Richie Siconolfi

Barry McManus

AN INTRODUCTION TO AGILE METHODOLOGIES

FIRST IN A SERIES OF ARTICLES FROM THE DIGIT COMMITTEE'S AGILE WORKING GROUP

Increasingly, regulated organisations and their IT partners are turning to Agile software development methodologies to assist with improving the speed to production of their IT solutions. The arguments for and against using Agile methodologies versus the traditional V-model or Waterfall styles of software development lifecycles (SDLC) are many. Recent articles and podium presentations are challenging the norm with examples of where Agile processes and customer integration have enabled faster delivery and reduced cost whilst maintaining an emphasis on quality.

The barriers to adopting Agile are more likely to be Enterprise culture related than technical capability. Our IT management and business leaders are already forging ahead with these new methodologies, thus we need to ask the question, how does the quality unit need to adapt?

In this first in a series of articles, we look at the fundamentals and history of Agile, which contrary to belief has been around for many years.

INTRODUCING AGILE

It is interesting to point out that there are over 65 software development methodologies[1] many of them describing Agile practices. To understand Agile, we should first understand the evolution of software development, the timeline below illustrates the introduction of some methodologies. (See Fig. 1)

EARLY AGE

At the beginning of the computing age, the approach taken to build software was to code, (test) and fix any issues. This was acceptable given that the prohibitive cost of computing limited its availability to very few people, (often the same people who coded the software).

During the 1970s, formal (traditional) methodologies were established to coordinate the tasks required to deliver greater software complexity. These methodologies shared the following traits: highly organised, highly structured, sequential activities providing a high degree of assurance (e.g. V-Model, Structured System Analysis and Design Methodology [SSADM]).

These software development lifecycles focus on providing complete functionality according to a set of user requirements, for complex and critical systems.

MIDDLE AGE

The changing pace of technology and business needs placed a strain on traditional methodologies, e.g. late system delivery, out of date functionality and excessive costs. Thought leaders derived new methodologies to overcome their problems. Here are a few selected examples:

- SPIRAL introduces active user involvement in iterative (several cycles to complete) prototypes to elicit requirements before settling into a traditional sequential approach[2]
- Rapid Application Development (RAD) incrementally (bit-by-bit) builds functionality to deliver important features sooner and performs parallel tasks to reduce delivery time[3]
- DSDM (Dynamic Systems Development Methodology) fixes costs, quality and time as system development constants and treats functionality as flexible. The incremental and iterative focus increases flexibility to meet ever changing business needs[4]
- SCRUM seeks to reduce the reliance on prescribed processes and allow software teams to self-organise themselves to decide on the best way to work[5]
- Extreme Programming (XP) concentrates on taking good engineering practices to 'extremes' e.g. test driven development (writing unit tests to fail and then writing the code to pass those tests), test early and test often and increased feedback (team feedback, code feedback and customer feedback)[6].

