So You Think You're Agile?

Colm O'hEocha[1], Kieran Conboy[1], Xiaofeng Wang[2]

[1] National university of Ireland Galway, [2] University of Limerick,
{c.oheocha2, kieran.conboy}@nuigalway.ie, {xiaofeng.wang}@lero.ie

Abstract. Some agile projects succeed, some fail miserably. Research shows that time does not necessarily cure such ills and there can be many complex underlying reasons. Evaluating the ways agility is supported across three supposedly agile projects reveals a myriad of organizational, human and political issues. Using a novel approach to assess agile projects from first principles, this paper outlines several key findings and recommendations beyond mere compliance to textbook methods.

Keywords: agile methods, adoption, assimilation, experiences, assessment

1 Introduction

When adopting an agile information systems development (ISD) method, organizations will normally select one or more the defined 'textbook' examples such as Scrum or XP. These provide varying levels of prescriptive practices and tools which can be implemented directly and promise team agility as a result. When a team adopts such methods, they normally 'cherry pick' practices and adjust them to suit their project context. There is often little thought of dependencies between practices, and how the use or non-use of one could affect others. For example, how useful is continuous build without an automated test suite? Similarly, the way in which the practices are implemented can vary widely, such as daily stand-up meetings that last an hour and take the form of upward reporting versus a team 'touch base' that takes 10 minutes. Regardless of the practices selected, or how faithfully or effectively they are implemented, the projects tend to be generally regarded in the organization as 'agile' and management will expect to see the perceived benefits such as more flexibility, better quality and faster delivery. However, with many initial adoptions these benefits can be elusive with the result that the value of agile methods and their general perception in the organization can be called into question.

Such early faltering can lead to various 'agile assessment' attempts to try to identify the source of the problems. However, these tend to measure compliance to the 'defined' method, assuming that if everything is implemented as per the documentation it will resolve the problems. This approach fails to take into account the particular organizational context of the implementation, and often will encourage adoption of practices which are defined in the method but may not be appropriate in the particular case. Furthermore, it does not address the manner in which the practices are implemented, which can vary widely from project to project. In this paper we

present the experience of a global financial services firm with a novel agile assessment approach, where the true contribution of each practice to the agility itself, rather than compliance to a defined agile method, is evaluated.

The firm in question has approximately 45,000 employees worldwide. Up to 10,000 of these are IT personnel developing systems to support the business, distributed across multiple sites in the US, Europe and India. With a history of using highly formalized, waterfall methods over many years, and with a strong emphasis on process predictability, the organization has developed programs for CMMi compliance, ITIL adoption and so on. More recently, agile and lean methods have gained traction in pockets of the organization. This led to early, ad-hoc trials of Scrum and XP in some teams. A newly developed proprietary method incorporating many principles and practices from agile methods such as Scrum is currently being piloted in several sites. This adoption is being sponsored as part of a larger 'IT Transformation' initiative and is being driven by the global IT organization. In a collaborative research initiative with practitioners in the company, the authors have assessed three such trial projects located in an Irish office between July and September 2009. All were part of distributed teams, but with the majority of analysis, development and test based in Ireland. One was a 'green field' project with some US members and a small, inexperienced team of five. The other two were larger (10-20) with US and India based members and were part of larger enterprise wide programs.

2 Assessing Agile Projects

Across the three selected projects, the agile method was being implemented differently in each. To establish how agile each project was we chose not to look at compliance to the documented method, but rather look at how each practice supported or inhibited agility. For this we used a 'conceptual framework' for agility which defines the underlying aspects of an agile team such as creativity ad simplicity. Therefore, the agile methods in use could be assessed effectively regardless of the particular practices each project did or did not implement, or indeed how the project implemented each practice. This approach allowed effective comparison of the three projects using three different agile implementations – in effect allowing us compare 'apples and oranges'. Therefore the assessment involved answering the following questions for each project:

1. What are we meant to be doing? This we call the *defined* method
2. What are we actually doing? This is the *method-in-action* [1]
3. Is what we are doing helping us be 'agile'?

