NETWORK FOR AGILE METHODOLOGIES EXPERIENCE (NAME)

Research Roadmap.

Abstract*. This report identifies important areas for further research into the evaluation and application of agile software development methodologies in the European Union based on extensive consultations with industrial and academic partners associated with the NAME network. The report identifies key questions and issues to be addressed and a research methodology for investigating the area.*

1. Introduction. Agile Methodologies (AMs)

An agile software development methodology is an approach to software development that recognises that software applications will change as the business context changes. In many cases these changes will occur during the development of the software and this poses a major challenge to software engineers. In traditional software development the design phase is a significant part of the process but few design methods or notations support the management of rapidly changing requirements.

A number of new approaches to software development have emerged during the last 2-3 years. These, so called *Agile Methodologies*, or *AM*s, are mean to be more able to cope with rapidly changing business requirements without compromising quality or integrity.

The most notable are:

Extreme Programming, also known as XP, [Beck1999]
Dynamic Systems Development Method (DSDM) [Stapleton1997]
SCRUM [Schwaber2002].
Feature Driven Design (FDD) [Coad1999].
Crystal [Cockburn2001].
Agile modelling [Ambler2002].

To a significant extent these all subscribe to what is called the *Agile Manifesto*, a collection of principles that relate to modern pressures that face software development, emphasising, not only the need to be able to adapt to changing needs during development but also the need to engender a culture of personal responsibility, respect and collaboration between all the individuals involved, the so called "human dimension".

The Agile Manifesto.

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

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

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

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

Build projects around motivated individuals.

Give them the environment and support they need, and trust them to get the job done.

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

Working software is the primary measure of progress.

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

Continuous attention to technical excellence and good design enhances agility.

Simplicity -- the art of maximizing the amount of work not done -- is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

This radical departure from the large, design-led, and often bureaucratic, approaches to software engineering raises many questions, questions about its feasibility, effectiveness, success, relevance and cost. These questions will be the main focus for the research proposal for NAME. A rigorous evaluation of AMs and their value in software development will assist Europe's business community by providing a clear framework for planning future software development projects. If being an agile business is vital for success in a dynamic global economy then the systems support for the business must also be of an agile nature. The question is: do agile methodologies deliver this?

2. The research agenda.

The research questions can be partitioned into several broad areas.

Information is required about the efficacy of AMs, how good they are for different types of projects and how senior managers and decision takers in software houses and client business can make decisions about whether to use them, and if so, which AM would be most suitable.

What implications do AMs have for the management of projects, for management structures and the personal development of software engineers?

What technical issues are involved in using AMs and what infrastructure support is appropriate?

In order to answer these questions it will be necessary to develop suitable research methodologies and protocols so that reliable and relevant empirical data can be collected and analysed.

The research results will then require dissemination to the community in an effective manner.

3. The business dimension.

Agile Methodologies have emerged because of a strong pull from business. The problems of traditional methodologies in delivering high quality software, that provides the support that the business currently needs, in a timely manner, have been a major discussion point in software engineering. Many surveys of the industry demonstrate that a large proportion of software delivered is not fit for its intended use, it doesn't support the business need, it's of poor quality etc.

3.1. How do we measure the benefits and the costs of using AMs as opposed to traditional approaches? When faced with a choice of whether to adopt an AM how could a decision be made and on the basis of what data? There is a need to survey the uses of AMs in a variety of business sectors and try to collect data that would provide insight into this question.

Is it possible to make a prediction of the costs of a proposed project carried out with an AM in comparison to the costs of the same project using traditional techniques? If the support of a dynamic business environment is critical how can we measure the cost of NOT being agile?

Does the use of AMs deliver significant benefits in terms of costs, efficiency, reliability etc. In short - are AMs profitable?

3.2. An increasing issue being faced by many software houses is the need to obtain accreditation. This might be generic certification, such as CMM, ISO or it might be industry specific such as Pharma. With stringent regulatory regimes in some industries it is important to understand how AMs fit into this landscape. Evidence needs to be gathered about the regulatory requirements of the industries, of the position of AMs within the various certification processes and whether specific certification of companies using AMs is feasible and desirable.

How should potential clients decide if an AM approach is suitable? What are the attitudes of clients towards AMs? What data would they need in order to make a decision?

What are the legal implications of adopting an AM? How would AMs fare in terms of Product Liability litigation?

3.3. How should an AM be chosen. What are the main differences between the various AMs/XP-based approaches? How should such a methodology be selected and adapted to fit an individual situation? How can we create a framework for selecting and/or adapting an AM? Can we use various process or project metrics, such as size, team localization, etc. if so what are they?

