. what role they perform, and their organisations software development context, i.e. the type of software development they perform.

This white paper will discuss some of the aspects of what Agile software development is, how do we do it, how do we get it implemented and how do we get it to work in our organisation. Or rather, the level of cultural and skills change required by the organisation for agile development to be successful is so monumental that it makes the likelihood of success so low, that it is never going to work within your organisational context.

Ironically, in these organisations where the perception and opinions of key individuals is polarised to the point that "agile will never work in our organisation", may be surprised to discover that they are already applying various agile techniques within there software development!

Below are the 12 Principles of Agile Methods. I have taken a couple of these to demonstrate how you may already be using agile techniques and therefore the adoption of an agile process may not be the great leap that you think it will be.

Having worked in a number of organisations across varying software development programmes, it is my observation that most teams are, in some shape or form, adopting some of the agile principles that were created when the agile manifesto was written.

Here are a few suggestions on adopting and applying some of the more common agile principles. If you haven't started down the agile path, then this list can also be used to help you identify where to start.

**Quality people, quality solutions.
Planit Software Testing.**

12 Principles of Agile Methods

1. *Our highest priority is to satisfy the customer through the early and continuous delivery of valuable software*

2. *Welcome changing requirements, even late in the development. Agile processes harness change for the customer's competitive advantage*

3. *Deliver working software frequently, from a couple of weeks to a couple of months, with preference to the short time scale*

4. *Business people and developers must work together daily throughout the project*

5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done*

6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation*

7. *Working software is the primary measure of progress*

8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely*

9. *Continuous attention to technical excellence and good design enhances agility*

10. *Simplicity – the art of maximising the amount of work not done – is essential*

11. *The best architecture, requirements and designs emerge from self-organising teams*

12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjust behaviour accordingly*

Practice 1: The Daily Stand-up Meeting

One of the most common agile practices and often the easiest to implement is the daily team stand-up meeting. Most project teams hold various levels of meetings to discuss current status, work to be completed before the next meeting and the issues and risks currently being faced or identified. The frequency of these project level meetings can range from daily, weekly or monthly.

Each of the individual skills teams, i.e. the testing team, will often increase the frequency of team meetings as they approach the start of testing the system i.e. executing the designed tests. These daily meetings are critical to understanding what is happening within the testing cycle, especially with respect to any issues that are slowing or preventing test execution from progressing. It is a collective meeting where team members offer to help others or ask for help and prioritise the teams work. It is not a meeting where the Manager tells them what they must do.

Also significant roadblocks are discussed at this meeting for example where we are waiting on confirmation of requirements or a particular defect is holding up a lot of test cases. It is also a great opportunity to talk about process improvements and continuous change. Often these suggestions are resulting from inefficiencies that have been identified by the team. Or it may be that one of the team has been trying a new procedure which when discussed is frequently taken up and used by the rest of the team.

It is important that this meeting focuses on the critical issues and not become a "complaints" meeting. The manager acts as the facilitator to ensure that the three basic questions used in an agile daily stand-up meeting are explicitly discussed. The same rules apply to these meetings as they do to daily stand-up, that is:

- The meeting is time-boxed - generally no more then 15 minutes

- Keep the focus of the meeting on the following questions:

    - "what have you completed yesterday"

    - "what are you doing today"

    - "what road blocks have you struck"

- The manager is the facilitator to assist the team in developing a collective responses and solutions to the issues and roadblocks raised

- The manager, like a scrum master, is responsible for eliminating the road blocks identified

**Quality people, quality solutions.
Planit Software Testing.**

Practice 2: Face to Face Communication

All projects need effective lines of communication, both internally within the team and externally, outside of the team. The most effective team communication is via face to face discussions and meetings.

Here are some examples that we have used and also seen used within organisations that facilitated effective face to face communication.

The Defect Management Meeting

This meeting provides an ideal opportunity to communicate face to face with a number of key team members. The meeting must be time-boxed and it helps to remind the defect meet attendees of what their purpose is in attending the meeting.

The goal of the meeting is normally to review and prioritise all new defects found and review the status of existing high priority defects to enable a new list of prioritised defects being defined.