We found that the *defined* method is a 'hybrid' combining both formal, deterministic elements from the Rational Unified Process (RUP) and agile elements from Scrum and, to a lesser extent, eXtreme Programming (XP). It can be regarded as an "iterative rigorous process" [2]. It has well defined disciplines and practices, and an overall iterative process within which they are executed. The method is expressed as concrete procedures, guidelines and templates designed to execute the implementation steps of a well defined project. The method is prescriptive in that it defines inclusive rules rather than generative [3]. For example, it defines how

requirements should be documented and how peer reviews should be executed. It could not therefore be regarded as providing only 'barely sufficient process' [4].
We then established the *method-in-action* for each project through interviews with project managers and senior team members. Research has shown that work methods are never implemented exactly as defined, varying by project, team and organizational context [5]. Agile methods generally acknowledge this explicitly, citing the 'tailoring' of methods to ensure effectiveness in specific situations. The *method-in-action* for each project was found to be quite different. Different sets of practices were used, and each of these was used differently depending on development context, team context and rational and political roles the applied method plays. Table 1 below gives a snippet of the different method-in-action in the three projects regarding the iteration planning practice.

Table 1. The different method-in-action in the three projects (iteration planning practice)

Practice	Text book definition	Method-in-action		
		Team A	**Team B**	**Team C**
Iteration Planning	- Define scope, tasks & tests for the team. - Design Iteration Stories. - Team owns estimates. Estimates in finer detail.	- 3 week iterations are used. - Scope, tasks, estimates and detailed design are completed by each track before iteration planning meeting. - Planning Meeting is more of a brief review of stories. - Iteration is planned to deliver a fixed number of story points – it is not 'overloaded' with additional stories	- Iterations of 4 weeks are used. - Planning day is used to create user stories and link them to use cases. - Each user story is assigned an owner, who breaks it into tasks & leads detailed design - Early iterations in a release are 'overloaded' with story points to ensure there is always work planned	- 4 week iterations, long enough for largest use cases. - Received ABPs and existing iteration schedule dictate the use cases to be included in the iteration planning. - Joint design of lower level use cases/user stories and breakdown to tasks. - Iterations are planned to complete 100% of capacity – ie no overfilling, even though some level of overfilling has been introduced later on.

Once the method-in-action was established, the third phase involved a half-day focus group session with each team to establish how each practice supported or inhibited agility. From a foundation of organizational agility, and with reference to agile software development, the core contributory concepts of agility have been distilled [6]. Creativity, proaction, reaction, learning, cost, quality and simplicity are the foundations of agility. Table 2 below shows how the three projects perceived the contribution of their version of iteration planning to agility. Depending on how the method is implemented in each project, different project teams perceive differently the contribution of the method-in-action to the overall agility of the team. In the following discussion we take iteration planning as an example to further illustrate this.

In the case of Team A, the iterative planning is regarded negatively in terms of creativity – the iterations of 3 weeks are regarded as "*tight*" to deliver the end to end functionality required for a user centric story. Also, several comments indicate there are considerable story and scope changes within the iteration, which is likely to consume what should be implementation time, and further restricting latitude for creativity. However, proaction and reaction are supported through iterations, though

the need for detailed design and changes within iterations indicates shorter cycles may be beneficial from this viewpoint. One concern is that stories are often not completely finished or 'done done' within an iteration which could reduce the ability to address new circumstances effectively through the iteration practice, evidenced by this comment: "*Tough to start an iteration with a clean sheet. Often some queries or issue from a previous area you worked in crops up which knocks you off*". As with estimation practice, learning can be inhibited due to the same developers being assigned tasks similar to ones they previously completed. Story implementation design is carried out before the planning meeting, with only a review and estimates shared with the larger team. Additionally, learning is constrained by the lack of on-going customer feedback: "*The result of an iteration is sometimes meaningless since customer is not engaged and not testing the deliverable of an iteration*". Initially, iteration planning meetings lasted most of a day and included joint design of stories by the whole team. However, they were found to be long-winded and '*boring*'. Now track leads are asked to perform task breakdown and estimation before the planning meeting, which now lasts less than two hours. This is perceived by the team as a cost saving since all members do not have to sit through the minute of each story. However, deployment of each iteration to QA environment is seen as a significant cost, and one that must be born for each iteration. Together with re-estimation of stories mid-iteration and problems accommodating these changes in the management tools, additional cost is added to the iteration practice. This effect is likely to reinforce the pressure to extend iteration durations, which in turn may exacerbate the overhead of managing them – in effect creating a 'vicious cycle' effect. There was no perceived effect on the quality or simplicity due to the iteration planning practice.

