Supporting Agility in Software Development Projects – Defining a Project Ontology

Roy MORIEN[1], Pornpit WONGTHONGTHAM[2]
[1]roymorien@hotmail.com
Research Fellow
Digital Ecosystems and Business Intelligence Institute (DEBII)
Curtin Business School
Curtin University of Technology, Perth, Australia
and
Sometime Visiting IS Specialist
Naresuan University, Phitsanulok, Thailand
[1]pornpit.wongthongtham@cbs.curtin.edu.au
Research Fellow
Digital Ecosystems and Business Intelligence Institute (DEBII)
Curtin Business School
Curtin University of Technology, Perth, Australia

Abstract -- **The popularity of agile software development methods, and agile software project management has been accompanied by significant successes in the delivery of software of business value and quality to client organizations, but has also given rise to more pressing difficulties especially in the support of remotely located teams, and distributed or multi-team development activities. The question of whether or not agile methods, which imply small, focused teams, can be successful in 'big' projects also arises.**

This paper discusses the essential elements of agile methods, and agile project management methods, and discusses possible applications of ontology-based project support mechanisms, within the application of the digital ecosystem concept.

Keywords: agile software development, agile project management, project ontology, distributed development teams

1. INTRODUCTION

Agile software development methods, such as Scrum [1, 2], Crystal [3, 4, 5], Feature Driven Development (FDD) [6, 7], Dynamic Systems Development Method (DSDM) [8, 9], Lean Software Development [10, 11, 12], amongst others, have gained a significant popularity in recent years. This popularity can be judged by the number of user groups, books, articles and annual and *ad hoc* conferences that exist, and the popularity of the annual Agile Development Conference, which now seems to be inevitably 'sold out' weeks prior to the conference commencement, despite subsequent efforts to make the attendance larger in the next year.

A significant characteristic of all agile development methods is the emphasis on face-to-face communication between client, developer, and project team leader. Further, transparency enabled by constant communication via daily 'stand up meetings', and 'information radiators' (perhaps progress charts displayed openly and obviously, for example) ensure that all development team members and other interested parties remain fully informed at all times.

This heavy emphasis on transparency, communication and collaboration also brings with it a significant constraint; how to maintain that level of communication in remotely located and distributed development teams.

The Agile Manifesto [13] states the following guiding principles:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

These guidelines, or guiding principles, highlight this conundrum. Each one of these in some way reinforces the notion of direct communication, the need for collaboration, the need for communication and openness. Where 'Individuals and interactions' are emphasised, there must be the ability for individuals to interact. If working software is the hallmark of successful progress, then that must be able to be distributed, and collated, readily to and from remote developers. Equally, if customer collaboration is an essential practice, then supporting that when customer and developer may not even be in the same country or timezone is a necessity. The implications of 'responding to change' are clearly that developed features must be readily available to clients, for their feedback, and the ability for remotely located customers to make their changes known, and for the developers to understand what those changes imply, is also important.

So, how to do that?

2. CHARACTERISTICS OF AGILE AND LEAN METHODS

Before proceeding, it is informative to discuss what is really meant by 'agile software development'. As previously indicated The Agile Manifesto (Agile Manifesto, 2007) states the essential guiding principles. These are elaborated upon, and their implications are focused by a number of authors.

Evans [14] suggests that the essential difference between the traditional approaches to systems development and the agile approaches is the difference between planned iteration and the unplanned rework so common in waterfall-based projects. With the traditional approach, adherence to the prescribed software process is considered the major determinant of success. With the agile approach, adaptation toward achieving the end-goal – working software – is the major factor in success.

