Aligning Adoption Theory with Agile System Development Methodologies

Mark Toleman
Mark.Toleman@usq.edu.au

Mustafa Ally
Mustafa.Ally@usq.edu.au

Fiona Darroch
Fiona.Darroch@usq.edu.au

Department of Information Systems
University of Southern Queensland
Toowoomba Qld 4350 Australia

Abstract

Studies show that many software developers are reluctant, for a variety of reasons, to employ system development methodologies (SDMs) in the course of building their applications. The proponents of Agile methodologies suggest that many of the factors that have inhibited the use of SDMs to date have largely been addressed in the underlying principles of Agile methods. Using adoption theory as a basis to inform, this paper contextualises the results of a preliminary study into the adoption process of an early adopter of Extreme Programming (XP). The case study undertaken provides further insights into the factors that have thus far inhibited the use of SDMs and describes the potential for the more widespread adoption and diffusion of Agile methodologies, as a relatively new systems development approach.

Keywords: Adoption, Agile Methodology, Extreme Programming, System Development Methodology

1. Introduction

Avison and Fitzgerald (1995) defined an information systems (development) methodology as 'a collection of procedures, techniques, tools and documentation aids which help the systems developers in their effort to implement a new information system'. There are many such methodologies including the Structured Systems Analysis and Design, Rational Unified Process, Rapid Application Development, Sterling Software's Enterprise Component-Based Development, etc. In addition, many companies develop their own or adapt existing methodologies or simply do not use one at all. The Software Engineering Institute's (SEI) Capability Maturity Model (CMM) identifies five levels of maturity for an organisation in terms of its software development maturity. At level-1 no methodology is in place, while at level-2 there is the beginning of a methodology, and so on up the scale. The arguments for using a methodology are persuasive but many organisations are resistant to change, either to using any methodology or to changing methodology and only one-half of organisations follow any methodology (for example, Fitzgerald 1998; Glass 1999; Roberts et al. 1998).

According to Fitzgerald (1998), practitioners have been reluctant to adopt Software/System Development Methodologies (SDMs), with more than 60% of them abstaining. Furthermore, he noted that nearly 80% of the non-adopters intended to stay that way. While there is a strong belief in their potential intrinsic value to the processes and quality of output of system development projects, SDMs have largely failed to deliver and this is reflected in the adoption rates. Fitzgerald (1998) identified a number of 'arguments' from practitioners against the use of methodologies, and 'pressures' preventing their adoption. It has been argued that the so-called Agile methodologies may provide a solution.

Extreme Programming (XP) (Beck 1999) is perhaps the most well known Agile SDM (Fowler and Highsmith 2001). There are now at least two major international conferences annually (XP 2004, XP Agile Universe 2004) and there have been several special issues of journals on the topic (for example *IEEE Software*, November/December 2001, *Journal of Defense Software Engineering*, October 2002, *IEEE Computer*, June 2003, *Journal of Database Management*, April 2004). The Giga Information Group has predicted that by 2004, Agile processes will be incorporated in two-thirds of corporate IT departments (Barnett 2002). Also, with software development luminaries such as Tom DeMarco (cited in Beck 2001) making statements such as:

> "XP is the most important movement in our field today. ... it will be as essential to the present generation as the SEI and its Capability Maturity Model were to the last"

there can be little doubt this is no passing fad, but in fact a topic worthy of serious research from the information systems (IS) community.

The Agile Manifesto (Manifesto, n.d.) is based on four main tenets, where the items on the left are valued more than the items on the right, viz:

1. Individuals and interactions over processes and tools;
2. Working software over comprehensive documentation;
3. Customer collaboration over contract negotiation; and
4. Responding to change over following a plan.

XP is centred on twelve core practices, also known now as *Xp Xtudes*, which guide the software development process (XP Core Practices n.d.). These practices reflect the sentiment/intent of the twelve principles underpinning the Agile Manifesto (Manifesto n.d.). Most of these practices are not new but the way they are presented as a package in XP represents to many software developers how they really develop software systems (Sleve 2002) or, in some cases, desire to develop software for clients.

XP has been successfully adopted in many projects although it is still not widely known or taught at the tertiary level. Many case-studies have been published that demonstrate the wide variety of situations that have been considered suitable to trial Agile methods. These studies fall into several categories including academic teaching (Mugridge et al. 2002; Lappo 2002), tertiary student projects (Karlstrom 2003), small-scale industry developments (Bossi and Cirillo 2001), and large-scale industry developments (Schuh 2001; At Chrysler, Objects Pay 1998; Elssamadisy 2001, Grenning 2001, Pedroso et al. 2002). However, there has been no attempt to grapple with the factors affecting the adoption of this new methodology.