3.4. What are the critical success factors and the major impediments for the successful introduction of AM/XP into an environment / application domain? How should a software company migrate to the use of AM/XP? What is the starting point for adopting and adapting an AM? What experiences and processes from a traditional approach can be translated into AM-based activities? What are the training costs for AMs and how do these compare with the costs for traditional methods?

3.5. How and to what extent can we transfer experience from an AM/XP project to another AM/XP project, taking into account the fact that AM/XP are human intensive?

3.6. Can AMs/XP be extended to contexts where they have not been applied previously? For example, is it possible to apply AMs/XP to develop embedded systems? In areas where safety is a primary concern is the emphasis on testing in XP a potential advantage? Running tests in embedded systems does not fit with current views on XP automated testing, can the approach be adopted to this context.

3.7. Are there ways to perform some sort of Agile Business Process Modelling for businesses? Can the agile principles be applied in other business contexts?

3.8. What is the relationships between AMs/XP and various development approaches, such as:

i. Framework based development
ii. Component based development
iii. (Web) services
iv. Model driven architectures (MDAs)
v. Open source development
vi. Other approaches?

3.9. How and in what domains do AMs/XP increase profits (efficiency, quality, reliability) for the developers, for their customers?

4. Management implications.

Agile methodologies, to be successful demand a new attitude to the management of a project, both for the customers as well as the developers. It is important that research is carried out into the management issues of AMs to understand better the benefits and the pitfalls in their use.

4.1. AMs, being dynamic and adaptive processes, will interact with the customer’s administrative process in a different way to traditional projects. Documentation for business purposes such as project approval, budget and financial planning, legal and contractual agreements will have a different role. AMs tend to emphasize a lightweight approach to the productions of documentation, taking the view that all documentation must have a real purpose. The issue of the requirements document is a key one, if the requirements are changing how can a requirements document be constructed. On the other hand, in a *post-Enron* world there is likely to be more emphasis in commissioning companies on due process and clear budgetary approval of a development project. If the outcome of the project is not described in a document such as a requirements document then will Chief Executives feel vulnerable? Which kind of documentation is essential, necessary, useful and for what goals?

4.2. AMs talk a lot about customers. Who are the customers and where are they? Essentially the issue is about effective and continuous communication between the customer company and the development company. All involved in this must be in touch with both sides. How easy is this, what are the problems that make communication of this sort difficult? Can a remote customer still have excellent communications with developers? What if the representative of the customer changes during the period of the project? Are there issues if the customer is representing another part of the development company, perhaps the project is a large scale distributed one with many teams working on separate parts of an overall project?

4.3. What anxieties do developers/managers experience introducing AMs/XP processes? How can these be ameliorated? Are there specific issues that managers cite as obstacles to adopting an agile approach - are these driven by business, cultural or political pressures? What on-going information, support mechanisms etc. do managers need?

4.4. Does an AM/XP Team have to be in the same place? How can a distributed, agile project be managed? If there are several teams of developers in different places and several different customers how can the communication between them be maintained and supported?

4.5. What are the most effecting ways of managing teams with unequal knowledge/capabilities? Do AMs support personal and technical development better than traditional methods. Do they provide long term gains in developing the potential of staff?

5. Human factors.

Software development is a social activity carried out in teams, teams of developers and, often, teams of customers. The personal interaction between the people involved is often a key factor in success. Research is needed in identifying how the personal, human dimension relates to AMs and their possible success.

5.1. What are the cultural perspectives in software development methodologies? Do AMs work better in some cultural settings than in others? Different countries may have a different approach to software development and this may impact on the adoption of AMs. What are the key cultural factors? Can AMs provide a more uniform platform for development in the EU?

5.2. There are claims that AMs give better motivation, better job satisfaction and a better quality of life? Is there evidence for this? Does this lead to less staff turnover? Keeping key staff is an important issue, is the lack of a rigid hierarchy in an AM development team a positive or a negative factor in terms of career progression?

5.3. What are the training needs of AMs as opposed to traditional techniques? Do AM customers need training as well as developers? How does one cost training needs - whether traditional or AM?

5.4. Are some of the agile practices difficult to adopt? Are agile methodologies only suitable for the best programmers?

6. Technical considerations.

Agile methodologies are still evolving and there are many technical issues that need to be investigated. Every project is different and requires a specific approach. It is unlikely that there is a single detailed methodology that will fit them all. Do AMs provide a framework around which effective and adaptable methodologies will develop? As new computer technologies emerge these will pose new challenges for all types of software development. Can AMs adapt and exploit these new technologies better than traditional methods?