Table 2. How iteration planning is seen to affect agility across the three projects

Practice: Iteration planning	**Agility**						
	Creativity	**Proaction**	**Reaction**	**Learning**	**Cost**	**Quality**	**Simplicity**
Team A	Poor	Good	Good	Poor	Poor	No Perceived Effect	No Perceived Effect
Team B	Poor	Good	Good	No Perceived Effect	Good	Poor	No Perceived Effect
Team C	Conflicting Opinions	No Perceived Effects	Poor	Good	Poor	Conflicting Opinions	No Perceived Effects

-

In Team B, contrary to the defined method, iteration planning appears to be exclusively dedicated to firming up estimates and delivery expectations from the iteration. The detailed design is either performed by the tracks individually before the meeting, if the user story is understood, or a "*placeholder*" is used if not. A firm commitment of deliverables is given to program management at this stage, "*expectation is set at the start as to what features will be delivered*", with failure to deliver as planned viewed negatively, "*customer wanted the story points to match up with functionality delivered, it was a big issue if it didn't match*". In attempts to avoid such shortfalls, project management front load the release to deliver more than the

teams sustainable capacity of story points in early iterations, thereby creating a '*buffer'* to absorb unforeseen delays later in the release cycle. Two contrary views on the effect of this on creativity are expressed. The first calls for longer (e.g. 2 days) iteration planning which "*would help in triggering the learning and creative thoughts in team*" and "*all team members participate and focus is on finding creating/innovative solutions for stories*". But another comment claims "*creativity is helped here by limiting time to define & deliver solution*". The method as defined calls for detailed design to be done at the iteration planning stage which aids with accurate estimation and occurs in a team setting before the iteration deliverables are committed and the 'clock is ticking'. This context may provide more scope for alternate approaches to be solicited and evaluated than the time-boxed iteration tasks allow. Where "placeholders" or "scope-less stories" are concerned, detailed requirements are not understood until the individual tasks are being executed within the iteration – at this stage estimates and deliverables have been committed which may again limit opportunity for creativity. Another concern with iteration planning is suggested by the comment "*too many stories to be closed out at the end of the iteration can have a negative impact on quality*". According to one comment, the ability to be proactive and reactive is enhanced for "scope-less stories" since these are not designed until mid-iteration, just before they are implemented; that is 'just-in-time' design. Interestingly, there were no perceived effects on learning. Progressing through the planning, design, development, test and deployment tasks might be expected to offer a strong learning opportunity. However, it is possible that these effects were attributed to the estimation practice. Initially, planning meetings were a full day for the entire team and this was regarded as a high cost – the length, and perceived cost of these has been reduced. However, the work of design, task breakdown and estimation still must take place – but only the people directly involved in implementation do this before the planning. Therefore, this cost could be considered to still exist but has been displaced from this practice. Another factor is that all team members do not contribute to these tasks for all user stories – this may also reduce the real cost of this exercise, but to the detriment of creativity and learning.