Rogers (1995) defined the adoption process as 'the process through which an individual or other decision making unit passes from first knowledge of an innovation, to forming an attitude toward the innovation, to a decision to adopt or reject, to implementation of the new idea, and to confirmation of this decision'. Thus, the adoption process is characterised by a sequence of stages through which a potential adopter of an innovation passes before accepting the innovation. Various models including for example the Technology Acceptance Model (TAM, TAM2), Perceived Characteristics of Innovating (PCI), Theory of Planned Behaviour (TPB), and the Model of Personal Computer Utilization (MPCU) have been proposed to model individual intentions to adopt various information technology artefacts. However, the suitability or otherwise of these models, or indeed the reporting of new models, for examining the intention to adopt system/software development methodologies has received relatively little research (Roberts et al. 1998; Johnson 1999; Roberts et al. 1999). Riemenschneider et al. (2002) in fitting the five models above noted

that although the models were resilient, there were important differences in relation to the pattern of significant factors affecting adoption of tools versus adoption of methodologies.

Clearly there is still scope for further investigation of the driving factors in methodology adoption including identification of factors not already considered in existing models and, indeed, potentially new models of adoption. This research is contributing to this by examining adoption of a relatively new methodology in a specific environment. The extent to which Agile methodologies might address the shortfalls in methodology uptake is examined as are the characteristics that influence adoption of a particular methodology. It is intended to discover express and implicit preliminary factors influencing the adoption of XP as a software development methodology, and hence support propositions that give added value to practitioners planning to embark on a similar action.

The paper is structured as follows. The next section outlines the background to the case study. This is followed by a description of the methodology used in this study, then the results and discussion and finally future research directions and conclusion.

2. Situation Background

NextEd Limited is a Hong Kong based provider of web-based software infrastructure. It services mainly tertiary education providers in the Asia-Pacific region, including the University of Southern Queensland, by providing platforms for delivery of study materials and communication services to students who study, principally, in online modes.

The project discussed in this paper required the development of a suite of tools for a scalable, flexible, and efficient continuous publishing system. The tools facilitated the generation of print and web-based study materials provided by content experts. The target operating system was Windows NT and the languages used included Delphi, XSLT and XML. Visual Source Safe was used for configuration management and although not ideal, proved effective.

The newly formed project team felt that an iterative methodology was most suited, and according to one of the informants it was fundamental to the project to produce 'a constant stream of outputs and engage the customer on a regular basis'. Initially made aware of XP by the organisation's Chief Technical Officer, the team took on the initiative to study this approach to software development and considered the project a suitable candidate for its use. No particular development methodology for this type of project was in place in the organisation. There was also recognition that management's requirements of the project were not well defined, and that the project size was not expected to be large (a few thousand lines of code), and its development time was expected to be relatively short (about six months). XP provided an alternative to a traditional heavyweight approach since there was a small team and less need to follow a process-oriented methodology. Being a small team meant members had multiple roles (project leader, proxy customer, system architect, and programmer). Management was not particularly concerned with the product development approach adopted but was concerned with the outcome and monitored the progress accordingly. They did not put limitations on the trial of XP but noted that XP required the developers to regularly deliver software which could be trialled and tested by the customer.

3. Research Methodology

A combination of two research techniques was employed in this study, viz. a case study based on in-depth interviews and a focus group. Cavaye (1996) points out that a single case can enable researchers to examine a phenomenon more deeply, resulting in richer information.

Furthermore a single case approach may be used to test a theory, and this has been done in this study as a useful start to examining the efficacy of various adoption models in explaining the adoption of Agile methodologies (Cavaye 1996). Case-studies provide a mechanism by which trialled methods and ideas can be observed. Other organisations or individuals, in particular practitioners considering the adoption of such methods, are provided with a view of an implementation through which they can make their own assessment of the advantages and disadvantages. Although the success or otherwise of a project and the application of the new method may be subjectively assessed by the report's authors, case-studies still provide insights not otherwise available. In fact the importance and merit of case-studies has been openly recognised by the editorial board of *IEEE Software*, with its decision to expand the space made available to them (*IEEE Software*, November/December 2001). The details presented in this paper are unique in that most reported industry case-studies are written from within the development team, and therefore inherently subjective. In contrast, this report is an objective analysis undertaken with the express purpose of furthering research on the use of Agile methods and by researchers who were external to the development.

The use of qualitative methods to study software development issues allows the researcher to study selective issues, without the pre-determined constraints of 'categorised' analyses. To develop a deeper understanding of the complexities associated with the relatively new concepts of Agile methodologies, and XP in particular, in-depth interviews were conducted with all members of the case study project team. The software house was chosen because it was considered to be an early adopter of this software development approach. The interviews formed the main source of evidence for the case study. While an interview guideline was used as a basis, the interviewees were allowed to digress when other interesting issues surfaced. Interviews were transcribed and reviewed for accuracy by the interviewees, thus enhancing the internal validity of the researchers' accounts of the interview in accordance with Yin (1994). Quotations or indented text presented throughout the rest of this paper are either verbal or written statements from these primary data sources (denoted in the text as N1, N2, and N3).