This table, taken from Evans (op.cit) summarizes some *"contrasts between these remarkably different approaches.*"

	Waterfall	**Agile**
Guiding metaphor	Manufacturing /Engineering	Organic /Emergent
Focus	Documentation, Schedule	People, Working Code
Dynamic structure	Cause and Effect, Preventive Approach	Chaordic (Ordered Chaos), Adaptive Approach

Welcome Change

If there is one guiding principle of agile development, it is "Welcome Change'. This implies the acceptance of the fact that requirements in detail cannot properly, comprehensively or accurately be defined at the beginning of the project (the "Big Bang" approach, or the Big Design Up Front (BDUF) approach [15, 16]), and are almost certainly subject to change in extended period projects. Highsmith [17], in Orr [18]) states that "*By the time a three-year project delivers its first working versions, many of the users have forgotten what they agreed on in year one or have moved on so that the people who have to work with the system have little or no idea what it was developed for" He further suggests that "if a project takes three years to implement, you can be sure that the requirements will be at least two years out of date by the time it comes into existence.*

Specific definitions of agile development have been attempted by Mahanti [19] as *"a departure from plan-driven traditional approaches, where the focus is on generating early releases of working software using collaborative techniques, code refactoring, and on-site customer involvement".* And Melnik and Maurer [20] as *"human centric bodies of practices and guidelines for building usable software in unpredictable, highly-volatile environments".* Software development projects are considered to be an unpredictable and highly volatile environment.

For the purpose of this discussion, a definition of Agile and Lean development is *"A software development method is said to be an agile software development method when a method is people focused, communications-oriented, flexible (ready to adapt to expected or unexpected change at any time), speedy (encourages rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development)"* [21]

So, for a software development activity to be agile, it should encompass practices that can be variously described as:

- People Focused: (1) Collaborative: collaboration between developers and clients is continuous and continual.. (2) Self-Organising and Self-Managing Teams: Significant responsibility is handed to the team members, rather than the Project Manager, to decide on the work to be done in the next iteration.
- Empirical and Adaptive: Project management practices that have been published to support 'agile development' practices are described as 'empirical', 'adaptive', 'evolutionary' or 'experiential' rather than 'prescriptive', or 'pre-planned'.
- Iterative: Development is achieved through a series of short iterations each of which produces a useable enhancement to the system.
- Incremental: Development is achieved through a series of delivered increments to the system, each of which produces a fully developed, fully tested and certified extra feature or component of the system.
- Evolutionary: the system grows in size, the requirements *in detail* are continuously discovered, and are continually emergent during the development period.
- Just-in-Time Requirements Elicitation: Requirements are stated in detail 'just in time' to develop them, in the iteration in which those requirements will be implemented.

Knowledge-Based: Development activity is decided upon by the knowledgeable, self-managing members of the team, with continual knowledge sharing about the product, the technology and the progress of the project.

3. DISTRIBUTED AND REMOTELY LOCATED TEAMS

A software development project comprises a number of different players, with different but converging interests. The client (the ultimate recipient of the project activities' outcomes, and the financier of the project) expects to receive a system that will provide business value to them, and to the organisation within which the system will ultimately reside.

One difficulty faced by the client is to be able to authoritatively, comprehensively, unambiguously and correctly define the requirements of the system, and possibly more importantly, the measures of business value that those requirements will offer. Ranking the various, and probably multitudinous set of requirements in some order of importance, business value and development priority is the client's difficult task. This is compounded by the obvious necessity to communicate those requirements to the other major player in the project activity, and that is the development team. The development team is presumed to have technical skill and competence, and to have an appropriate portfolio of development tools available to it to achieve the development task, but cannot usually be expected to have the domain knowledge of the client. Again, the ability to record and communicate those requirements is of extreme importance to the development team.

A third, and also significant 'player' in the project activity is the project administrator, project leader, project manager; call it what you will. There is some hesitation in using the term 'project manager' given the attitude against such a role by the adherents of the agile development approach, who eschew the whole concept of a project manager, with a 'command and control' role. The agilists prefer the concept of a self-directed team, with what is tantamount to a project protector, whose role is to run interference on the project team, clearing the way for them to vigorously pursue their short-term goals defined for an iteration (of possibly even as short a time as a week). The 'project protector' (my term) assists in clearing the way, removing obstacles, garnering required resources, and being the outward turned face of the project team, to which all requests from 3rd parties are addressed.

There is therefore a significant amount of 'knowledge' required, and extant, to ensure the productive and useful efforts of the various players in the project activity.