In the case of Team C, the team have different opinions on the impact of this practice in terms of creativity. Since the whole team get together for the planning day, with "*war room allocated*" and "*shared network*", the team members get good opportunities to discuss issues and tasks, "*think of new ways and better ways to do things*", and thus be more creative. There was a perceived negative impact on reaction. One developer commented that the ability of responding to change may be compromised if the plan was "*treated as in stone*", especially by the project management. The fact that the iteration plan already exists before the planning meeting may be the factor that influences the attitude of the project management towards the plan, and eventually impedes the team's ability to respond to changes. Iterative planning turned out to be a good learning experience for the team on how to "*gauge work*". As one developer comments, the team's ability to plan has been improved and they get more accurate estimates from iteration to iteration, which may lead to higher quality of resulting plans. However, since the team use 4-week iterations, typically iteration planning is done for 4 weeks, which is not easy and makes the planning day very busy and intensive. Quality of resulting plans may be

hampered when people are hurrying to get the big planning done in one day. The team members feel that it is a huge cost to spend a full day on planning, basically due to the overhead involved with the project tracking tool associated with the method. Increasing effort such as loading estimates and stories and maintaining the tracking tool takes more time than necessary, and the team felt it impacted negatively on simplicity.

3 Findings and Recommendations

Although considerable data was collected for each of the twenty two defined practices in the method (as per the iteration planning described above), due to space limitations we can only provide a summary here (due to confidentiality concerns of the company please contact the authors directly for further access to detail data from the study). Analysing input from across the three project teams, a number of common 'themes' emerged. Three of these major areas are discussed here, along with actions being taken to improve them. The recommendations have led to improvements in the three projects, but more importantly, in the enterprise wide agile adoption program.

3.1 Iterative Development is not Agile Development

Performing planned but iterative development does not equate to agile development. The method studied here is a variant of the Rational Unified Method (RUP) and combines up-front planning with iterative development. This is sometimes described as 'Serial in the large, iterative in the small' and is often justified as an enterprise scalable approach to agile development. It includes up front commitment to a release plan with major features agreed with the customer, and detail to be added later. However, the method cannot be considered highly agile, even though it does allow for the iterative delivery of applications. This is reflected in developers comments such as "*Feels like we're doing mini-waterfall instead of agile*" and "*Agile development, waterfall everything else*".

A fundamental concept in agile methods is an effective feedback loop – where plans are frequently evaluated against current reality and adjusted accordingly. In the projects studied, iterations did deliver software, but not necessarily working software whereby customers could interact fully with it. User stories often required coordination of several tasks across various 'component teams' and the hand-offs and synchronization involved meant end-to-end functionality could not be completed in a single iteration. Therefore, reflection on the iteration was normally confined to the development team rather than involving all stakeholders, and adaptation was therefore limited.

As implemented, the proprietary method lacks an effective feedback loop. Customers are not involved in the process on a continuous basis, developers are pressured to comply with original plans and schedules rather than adjust them based on current experience, and even feedback between dependant projects in the same program are not synchronised. This lack of ongoing communication intensity leads to

a reversion to ‘management by plan’, which, in turn, severely limits agility of the method.

To tackle some of these problems, several recommendations were made. The intensive face to face ‘visioning’ and planning session used at the start of the project, although getting initial development off to a great start, is no substitute for ongoing customer involvement. This ‘group solve’ process [7] involved all stakeholders in extended, co-located and facilitated workshop sessions over a period of six weeks which served to form relationships across the team and define and prioritize requirements. Such intensive, face-to-face communication should be made a mandatory step in the initiation of any major project. However, this must be followed by on-going, rich (ideally face to face, but at minimum video based) communication between stakeholders, especially customers. Such an ongoing arrangement could mean a shorter and less costly initial planning phase. The cost of keeping stakeholders aligned would be spread throughout the project, rather than focused in a single intense effort. The time and resources to facilitate this critical stakeholder feedback must be built into the project plan. Senior management must understand the necessity and value of this practice and ensure it doesn’t lapse later in the project. In addition, the root causes why user stories cannot be shortened should be investigated and debated. Are component teams the best organizational structure if agility is the end goal? End of iteration customer checkpoints should be made mandatory, and only ‘done done’ stories should be demonstrated.

Where the project is a minor development, the use of a Project Charter type document jointly developed and owned by the various stakeholders is a cost efficient, though not as effective, alternative to establishing a baseline for the project. This document should include business objectives, how business value from the project will be measured and communicated, and a high-level release plan and associated themes. However, ongoing, effective communication with customers is still essential.