A focus group was also conducted as part of this study. The use of focus groups is an exploratory technique that is usually quick, inexpensive and flexible (Zikmund 2003). Many of the issues raised by Healy and Perry (1998) (such as site selection, collection of preliminary information and problem definition) which relate to the establishment and conduct of the session were addressed during this study. As in Healy and Perry's 1998 study it was useful to test the appropriateness of the theoretical framework arising from the literature review. The twenty members of the focus group participated on the basis of their experience and interest in software development, and the use of software development methodologies. One of the authors facilitated the session, while another recorded proceedings and detailed notes. The session explored the characteristics of Agile system development methodologies in terms of the five theoretical constructs arising from Rogers's (1995) work on adoption theory. Statements attributed to the focus group are denoted in the text as F1.

The adoption of a multi-method approach as described by Creswell (1994) has enabled this study to take advantage of triangulation to counteract any individual method's bias. In this situation the focus group data acted as a balance to the case study interview data when examining Rogers's (1995) adoption theory.

4. Results and Discussion

The project undertaken by NextEd had several distinguishing features:

- the system under construction was not a typical business application, but a software infrastructure development with difficult to define, abstract requirements;
- the complexity of the system development environment required the use of multiple software products for development;
- much of the current debate on using Agile methods centres on whether it is developers or management who resist their adoption. The situation under review was notable in that the impetus initially came from management, but the development team were also very keen to trial the XP methodology;
- this trial was conducted without any expenditure on mentoring, training, etc. – it was all based on internal research;
- the implementation of XP (according to the developers) was a 'success story'.

Table 1 shows a summary of the level of adoption of the XP core practices for this project.

Table 1 XP core practices experience summary: Full=full adoption, Partial=partial adoption, Nil=not adopted.

XP CORE PRACTICES	IMPLEMENTATION	COMMENT
The Planning Game	Full	Worked well for both developers and client
Small Releases	Full	Successful
System Metaphor	Nil	Developers would like this
Simple Design	Full	Successful
Test Driven Development	Full	Very beneficial for development
Design Improvement (was Refactoring)	Partial	No tools and not regular
Pair Programming	Partial	Useful for developers to cross-train
Collective Code Ownership	Full	Very successful for developers – aided skill transfer
Continuous Integration	Full	Successful – infrastructure can be reused
Sustainable Pace (was 40-Hour Work Week)	Nil	Would be desirable for developers
Whole Team (was On-site Customer)	Full	At least during business hours
Coding Standards	Full	Worked well

The discussion that follows is an analysis of both the case study and the focus group within the framework of the relevant adoption theories.

4.1 Inhibitors to Adoption of System Development Methodologies

According to Fitzgerald (1998), practitioners have been reluctant to adopt SDMs resulting in more than half of them abstaining. Furthermore, it was noted that 79% of the non-adopters intend to stay that way. While there is a strong belief in their potential value, SDMs have largely failed to deliver and this is reflected in the adoption rates. Fitzgerald (1998) also identified a number of 'arguments' from practitioners against the use of methodologies, and detailed the 'pressures' preventing SDM adoption. A discussion on the extent to which these shortfalls have been addressed by the Agile methodologies follows.

One of the problems of SDMs is that they treat the process of software development as an orderly, rational process. Fowler (2003) challenges the notion of the engineering metaphor as a basis for 'predictive' software development and proffers an alternate view of it as an 'adaptive' process that recognizes the reality of uncertainty as part of the requirements and

design processes. One of the highlights of the case study experience was that with the customer choosing the functionality (user stories) to be developed, as well as setting the priority of each cycle, flexibility became a positive for both the developers and the clients. A further problematic aspect identified was that SDMs traditionally focus on technical rather than social issues. Agile methodologies address this issue in one of the aforementioned tenets where individuals and their interactions are more valued than processes and tools (Manifesto n.d.). This sentiment was reflected in the case study:

N1: '...it's about ... right mix of people so you can change the tone of your team'.

N3: 'a very human-oriented process ... in the end you are selling a belief system'.

Another argument against the adoption of SDMs is that the importance of the actual system being developed is often subjugated by the processes of the methodology (Fitzgerald 1998). Again, this is countered by another of the Agile tenets where working software is valued over documentation. In the case study situation, processes were changed when they were seen to be hampering the process of delivering production software, as seen in the decision to modify the gathering and publication of requirements.

Traditional SDMs are seen to be developer-unfriendly (Fitzgerald 1998). On the other hand, Agile methodologies are people-centric, as evidenced in one of the Manifesto's aforementioned twelve principles, which encourages motivation of individuals to build projects with appropriate environments and support. This notion of Agile methods embodying practitioner-focussed ideals was reinforced by the case study experience where there was a surge of interest by other local developers in the case study project.