The key challenge with a time-boxed defect meeting is avoiding the game of "defect ping-pong". This is where a defect bounces backwards and forwards between individuals or teams as the attendees try to understand what exactly the defect is, how it manifests itself, what the level of impact is on the system or the business and who is responsible for fixing it. While this last point about "who is going to fix it" is less of an issue for an agile team then it is for software development projects which have outsourced actual development to an external vendor. The project managers (or all team managers) need to be able to facilitate a collective response and agreement on who will fix it to minimise delays. In these cases, having high level stakeholder support is often required to assist the vendor accept responsibility where appropriate.

Reviewing defects amongst the audience responsible for making the decisions at this meeting will clear up any communication issue and ensures that the prioritisation is directed by the whole team focusing on resolving the most critical issues first.

To ensure that the time-box is preserved, the meeting prioritises the defects starting at the highest severity working down the severity levels until the allotted time have been reached.

Walkthroughs and Workshops

The use of walkthroughs and workshops with the Business, Architects and Development team to discuss the changing parts of the system, the new working processes or any particular parts of the Business Requirements or Functional Specification can help to quickly disseminate information to a large number of people. These types of meetings also discover new information which will result in some project response to deal with the new details uncovered.

One of the key workshops/walk-throughs is between the development team and the testing team to "openly" discuss how the system is going to be tested. The topics covered in these meetings look at what supporting software like drivers or test harness will be required to facilitate more efficient and effective testing as well as understand how each group will support each other during test execution. By including the development team in how you believe the system should be tested, the developers will provide there perspective on the approach. Facilitating this discussion is probably the most important topic to be covered as it results in a shared understanding of how the system will be tested and what areas should be focused on more then others. Discussing these topics with the development team will establish a healthy working relationship such that informal discussions can occur about any problems found (and potentially fixed as a result).

When clarifying the requirements in a requirements walkthrough with the business analysts it is important to involve both the testers and the developers attend to ensure that both teams receive the same information and everybody gains a collective understanding of what the requirements are. This meeting will turn into a workshop as questions and challenges are raised about what a requirements actually mean as will as identify the details that are missing.

Walkthroughs of software often take place between development, testing staff and Business representatives at an agreed period prior to formal release to the testing environment. This is certainly easier to perform where the software development occurs in-house, but video conference and web based technologies can enable the process to occur even when the software development team is remotely located.

The developers present the software that they are about

Quality people, quality solutions.
Planit Software Testing.

to release to the testing team. This promotes dialogue between the two teams and often defects will be found at this stage. Some of these defects can be fixed prior to the softwares release into the testing environment. Fixing the defect at this point is much cheaper then following the more formalised approach adopted once the software is in the test environment. This approach can occur with the developers during the actual software development phase but is an approach that works more with co-located teams then geographically dispersed ones.

If there are a number of defects it needs to be agreed between the two teams whether there is enough functioning software to proceed with the scheduled release or to delay the release and fix all/most critical defects before a new deployment.

Change requests are also an example of gathering "just enough" information required to implement the change. Change requests that are raised later on in the project are often only a high level statement of the intent of the change. All affected parties should meet to discuss the lower level detail of the change, i.e. what does it impact, do we understand what the change is actually for, and provide estimates for completion of this work. Basic change management processes are good examples of an iterative planning approach and gather "just enough" detail to be able to start the tasks, expecting to find out more details as you progress.

All of these examples are about opening up lines of communication to enable information to flow freely and openly between the teams. It streamlines communication reducing the need for unnecessary documentation such as emails as well as reducing the number of similar questions from multiple team members.

Practice 3: The Business must be involved on a daily basis

Subject Matter Experts (SME) and BAs joining the testing team at specific periods within the project also promotes better communication. The following are examples of where the involvement of the SME or BA can be extremely beneficial:

- ♯ Reviewing of defects with the testing team before they are formally raised

- ♯ Working with the testing team during test case preparation to answer questions immediately and ensuring that the test cases contain the correct information through informal reviews and that the test cases provide full coverage of critical Business processes and functions. The Business should also provide the priority for the test cases

- ♯ During test case execution to answers questions where the test case does not match the software delivered or to clarify such information as new working practices. They may also supply information on how the existing system works which may not have been detailed within the requirements documents, but may be needed to complete the testing

Developing relationships with the user community and organising some of the testing staff to sit with the users for a couple of days to observe how they work can provide a significant improvement of the understanding of how the users actually use the system. Consideration needs to be given to ensuring that any disruption to the users can impact there work, but the benefits that this contact with the users can provide is extremely beneficial early in the project as the testers understand the usage of the system. This approach can create relationships which continue throughout the project and can result in assistance being providing with test case preparation, execution and defect detection. It is not unusual to see some of the users join the testing team either as an SME or during test execution.