To underline the importance of embracing change in agile projects, the change 'control' boards for the projects should be renamed to change ‘facilitation' boards or another title that doesn’t cast change in a negative light. This would support the agile manifesto principle whereby we should “harness change for customer competitive advantage”.

3.2 Focus on Value Delivered, not Effort Expended

Planning on the projects was focused on estimating Level Of Effort (LOE) and creating a plan accordingly. The role of project manager was little different from the waterfall approach with establishing and driving the plan still very much in evidence. Story points are based on the time taken to complete work. Tracking of the project progress is based on the number of story points completed. In one project, management insisted that the iteration deliver exactly (or more) the story point capacity of the team.

This approach leads the team to focus on the cost of delivery, rather than the customer value being delivered. This adversely affects several important agile principles. Delivering early & continuous value flow through short iterations of working software is important for maximising value, but has little affect on LOE. The

tenet of ‘Quality is not Negotiable’ is undermined in preference to maximising the scope delivered, and hence story points. In Lean thinking, the creation of WIP is discouraged as it consumes effort, even though it creates no value. In one project, so-called ‘administration stories’ are created by the team to cover tasks such as code reviews. These were initially included as a task in the user story, but it was found to be more efficient to review several stories in one meeting. So to allow story points for the work done on these before they were code reviewed, the task was moved into an administrative story. Although it may be easier to manage, this encourages WIP and ignores customer value flow concerns.

Another finding was a perception by some that the method was being used to ‘micro-manage’ development. Senior developers and project managers are requested to provide initial estimates at project and release planning stage. These are based on limited information of requirements or context. Project plans are drawn up based on these estimates and agreed with senior management and customers. During development, when reality does not reflect these plans, it is the responsibility of the developers to justify the divergence. Some comments on this topic included “any deviation from estimates has high visibility with restrictive results for the team” and would be seen “in a negative light” Team members felt this leads to pressure on development, and unnecessary overhead and stress when variances have to be explained.

It is difficult to see why the initial plans, based as they are on limited information, and with little buy-in from those performing the work, should be treated as the benchmark for the project. Interestingly, several senior developers emphasized with pride how their estimates had become increasingly accurate throughout the project. The attention given to the estimates indicated their high importance to the team and likely underscores how they are perceived as a measure of performance.

Since all projects faltered in getting a viable customer feedback loop in place, it is easy to see that measuring effort expended is easier than value delivered. This in turn drives ‘efficiencies’ such as grouping code reviews for several stories into a single meeting and making user stories large enough to allow developers get a ‘good run’ at a certain area of code. It also undermines the imperative to automate testing as this becomes an ‘occasional’ rather than ‘continuous’ activity.

Recommendations to tackle these issues included refocusing on user stories that are customer centric and deliver the smallest feature of value to the customer, delivering ‘done done’ stories from each iteration, driving training and resources into automated testing and establishing transparent and common test coverage metrics across projects.

‘Epics’, ‘themes’ and ‘stories’ should only define the bare minimum of detail required at the time, acknowledging that changes will occur as the project progresses and premature detail will be a wasted effort. These agile constructs should also express requirements in terms of customer value rather than application functionality and are fundamental to achieving continuous, early value flow and implicit traceability of requirements to implementation. Move from up-front, contractually-oriented scope definition to a more collaborative, scope-variable approach where shared ownership and responsibility are the norm. Innovation starts with requirements, and elaboration should include diverse perspectives and skills including end customer, developers and testers.

The ability to continuously integrate and automatically test all sub-systems of an enterprise solution should be invested in as a critical IT competency. CI is a cornerstone of agile development and must be recognised as such by senior management, with necessary resources provided to arrive at effective, re-usable technologies and to provide for their implementation in any substantial project.

Establish a mechanism to measure test coverage that is common across projects. For unit tests, develop guidelines on how coverage should be measured (lines of code, method calls, functions, boundary conditions, etc) and coverage targets. The measure should aim to ensure priority is to test critical code and to avoid writing test code merely to meet coverage targets. Where targets are less than 100%, justification for this should be required (if the code isn't used – remove it). Similarly, for acceptance tests, coverage of user stories should be measurable in a consistent manner.