It is also claimed that many SDMs have been put forward without sufficient empirical evidence to support their efficacy (Fitzgerald 1998). This issue was well understood by the architects of XP, who basically founded the methodology on what highly regarded, experienced developers identified as a proven collection of 'practices' that would support the timely delivery of quality software, 'We are uncovering better ways of developing software by doing it and helping others do it.' (Manifesto n.d.). However, none of the interviewees felt that empirical evidence was central to their decision to adopt, rather that the benefits of XP appeared to be obvious and that it was a commonsense approach:

N2: 'I simply found the explanation of XP sensible and fit for the purpose... '.

N3: '... the tools and techniques seemed to make good sense on their own'.

The inability of traditional methods to deliver production software in a timely manner is another serious concern raised by Fitzgerald. Agile methodologies focus on working software, and this is exemplified in many of the XP practices such as in the planning game, and in the practice of small releases. It is also well illustrated in the case study where regular releases built higher levels of trust and credibility between the client and the development team:

N1: '...they're happy because you always deliver on time'.

N2: 'This aspect was absolutely critical to the success of the project. ... a real customer using the system within months of beginning the project'.

4.2 XP Adoption Stages at NextEd

Theories of innovation and innovation diffusion seek to explore and explain why some new technologies/processes diffuse quickly and widely, while others do not. As a socially constructed process involving the development and implementation of new ideas, Rogers's (1995) innovation theory is a useful model to examine the adoption and diffusion of XP. Rogers (1995) defined the adoption process as having certain stages. His model's five 'Stages of Adoption' adequately supports the experience within the subject case study:

- Knowledge/awareness – the team had been informed about XP from the organisation's Chief Technical Officer.
- Persuasion/interest - the team consequently read about and examined the Agile approach to software development.
- Decision/evaluation - no particular development methodology for this type of project was in place in the organisation:

  N1: '... they didn't seem to have any kind of formal (methodology) ... Every job was basically working from scratch ...'.
- Implementation/trial – the team considered this project to be a suitable candidate for its use, also because development time was expected to be relatively short (about six months).
- Confirmation/adoption – any decision relating to the further adoption of XP at NextEd will depend on the nature of the project and the development team structure. However, in response to the question 'Do you see the adoption of extreme programming in the industry?' N1 echoed Tom DeMarco's sentiment: 'I can in my career'.

In Rogers's categories of adopters, based on degree of innovativeness, the subject case study could be regarded as an 'early adopter'; early adopters are typically pilot users who bring a novel technology to a realistic and strategic context.

4.3 Contextualizing XP As An Innovation

Rogers (1995) identified a framework of five characteristics of an innovation or change that would influence the adoption decision or encourage adoption viz. the relative advantage, complexity, compatibility, trialability and observability of an innovation. In this section the twelve core practices of the XP methodology are mapped (see Table 2) against this framework along with supporting evidence drawn from the case study and focus group as a means of exploring the adoption potential of XP.

Table 2 Roger's (1995) Innovation Characteristics and XP Core Practices.

Characteristics	XP Core Practices
Relative Advantage	Small Releases; Design Improvement; Pair Programming; Collective Code Ownership; Continuous Integration; Sustainable Pace
Complexity	The Planning Game; Simple Design; System Metaphor
Compatibility	Small Releases; Continuous Integration; Simple Design; Coding Standards
Trialability	All practices
Observability	Whole Team; Test Driven Development; The Planning Game

Relative advantage refers to the extent to which the potential adopter perceives that an innovation is superior to alternative products, services or concepts (Rogers 1995, p.212). In terms of SDMs, developers would have to view XP as a superior alternative to the traditional methodologies. This will give them a competitive edge by responding to customers' constantly changing requirements and improving software quality. The adoption of XP was seen to be an advantage in the case study:

N2: '... an agile development methodology made it run much more smoothly. However these ingredients would go a long way in the success of any project'.

Overall, the members of the focus group considered that XP offered relative advantages over traditional methodologies.

F1: 'I think it's very important to a manager. .....if they can track where they're going in the project and then being able to make a decision at the end of each cycle as to whether to continue on or stop. I think it is very important'.

There are several XP practices that demonstrate a relative advantage, such as:

- **Small Releases** whereby working software is released to customers regularly, usually weekly, but potentially daily. In the case study, after the initial build process (of about three months), the release cycle was fortnightly. This was considered advantageous because it was much easier to identify whether the project was on schedule. This is in contrast to traditional SDMs which tend to focus on delivering larger chunks of functionality much later in the development schedule:

  N2: 'Incremental progress and updates ensured everyone who was interested knew where we were and why completed or failed to complete certain tasks'.

  N3: '... the key is regular releases of working software and along with that getting people using a product from as early as possible....'.

  N3: '... it is better to get [sic] the bugs out early than to release all the bugs at once'.
- **Design Improvement** (was **Refactoring)** is the process of continual improvement of the code as the developer's understanding of the system grows. Since functionality is not altered during refactoring, all tests should still operate effectively. Refactoring was applied in the case study, but not using automated tools. It was a manual process of identifying and rectifying areas of code that had potential for improvement and removal of duplication. This was advantageous because it encouraged developers to improve their system designs. There is no such equivalent practice in traditional methodologies that tend to indulge in 'big upfront design' setting the application architecture early, making it relatively inflexible.
- **Pair Programming** is where developers work in pairs using one machine, continually reviewing code. Pairs rotate throughout the project. This practice was sometimes followed in the case study:

  N1: 'A lot of the time we did operate this way but there were many times when we worked alone'.