The involvement of the user in the agile team is a fundamental principle and this is not full time involvement it is some-one from the Business to bounce ideas off and ask questions if the tester is unsure of the systems functionality. It is often valuable in helping with the prioritisation of the test cases by understanding what

Quality people, quality solutions.
Planit Software Testing.

business processes are used more frequently then others as well as different work flows that the testers may not be familiar with.

Linked with the above is the invitation for users to join the testing team. This does need to be managed carefully in order to not disrupt the testers while adding value to the team and giving a meaningful experience to the user group. Again involvement of the users is a principle concept of Agile. As above they can provide the same value add.

Practice 4: Deliver working software frequently

There are a number of definitions for increment and iterative development which are fundamental to the agile methods. The following are two of the clearest definitions of what incremental and iterative development mean.

Project teams actually use iterative development, i.e. time boxing, within every project. The following are examples of where time-boxing and iterative processes are followed:

- Towards the end of test execution for managing late change requests and the defect fix and test cycle

- Determining which defects and change requests are delivered in which build to the test environment

- Often project deadlines are fixed and the resources are fixed, particularly nearing the end of execution when there is little value in introducing new members as the lead in time for familiarisation with the tests and system outweighs the execution the individual could undertake. This means that the original scope of the project may need to be cut and functionality moved to subsequent releases

Definition: Incremental Development

Is a staging and scheduling strategy in which various parts of the system are developed at different times or rates and are integrated as they are completed.

The alternative is to develop the entire system with a big bang integration at the end.

Better Software, April 2008, Incremental and Iterative Development: How and Why they are Different, Alister Cockburn

Definition: Iterative Development

Is a rework scheduling strategy in which time is set aside to revise and improve parts of the system.

The alternative development is to get it right the first time (or at least declare that its right)

Better Software April 2008, Incremental and Iterative Development: How and Why they are Different, Alister Cockburn

Conclusion

As you can see, from the examples above, many of these practices we perform everyday in our projects actually align with a number of the principles listed in the 12 principles of Agile Methods. While the overall software development method being followed by these projects examples uses traditional software development processes, these projects are still driven by the same business needs of getting projects to market faster and cheaper in an effort to stay ahead of the competition.

This global business context is forcing practitioners to look for more effective ways of performing there roles and none more so then an organisations testing team. Of all the roles in a project, testing has and needs to follow an iterative styled approach. Whether it is the requirements review and test design process which iterates over understanding and improving the requirements or it's the iterative approach to test execution performed against different release versions of the system or simply the normal defect fix and test cycles.

Testing has a natural fit with an iterative based approach. But while being fully agile requires all parts of the development process to fundamentally change, testing has the opportunity to actually employ agile principles without necessarily needing the entire project to be actually following an agile process. The "leap" to agile is therefore not necessarily as large as you may think and can be implemented within the team to make some aspects of the testing process and tasks more effective and efficient within the context of the project.

While testing has the potential to use a number of agile techniques, it doesn't exist in isolation from the rest of the project team or the business. Involving these groups in what your approach to testing will be is instrumentally to gaining their support with adopting these practices across your testing team.

Leanne Howard is a Principal Consultant with Planit. Over the past 20 years Leanne has worked in the IT Software Testing industry after working as a Financial Executive. She is a specialist in the implementation of practical testing methods with a strong focus on client satisfaction.

Sydney
Level 16
201 Miller Street
North Sydney NSW 2060
+61 2 9464 0600

Melbourne
Level 10
2-26 Elizabeth St
Melbourne Vic 3000
+61 3 9666 4444