6.1. AMs/XP minimize up front design and design documentation. Many projects experience a need for a more explicit design phase. Can AM/XP be successfully extended to include design and/ or modelling? Is it possible to combine XP practices with UML in a lightweight way? Is there a role for Interaction Design in AMs/XP and how can they be combined? When/where does refactoring become (re)design?

6.2. What effect has the use of AMs/XP during development on the later maintenance of the system? Is software that was developed using AMs/XP easier to understand/maintain? What is the role of refactoring on maintenance? What documentation needs maintenance - should complete documents be updated or should just the changes be recorded? Is documentation really needed for the maintenance of agile systems?

6.3. Code reuse is a major driver of some software technologies but the amount of reuse practised varies greatly. Do AMs result in more or less code reuse? Can the lightweight AM approach to documentation help increase code reuse?

6.4. What is the role of configuration management in AMs/XP? Does it extend beyond simple version control? Who should be in charge of it? Is it possible and necessary to trace requirements in code? Is that part of documentation or configuration management?

7. Infrastructure.

Many tools, support infrastructure and other aids are used in software engineering. The Agile Manifesto emphasizes people rather than bureaucratic processes and tools. But effective tools can provide people with an enhanced working environment. Some of these are valuable but some may actually impede agile processes. Research is needed into what tools and infrastructure are useful for AMs.

7.1. What are the tools that are really needed in AMs/XP? Do we need automatic tools supporting any of the following aspects of AMs/XP? What is the added value of each of these tools with respect to the cost of introducing them?

i. Agile Planning
ii. Testing
iii. Refactoring and Reuse

iv. Eliciting and Managing Requirements
v. Configuration
vi. Continuous Integration
vii. Documentation management
viii. Project/people management
ix. Other support?

7.2. Are there tools facilitating the use of AMs/XP with distributed development teams? If not, what are their requirements/how can we create them?

7.3. Although the agile methodologies share many common philosophical perspectives this does not mean that there are straightforward relationships between them. Is it possible to create an integrated tool that can be used for all AMs? Is this of any benefit if it were possible?

7.4. What are the metrics that give us useful information about the current status and evolution of an agile development project? Are there specific differences between metrics for AMs/XP versus those used for classical software development methods?

7.5. Very little attention is given to the concept of testability in software engineering. Unlike hardware design, the concept of *design for test*, the process of making design decisions to enable more thorough testing of the finished artefact is almost unheard of. Some agile methodologies, for example, XP, put great emphasis on testing, both at the unit and system levels. Which kinds of tools, method and techniques are there:

i. For measuring testability and improving testability through refactoring?
ii. For adding tests to code that is hard to test?
iii. For identifying and dealing with un-testable code?

8. Research methodology.

In order to make progress in trying to throw some light on all these questions it will be important that rigorous techniques for experimental design, data gathering and analysis are employed. With the strong support of our associated industrial and academic partners this will be a key activity.

Our initial methodological approach will pay special attention to the *Preliminary guidelines for Empirical Research in Software Engineering,* [Kitchenham2001] published by the National Research Council for Canada and recommended as a basic benchmark for empirical research for journals and conference referees.

Empirical research proceeds through two principal routes, *qualitative* research and *quantitative* research. In some senses the former is best suited to initial research into complex situations where the issues and concepts are still to be revealed [Ely1991]. It can take the form of a number of approaches, a *positivist* one involving the use of, for example, questionnaires, interviews, focus groups and action research, [Yin1989]. In some cases combining such an approach with an interpretativist one, [Walsham1995], can lead to significant insights, [Trauth2000].

Quantitative approaches have received considerable coverage in the empirical software engineering literature. Although it is difficult to set up robust comparative experiments progress has been made in this direction in evaluating agile methodologies in small scale industrial project settings, for example [Macias2002]. The important issues are to ensure that an independent perspective is taken to the design of the experiments and to the analysis of the data. The definition of suitable metrics and methods to collect them is crucial, [Fenton1997]. Hypotheses need to be stated both clearly and also in a way that can be tested by experiments which, themselves must be properly conducted.

Looking into cultural issues can be assisted by using modern ethnographic research techniques, [Harvey1997] and these will be deployed where possible.

8.1. There are two basic principal observations that will guide fruitful research:

a. Experiments involving AMs/XP must be conducted impartially. AMs are a way to make software they are not a religion.

b. AMs/XP should not be considered in isolation but in the context of other promising research fields, such as Open Software development, Component Based Development, Software Product Lines, as well as business, social and cultural dimensions.

8.2. The research should be multi disciplinary and learn from other fields, e.g., human factors, management, psychology of work, it should avoid re-inventing the wheel

8.3. The research should proceed in close collaboration between Academia and Industry. The associated partners will provide many opportunities for detailed research into the research questions.