  Pair programming was used to address particular problems, and to aid other developers' understanding of certain procedures or parts of the system. It also acts as a contingency for the loss of key development staff. Traditional methodologies do not support this type of productive exchange and review of coding, and the nearest process is that of code walkthroughs, a form of Quality Assurance. Unfortunately walkthroughs only identify problems after the code has been developed, and are typically abandoned as soon as schedules become tight. Furthermore there is no risk management explicit in traditional methodologies to defray the exposure to the loss of key technical staff.
- **Collective Code Ownership** means that code is not owned by individuals. Any code may be changed provided it is done by pairs of developers, complying with coding standards and subject to a satisfactory run of all tests. In the case study, all developers were free to work on any code. This assisted in building the expertise of all involved in the project and was a particularly successful aspect of the project from a developer's perspective. This approach is not consistent with the traditional methodologies, and as such there is a much greater likelihood that individual styles would result in inconsistencies across the system. Also, the inherent characteristics of the Object Oriented (OO) software development methodology facilitates code sharing and component reuse, and as the OO paradigm becomes more pervasive the need for such mutual co-operation should become even more compelling.
- **Continuous Integration** is the process of adding unit tested code to a repository where the system is rebuilt and all tests are run. This may occur several times a day so that the team has access to the most up-to-date version of the system. Within the development environment of the case study the integration of new code into the project was a natural process, with system builds and all automated unit tests conducted every time code was checked into the source repository. A batch system controlled the build process

including compilation and testing, and notified the developers, by email, if errors occurred. In the traditional methodologies, the usual trend is to conduct integration testing during the latter phases of the project resulting in problems being detected much later, and resulting in more severe project impacts (the later a problem is detected the higher the cost of addressing it). While some initial effort was required by the development team to create an environment supportive of XP the view taken was that once established it would form the basis for other systems and make maintenance much simpler.

- **Sustainable Pace** (was **40-hour Work Week**). To avoid burnout so common in the IT industry, developers are restricted to about 40 hours of work per week. This also improves the accuracy of time and resource estimates for the development effort required. The developers in the case study did not comply with this practice. Traditional methodologies have not explicitly addressed this issue, and realistic resource allocation is another persistent problem with traditional approaches.

Complexity refers to the extent to which an innovation is perceived as difficult to understand and use (Rogers 1995, p. 242). In terms of SDMs, this means that developers would have to view XP as more easily understood than the traditional methodologies. There were mixed responses from the focus group:

> *F1: from 'I found the whole thing very sensible and easy to understand, you know, get the major concept behind it very quickly' to 'More complex to manage'.*

There are several XP practices that relate to the issue of complexity:

- **The Planning Game**. System requirements are developed and decisions about which functionality to implement happen each cycle in what is 'metaphorically' described as a game. The customer decides on the functionality to be implemented while the developers estimate the time required, and through negotiation, work for the next cycle is planned. In the case study, the customer was involved in selecting the business functionality required for implementation. The customer knew what would be provided, the project team provided it, and management could see the progress being made. Hence the customers (and management) were able to drive the system development but through the process of setting priorities, the project team felt it had some control too:

  > *N1: '...which meant that we are driving the development of it, we are forcing our company to drive.... The other advantages are when you have this prioritised, your customers are going to say well I really only need that, I did need it ten minutes ago but it is not that important now ... '.*
  >
  > *N3: 'Yes, it is less complex. Mainly in the area of planning. Typical planning processes for software development are pure fiction. A lot of upfront effort goes into creating charts and dependencies graphs, but I have never seen a plan like this actually followed up or kept up-to-date'.*

- **Simple Design** emphasises only implementing features indicated as important by the customer. Keeping the design simple means that change, as and when it is required, is less problematic. At all times throughout the case study, the developers avoided unnecessary complication with respect to software architecture and coding, staying with the stories agreed to each cycle with the customer. In this way the team took a minimalist approach to the addition of functionality and ensured the customer received essential features in the order of priority requested. In traditional methodologies, design architecture is usually predefined, which does not offer the same flexible approach.
- **System Metaphor**. The team should have an overarching view or model of the system being developed. At the very least a common vocabulary is required. In the case study, this was perhaps the least successfully implemented core practice of XP. N1's view was

that '... in hindsight that (a metaphor) would have been really helpful because we really struggled to get the idea/concept for the system out of the head of the customer'.