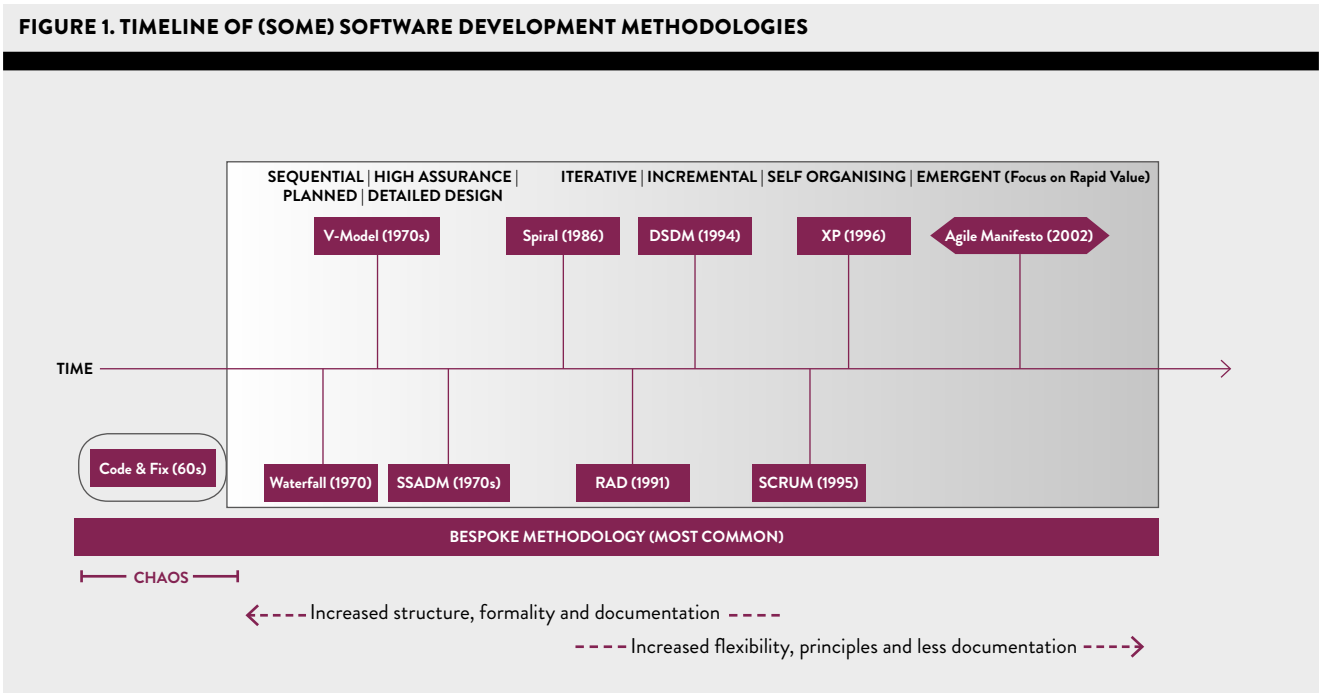
AGILE MANIFESTO

In 2002, Thought Leaders in software engineering defined and signed the Agile Manifesto which favours:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan[7].

The Agile Manifesto does not specify a single methodology. It is a philosophy in the approach to development and as can be seen from above, many methodologies can be defined as Agile. While adherence to a single Agile methodology is rarely practiced, an Agile development process is often implemented by selecting the parts from a number of described methodologies. Furthermore, some of these methodologies are dynamic, they are ever changing as in-house groups and suppliers learn what works and adapt their Agile processes accordingly (note: the move to the right in Fig.1 reflects increased flexibility of the approaches). The key takeaway for the QA professional is that there is not a single Agile process to learn, the methodology used will most likely be unique to the IT program.

FIGURE 1. TIMELINE OF (SOME) SOFTWARE DEVELOPMENT METHODOLOGIES



AGILE COMPARED TO TRADITIONAL

Agile approaches are iterative, incremental, seek active user involvement, favour personnel knowledge over comprehensive documentation and focus on rapid feedback. Traditional methodologies have up front planning, predefined phases, detailed processes and comprehensive documentation[8]. The objective of Agile is rapid value whereas the objective of traditional is high assurance[9]. Software developers prefer the level of freedom provided by Agile and are sceptical of the benefits of comprehensive documentation.

As we have commented, it is rare to find an organisation that adheres to the letter of a methodology definition, perhaps reflecting the preferences of strong individuals within a development team or the demands of a project (if it is the former, the question must be asked as to what activities are diluted or omitted). The volume of methodologies available might hint that not one of them is a remedy to all the challenges facing computerised systems[10].

There are many development methodologies for different software challenges. Each methodology pertains to overcome problems as seen by their originator. Agile activities are not new as they have been in use since the 1970s. For example, the use of daily stand up meetings (SCRUM) to plan tasks, track progress and identify impediments, was a feature of any project approach that was in fire-fighting mode, when delivery dates were at risk of not being met. Agile activities yield great benefits when utilised correctly, but this is true for all development activities.

When assessing the development approach implemented by a supplier, it is recommended that an understanding of the objectives of the approach is sought and to ascertain whether those objectives are compatible with business needs (e.g. if the requirements are volatile or whether they are rigid) and quality needs (e.g. the level, time and relevance of documentation produced). The successful supplier is likely to be the one who balances the strengths of both development paradigms to the benefit of their customers.