8.4. The first phase involves each node disseminating locally, current state of the art knowledge concerning AMs/XP and agile development cultures.

8.5 A methodology should be identified to determine the business value and to make cost-benefit analysis.

8.6. Pilot projects should be designed and run. These will enable us to triangulate our experiences and set up more detailed and focused investigations.

8.7. Experimental protocols should be defined to compare the results from the pilot projects. Within the experimental protocol there should be rules on how to collect the data so that broader conclusions can be made and comparisons between different contexts made.

8.8. It would also be important to obtain evidence in the experimental data of the different viewpoints on the various practices of the key stake holders in an implementation of AMs/XP, for example, Management, Developers, Project Leader, Customer. A grid-based mechanism for presenting this data could be set up.

8.9. A knowledge base should be set up containing the state of the art and of the experimentation to share among researchers and practitioners.

8.10. The results of the experimentation should also be spread to the larger community via (Web) publications, technical reports, and meetings etc.

8.11. Companies already collecting data should be included, including companies not using AMs/XP, as in this way it would be possible to determine the real benefits of AMs/XP in comparison with other approaches.

9. Conclusions.

This programme offers a wide ranging evaluation of what is a rapidly emerging and exciting area of software development, namely the agile methodologies. Already, good progress has been made in identifying the issues that are of primary interest to a wide consortium of industrial and academic partners. We propose an ambitious investigation into the real business drivers for choosing a development methodology together with in-depth research into many issues relating to the agile practices and their use in a variety of business contexts. We will consider how agile approaches are affected by technology, tools, management processes and human behaviour. The ultimate goal is to understand better, how high quality, pertinent and cost effective software can be created in a sustainable way.

References and brief bibliography.

[Ambler2002]. S. Ambler, *Agile modelling*, John Wiley, 2002.
[Astels2001], D. Astels, *Practical Guide to EXtreme Programming*, Prentice Hall, 2002.
[Auer2001], K. Auer & R. Miller, *Extreme Programming Applied*, Addison Wesley, 2001.
[Beck1999]. K. Beck, *Extreme Programming Explained*, Addison-Wesley, 1999.
[Coad1999]. P. Coad, J. de Luca & E. Lefebre, *Java modelling in color*, Prentice Hall, 1999.
[Cockburn2001]. A. Cockburn, *Agile software development*, A. Cockburn & J. Highsmith (eds), Addison Wesley, 2001.
[Ely1991], M. Ely, M. Anzul, T. Friedman, D. Garner & A. Steinmetz, *Doing qualitative research: circles within circles*, Farmer Press, 1991.
[Fenton1997]. N.Fenton & S. Pfleeger, *Software metrics: a rigorous and practical approach*, Brooks-Cole, 1997.
[Fowler2000]. M. Fowler, *Refactoring - Improving the design of existing code*, Addison Wesley, 2000.
[Harvey1997]. L. Harvey & M. Myers, "Scholarship and practice: the contribution of ethnographic research methods to bridging the gap", *Information Technology and People*, 8, pp. 13-27, 1997.
[Jeffries2000], R. E. Jeffries, et al, *Extreme Programming Installed*, Addison Wesley, 2000.
[Jeffries2001]. R. Jeffries, *XP Installed* is available on the website (www.xprogramming.com)
[Kitchenham2001], B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El-Emam & J. Rosenberg, "*Preliminary guidelines for Empirical Research in Software Engineering*", National Research Council for Canada, 2001.
[Lindvall2002], M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams & M. Zelkowitz, "Empirical findings in agile methods", *XP/Agile Universe 2002*, LNCS 2418, pp. 197-207, 2002
[Macias2002], F.Macias, M.Holcombe & M. Gheorghe, "A formal experiment comparing extreme programming with traditional software construction", Tech. Report, University of Sheffield, Department of Computer Science, 2002..
[Marchesi et al 2002], M. Marchesi, G. Succi, D. Wells, L. Williams, *Extreme programming perspectives*, Addison-Wesley, 2002.
[Schwaber2002]. K. Schwaber & M. Beedle, *Agile software development with SCRUM*, Prentice Hall, 2002.
[Stapleton1997]. J. Stapleton, *DSDM: The Dynamic Systems Development Method*, Addison Wesley, 1997.
[Trauth2000], E. Trauth & L. Jessup, "Understanding computer-mediated discussions: positivist and interpretative analyses of group support systems use", *MIS Quarterly*, 24, pp. 34-79, 2000.

[Walsham1995], "The emergence of interpretivism in IS research", *Information Systems Research*, 6, pp. 376-394, 1995,
[Yin1989], R. Yin, *Case study research: design and methods*, Sage Publications, 1989.