Compatibility of an innovation refers to the extent to which an innovation conforms to the potential adopter's existing values, previous experience and needs (Rogers 1995, p.224). This issue is addressed from two viewpoints. From the organisation perspective, any new change to job content and uncertainty of a new system will meet with organisational resistance (Hong and Kim 2001). Organisation and process changes induced by Agile methodologies are likely to be disruptive to existing practices, and such resistance has the power to inhibit infusion of a new process (Cooper and Zmud 1990). The adoption of agile methodologies runs counter to the traditional practices and resistance may be encountered:

> *N3: 'There is still a culture in some of the managers and project staff that lists of features to be delivered at a series of dates are a valuable management tool.. ... (but) as soon as they lock down a feature set they think they need, they introduce a lot of new risks in an ongoing process. They seem to think that using an agile methodology is some kind of con trick'.*

From the viewpoint of the developer, compatibility is identified as the extent to which developers, drawing from their practical experiences, considered XP as reflecting their own understanding of how systems should be developed in contrast to having to make accommodations for traditional methodologies. The focus group's response affirmed this:

> *F1: 'For me, yes it was quite natural' and 'I think for me it sort of summarises moreover what you probably already did without actually knowing everything'.*

Some of the practices such as small releases, continuous integration, and simple design, discussed earlier, also relate to the issue of compatibility in this context. Most of the XP practices are not new, but their presentation as a cohesive package reflects many software developers' actual or desired approach to system development, for example:

- **Coding Standards** must be rigorous and are essential since code may be worked on by any programmer at any time. In the case study environment a coding standard was developed during the project and is used currently within the organisation. It is largely language-independent. Coding standards have also long been incorporated into projects run under traditional methodologies.

Trialability is the extent to which an innovation can be tried on a limited scale (Rogers 1995, p. 243). In terms of SDMs, this means that XP must be suitable for pilot studies. As reported earlier in this paper, XP has been successfully trialled in many projects, large and small, in industry settings and for academic study, as well as the currently described case study. Nevertheless, traditional methodologies have also demonstrated a capacity for trialling. In the focus group, the ability to trial XP was considered important:

> *F1: 'Essential. ....I had to do a trial before even considering. ...presenting it in a pilot we were able to demonstrate how you'd work together, both management and the programmers..... to produce something'.*

Observability is the extent to which the results of an innovation are visible to others (Rogers 1995, p. 244). With regard to SDMs, this means that the results of implementing XP must be visible to the stakeholders, a characteristic which can be observed via many XP practices including:

- **Whole Team** (was **On-site Customer)** in an XP project gives developers continuous access thereby lessening the need for extensive requirements documents. They can ask the customer about functionality, test cases, interfaces, etc, at any time. An on-site

customer was available in the case study project. This represents a very visible difference to the traditional methodologies where customers tend to play a background role.

N2: '... our close contact with most clients requires a certain degree of structure(d) feedback and our version of XP helped us in this regard'.

- **Test Driven Development**. Customers develop acceptance tests while developers write test cases prior to writing code and implementing unit tests to focus on the required functionality. All tests are automated and run whenever changes occur. The case study also implemented this practice. All Delphi code had tests included because a testing framework existed. An XSL testing framework had to be developed since none was available. It was also noted that test development assisted in the code development:

  N1: 'If you cannot write those (test) unit specs up front, then you will fail the test runner ... so writing those sort of tests helps you map out your design in the first place and you get a much better design'.

Nevertheless, writing tests prior to code was a significant change of 'habit' for the developers, generally, and was thus a visible difference for both them and customers who develop acceptance tests. Testing in a traditional setting is much less pervasive and has a much lower profile. There are many other observable elements associated with the implementation of XP reflected in other practices, including the planning game, where the role of the customer is extended well beyond the bounds of a traditional project, to include responsibility for the choice of functionality for each release as well as to set the priorities.

The observability of Agile methods was affirmed in the sentiments of the focus group:

F1: 'We wanted to make it as transparent as we could right (from the start). And that was part of the goal that the three of us worked on'.

The rate of adoption of an innovation depends on the characteristics as defined above. The evidence from the case study and focus group suggests that Agile methods, as a new software development approach, exhibit all of the characteristics likely to foster their more widespread adoption and diffusion.

4.4 Methodology Adoption Intention – Some Factors for Consideration

Recent research (Riemenschneider et al. 2002) to explain intention to adopt a methodology includes a comparison of five models (TAM, TAM2, PCI, TPB, MPCU) previously used in modelling adoption of information technology tools. Using these adoption models, their research found four significant determinants of usage intentions: usefulness, subjective norm, voluntariness and compatibility. However none of the models contained all four factors. In addition, some factors found significant in tool studies (ease of use, perceived behavioural control – internal and external) were not significant for their study of methodologies and three of the four factors found significant in the study (voluntariness, compatibility and subjective norm) are sometimes found significant in tool studies and sometimes not. These three significant factors are now considered in terms of the case study presented here.