3.3 Agility Needs More than Agile Development Practices

The introduction of agile software development in the case organization focused on the new method and associated practices. Other organizational and people aspects received scant attention. Traditional roles such as project manager, team lead, developer, tester, analyst and the demarcation of responsibilities they represent still persist. Attention to individual capabilities and diversity of teams is not evident. There is little evidence of self-organising teams, or evolving from manager to facilitator roles. Tactics such as rotating team members between tracks, roles and projects to support diversity and cross team learning have not been adopted. Objective setting, performance reviews, training and other HR related activities do not seem to reflect the move to a new way of working.

In addition, the agile way of organizing work seems confined to the software delivery teams – portfolio and program management continue to work to predefined plans. One of the projects studied involved a component team building services to be called by front-end user applications which were being developed by other IT groups. Although these groups were nominally using the same ISD method, coordination between the projects was by plan rather than an agile reflect-adapt feedback loop. As reality impinged upon the project, the synchronization plan became irrelevant. The feedback loop between the component team and the front end feature team could not be maintained. As both teams belonged to different IT organizations, a very 'cautious' relationship developed as neither wanted to appear to fail to execute to plan. Some 'arms-length' solutions such as one project running an iteration behind the other were attempted but these do not seem to have resolved the difficulties, and have led to 'mini-waterfalls' in some cases. This failure indicates a need to apply more agile and effective co-ordination at the level of the portfolio, product or program.

A re-examination of the roles in the project was recommended, including a move from management to facilitation and the development of self-organizing teams. From simple measures such as rotating the role of facilitator in team meetings, to a re-examination of the role of the project managers and a move away from component based to product, project or feature centric organizational structures were recommended. To increase information redundancy and thereby increase team cohesion and resilience, job rotation within long life teams were advocated. Task estimation should be a team activity, using planning poker or a similar technique,

rather than being the preserve of the ‘expert’ in the technical area involved. To encourage continual learning, tasks should be allocated as people become free, rather than on the basis of expertise, which reinforces the development of silos and indispensable ‘heroes’.

4. Conclusions

This study used the method-in-action framework to characterize how an agile method had been implemented differently across three projects. Although all three were regarded within the organization as using the same agile method, there were significant differences in how it had been implemented. By evaluating each project against a set of underlying agile concepts on a practice by practice basis, we were able to assess how each supported or inhibited agility. This approach revealed that, although adopting the agile method had led to improvements in certain areas, it could not be regarded as highly agile. An incomplete feedback-adaptation loop, a focus on effort expended rather than value delivered and a lack of attention to people and organizational structure aspects severely limited the agility of the method. The ‘cherry-picking’ of practices without due consideration of how they inter-relate, along with wide variances in how each practice was implemented, led to an agile method adoption delivering little agility. Agile adopters need to focus on achieving underlying agility by carefully choosing and implementing practices, but also looking at people, roles and organizational structure, as well as dependencies between practices and how they work together.

References

1. Fitzgerald, B., N. Russo, and E. Stolterman, Information Systems Development: Methods in Action. 2002, London: McGraw-Hill.
2. Ambler, S. Choose the Right Software Method for the Job. Available from: http://www.agiledata.org/essays/differentStrategies.html.
3. Highsmith, J. and A. Cockburn, Agile Software Development: The Business of Innovation, in IEEE Computer. 2001, IEEE.
4. Highsmith, J., Agile Software Development Ecosystems. 2002, Boston, MA: Pearson.
5. Madsen, S., K. Kautz, and R. Vidgen, A framework for understanding how a unique and local IS development method emerges in practice. European Journal of Information Systems, 2006. 15: p. 225-238.
6. Conboy, K., Agility From First Principles: Reconstructing The Concept Of Agility In Information Systems Development. Information Systems Research, 2009. 20(3).
7. Takats, A. and N. Brewer. Improving Communication between Customers and Developers. in Agile Development Conference (ADC’05). 2005. Denver: IEEE Computer Society.