AN INTRODUCTION TO SOME KEY DEFINITIONS

In Agile, the definition of 'done' is:

Agile Working Group: Each sprint starts with a kick-off meeting to discuss what is to be accomplished, namely what user stories will be coded and tested. The QA Professional must ensure that the definition of 'done' during the sprint includes adequate documentation. The adequate documentation expected is to ensure the sprints can recreate their work. The type of documentation required and to stay in the mode of Agile methodology is to document the following: user stories, (design), commented coding, testing and tracing the testing to the user stories. Once the sprint is completed, the final close-out meeting should document what was successfully coded and tested what wasn't and what was moved to a future sprint or discarded.

SCRUM: Software has been fully integrated, tested and documented and is potentially shippable.[11]

In this SCRUM methodology we also see the term Minimum Viable Product referred to, which allows you to test an idea by exposing an early version of the product to the target users and customers. This early product is only the start of the development process; it enables the collection of data and through user verification, provides a basis for the next development phase. Thus by agreeing what 'done' means is very important in deciding when to stop the iterative process.

DSDM: The Timebox (iteration/sprint) close-out session records formal acceptance of all the products delivered by the Timebox. It also determines the outcome of work that was initially planned for the Timebox but was not completed. Options are: consideration for the next Timebox; scheduled for some future Timebox or dropped from the project completely[12].

ABOUT THE DIGIT AGILE WORKING GROUP

The DIGIT Agile Working Group will provide articles that will seek to explain Agile activities, identify their strengths and weaknesses, provide analysis on some of the key challenges, (e.g. documentation) and provide advice on applying Agile within the regulated industry.

REFERENCES

1. Applied Software Measurement, Jones, McGraw Hill, 2008
2. Spiral Development, Experience, Principles and Refinements, Boehm, SMU/SEI-2000
3. Rapid Application Development, Martin, McMillan, 1991
4. DSDM Atern Handbook, www.agilebusiness.org/content/fundamentals-0
5. SCRUM Values, www.scrumalliance.org/why-scrum/core-scrum-values-roles?gclid=CMLfxbWS-tECFYML0wodiJIBEw
6. Extreme Programming, https://en.wikipedia.org/wiki/Extreme_programming
7. Agile Manifesto, http://agilemanifesto.org/
8. Agile software development: adaptive systems principles and best practices. Jain & Meso, Information Systems Management, 2006
9. Get ready for agile methods, with care, Boehm, Computer, 35(1), 64-69, 2002
10. Evaluating Agile and Scrum with Other Software Methodologies, Jones, INFOQ, Mar 20, 2013
11. (https://en.wikipedia.org/wiki/Scrum_(software_development))
12. www.agilebusiness.org/content/timeboxing-0

PROFILES

Richie earned a BS in Biology (Bethany College, West Virginia) and MS Degree in Toxicology (University of Cincinnati College of Medicine, Ohio). He has worked for The Standard Oil Co., Gulf Oil Co. and Sherex Chemical Co. prior to Procter & Gamble. Richie organised, trained and conducted GLP and GCP audits and compliance activities. Currently he is the Regulated System Compliance Leader for P&G. Richie is a co-founder of the Quality Assurance Roundtable – predecessor to SQA – where he was elected president in 1990. He is a member of the Beyond Compliance Specialty Section, Computer Validation Initiative Committee and Programme Committee. Richie is also a member of the RQA's DIGIT Committee and DIA's the Information Quality, Compliance & Technology Community. Research Quality Assurance appointed Richie as a Fellow in 2014.

Barry is a Principal Consultant for Empowerment Quality Engineering, focusing on Software Lifecycle (SLC) improvement and audits. He leads computer systems compliance and supplier audits globally.

He has an MSc in Computing Systems and worked as a software engineer from 1997, performing every role and working in every phase of the SLC, (traditional and Agile). He has used multiple programming and database technologies to write and test software: he has conducted complex, technical testing; from unit through to performance, security and disaster recovery within core network telephony systems.

He moved into the regulated industry in 2003 to take up management roles in CS QA, QC and validation. His process improvement activities resulted in risk based SLC systems, on time delivery and right first time validation.

His experiences provide a unique grasp of technical, quality and people problems associated with delivering robust and compliant software.

He is a member of the DIGIT Committee and a member of the ISPE Data Integrity Project team.