Riemenschneider et al. (2002) found that **mandated** adoption does not guarantee successful deployment. In the case study, the use of XP was not mandated but suggested by management and then readily adopted. They also found that when a methodology is **compatible** with developers' work patterns, it is more likely to be adopted. Many aspects of XP appeared compatible with the work practices of the developers in this case study. Indeed there was a view that several of the recommended XP practices mirrored their existing *modus operandi*. Developers take into account the views of co-workers and superiors (**subjective norm**) when making SDM adoption decisions. Staff at NextEd were aware of the positive views held by colleagues and proponents of Agile methods, and XP through personal contact, discussion lists, Special Interest Groups, etc.

The study in Riemenschneider et al. (2002) suggests differences between the domains of tool use and methodology with respect to adoption. The tool adoption models are robust but statistical fits still only accounted for 50-60% of the variance. The range and combination of factors influencing methodology adoption remains a question.

5. Future Directions for Research

There is a need to witness first-hand the use of Agile methods such as XP, if the soundness of these methodologies is to be assessed. This is the also most effective means to mature the methodology. Experience is the litmus test of reality, but it needs to be conceptualised and reported to a wider audience to be of benefit to practitioners, researchers and academics.

When examining any aspect of the software development process, anything other than actual experience is at best intelligent conjecture. Indeed, while there has been a great deal of interest and support from the developer ranks, the IS research community appears to have been a slow and/or reticent to embrace this new direction in software development methodologies. Our future research plans include experiments involving the use or otherwise of XP, further case-studies of several groups and projects implementing Agile methods, as well as the incorporation of Agile methods into the tertiary curriculum.

This study focused on the characteristics of XP as perceived by the subject of our investigation and the process by which XP was communicated within the organisation. However, complete understanding of adoption behaviour requires a model that captures the adoption process across many different contexts (Plouffe et al. 2001). Several models have been proposed to explain the adoption of various information technology tools but there is some question as to whether these models are comprehensive enough to explain adoption of SDMs which tend to be much more complex.

Social psychological issues have been dealt with in a number of behavioural models of adoption. The relevance or otherwise of such antecedent attitudinal constructs as perceived ease of use and perceived usefulness (Davis 1989), visibility, ease of use and voluntariness of use (Moore and Benbasat 1991) as well as image and results demonstrability (Venkatesh and Davis 2000) will be examined in more detail through surveys and further qualitative analysis.

6. Conclusions

The project at NextEd was largely a success story which applied many of the core practices of the Agile methodology, XP. From a developer's perspective it delivered on the requirements, limited as they were in initial detail, to produce a publishing system for documents to print and the web. While the opinion of NextEd management was not sought in relation to their perception of the success of this trial project, an appropriate product was delivered and used as a prototype for their current publishing system.

There has been much debate about the type of projects that are suitable for Agile methods. Practitioner experience suggests that they are particularly suitable for projects where requirements are more abstract and difficult to define, as in this study. It is not surprising that such organisations have either not adopted or moved away from traditional approaches.

This paper contextualised the results of a preliminary study into the adoption process of an early adopter of XP, and then triangulated those results with data from a focus group. A small team successfully adopted XP for web publishing software tools for use in delivery of

education study materials. Almost all XP practices were adopted, although some were adhered to more rigorously than others and some proved to be more successful than others.

By showing they address many of the reported arguments and pressures against the use of a SDMs there is evidence to indicate that Agile methodologies, in conforming largely to Rogers's theories, have significant potential for more widespread adoption and diffusion.

7. Acknowledgements

We wish to thank our developer colleagues from NextEd who gave freely of their time in helping us understand the potential role of an Agile method such as XP.

References

At Chrysler, Objects Pay *Distributed Computing*, October, 1998, viewed 12 Feb 2004, <http://www.xprogramming.com/publications/dc9810cs.pdf>

Avison, D.E. and Fitzgerald, G. *Information Systems Development: Methodologies, Techniques and Tools*, London, McGraw-Hill, 1995.

Barnett, L. "IT Trends 2003: Application Development Methodologies and Processes", *IdeaByte*, September, 2002, viewed 12 Feb, 2004, <http://www.gigaweb.com>

Beck, K. *Extreme Programming Explained: Embrace Change*, Addison Wesley, Boston, 1999.

Beck, K. and Fowler, M. *Planning Extreme Programming*, Addison Wesley, Boston, 2001.

Bossi, P. and Cirillo, F. Repo Margining System: Applying XP in the Financial Industry, 2001, viewed 12 Feb 2004, <http://www.xp2003.org/conference/papers/Chapter35-Bossi+alii.pdf>

Cavaye, A.L.M. "Case Study Research: A Multifaceted Research Approach for IS", *Information Systems Journal*, (6), 1996, pp. 227-242.

Cooper, R.B. and Zmud, R.W. "Information Technology Implementation Research: A Technological Diffusion Approach", *Management Science*, (36:2), 1990, pp. 123-139.

Creswell, J.W. *Research Design Qualitative and Quantitative Approaches*, Sage Publications Inc, Thousand Oaks, CA, 1994.

Davis, F.D. "Perceived Usefulness, Perceived Ease of Use and User Acceptance of Information Technology", *MIS Quarterly*, (13:3), 1989, pp. 319-340.

Elssamadisy, A. XP on a Large Project – A Developer's View, 2001, viewed 12 Feb 2004, <http://www.xpuniverse.com/2001/pdfs/EP202.pdf>

Fitzgerald, B. "An Empirical Investigation into the Adoption of Systems Development Methodologies", *Information and Management*, (34), 1998, pp. 317-328.

Fowler, M. and Highsmith, J. "The Agile Manifesto", *Software Development*, August, 2001, viewed 23 Feb 2003, < http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm>

Fowler, M. The New Methodology, viewed 12 Feb 2004, <http://www.martinfowler.com/articles/newMethodology.html>

Glass, R. "A Snapshot of Systems Development Practice", *IEEE Software*, May/June, (16:3), 1999, pp. 110-111.

Grenning, J. "Launching Extreme Programming at a Process-Intensive Company", *IEEE Software*, November/December, (18:6), 2001, pp.27-33.

Healy, M. and Perry, C. Focus Groups in Academic Research Projects, ANZMAC, Dunedin, 1998.

Hong, K. and Kim, Y. "The Critical Success Factors for ERP Implementation: An Organizational Fit Perspective", *Information and Management*, (40:1), 2001, pp. 25-40.

Johnson, R.A. "Applying the Technology Acceptance Model to a Systems Development Methodology", *Proceedings of the Americas Conference on Information Systems*, pp. 571-573, 1999.

Karlstrom, D. Introducing Extreme Programming – An Experience Report, 2003, viewed 12 Feb 2004, <http://www.xp2003.org/xp2002/atti/DanielKarlstrom--IntroducingExtremeProgramming.pdf>

Lappo, P. No Pain, No XP: Observations on Teaching and Mentoring Extreme Programming to University Students, 2002, viewed 12 Feb 2004, <http://www.agilealliance.org/articles/articles/PeterLappo--ObservationsonTeachingandMentoringXP.pdf>

Manifesto. Manifesto for Agile Software Development, viewed 4 June 2003, <http://www.agilemanifesto.org/>

Moore, G.C. and Benbasat, I. "Development of an Instrument to Measure Perceptions of Adopting an Information Technology Innovation", *Information Systems Research*, (2:3), 1991, pp. 192-222.

Mugridge, R., MacDonald, B., Roop, P. and Tempero, E. Five Challenges in Teaching XP, 2002, viewed 12 Feb 2004, <http://www.cs.auckland.ac.nz/~rick/5ChallengesTeachingXP.pdf>

Pedroso, M. Jr, Visoli, M.C. and Antunes, J.F.G. Extreme Programming by Example, 2002, viewed 12 Feb 2004, <http://www.xp2003.org/xp2002/atti/Pedroso-Marcos--ExtremeProgrammingbyExample.pdf>

Plouffe, C.R., Hulland, J.S. and VandenBosch, M. "Research Report: Richness versus Parsimony in Modeling Technology Adoption Decisions – Understanding Merchant Adoption of a Smart Card-Based Payment System", *Information Systems Research*, (12:2), 2001, 208-222.

Riemenschneider, C. Hardgrave, B.C. and Davis, F.D. "Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models", *IEEE Transactions on Software Engineering*, (28:12), 2002, pp. 1135-1145.

Roberts, T., Gibson, M., Fields, K., and Rainer, R. "Factors that Impact Implementing a Systems Development Methodology", *IEEE Transactions on Software Engineering*, (24:8), 1998, pp. 640-649.

Roberts, T., Gibson, M. and Fields, K. "System Development Methodology Implementation: Perceived Aspects of Importance", *Information Resources Management Journal*, (12:3), 1999, pp. 27-38.

Rogers, E. *Diffusion of Innovations*, 4th edition, New York, Free Press, 1995.

Schuh, P. "Recovery, Redemption, and Extreme Programming", *IEEE Software*, November/December, (18:6), 2001, pp.34-41.

Sleve, G. "Agile before Agile was Cool", *The Journal of Defense Software Engineering*, (15:10), 2002, pp. 28-29.

Venkatesh, V. and Davis F.D. "A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies", *Management Science*, (46:2), 2000, pp. 186-204.

XP2004. Fifth International Conference on eXtreme Programming and Agile Processes, viewed 12 Feb 2004, <http://www.xp2004.org>

XP Agile Universe 2004. viewed 12 February 2004, <http://www.xpuniverse.com>

XP Core Practices (undated) viewed 18 Feb 2004, <http://c2.com/cgi/wiki?ExtremeProgrammingCorePractices>

Yin, R.K. *Case Study Research: Design and Methods*, Sage Publications Inc, Thousand Oaks, CA, 1994.

Zikmund, W.G. *Business Research Methods*, Thomson Learning, Mason, OH, 2003.