The problem of gathering project information, making it available to all players in the project is difficult in any case, but when team interaction, collaboration and communication is heavily emphasised, and must be supported for remotely located and distributed teams, then the difficulty is multiplied significantly.

If this substantial body of knowledge, much of which must remain available long after the development phase of the project has been completed, is to be recorded and made available, then one highly successful approach is to create a project ontology.

4. AGILE METHODOLOGIES – COMMUNICATION & COLLABORATION

A number of agile development methods and approaches have been published. These include:

- EVO www.xs4all.nl/~nrm/EvoPrinc/, [22].
- Spiral Model [23,24].
- Extreme Programming (www.xprogramming.com/xpmag/whatisxp.htm), [25]
- Scrum (www.controlchaos.com), [2]
- Crystal (alistair.cockburn.us/index.php/ Crystal_methodologies_main_foyer) [3, 4, 5]
- Feature Driven Development (FDD) (www.featuredrivendevelopment.com/) [6, 7]
- Dynamic Systems Development Method (DSDM) (www.dsdm.org/) [8, 9]
- Lean Software Development (www.poppendieck.com/), [10, 11, 12]
- Agile Unified Process (www.ambysoft.com/ unifiedprocess/agileUP.html), [26]
- Agile Data Modelling Method [27]
- Rational Unified Process (RUP) [28]

A recent industry survey indicated that Scrum was the predominant agile method in use, being applied in 37% of organisations that were using an agile approach in their development activity [29]. This survey had over 1680 respondents from 71 different countries.

The Scrum agile method emphasises project transparency, continual communication and collaboration between project partners. The Scrum method is illustrated in this diagram (from www.controlchaos.com)

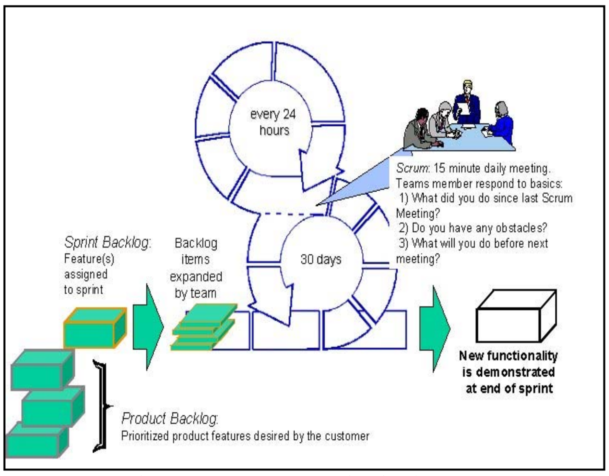
every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team
30 days
New functionality is demonstrated at end of sprint
Product Backlog: Prioritized product features desired by the customer

Figure 1: Scrum Method Diagram

The Scrum method depends on a Product Backlog that is a prioritized list of all the requirements as they are known and understood at any given time. There is no expectation of a complete and comprehensive list. There is an expectation of change, however. The Scrum project starts with analyse meetings between all players to elaborate the Product Backlog as much as is possible at that early time in the project. The project proceeds in a series of short iterations, called Sprints in this method. Sprints can be as short as a week, and as long as a month. Shorter sprints are suggested to keep the project highly visible to all players, and then maintain a rapid and constant output of useable components. (The diagram suggests a 30 day sprint, but there is a preference for a weekly sprint). At the start of each sprint a full team meeting takes place, and team members volunteer for tasks as stated on the Project Backlog – the highest priority tasks being taken first.

To maintain project impetus, and high transparency and visibility, there is a formal Daily Scrum, or Daily Standup Meeting, where all members of the development team share their day's experience. This is not a 'report to the Project Manager' meeting, but a collaborative, knowledge sharing, help-seeking (where necessary) meeting. It is intended to not last more than 15-20 minutes, but is essential to keep all developers 'in the loop' and fully aware of the project activity.

At the end of each sprint, the completed outcomes of the sprint are demonstrated to the client, and if possible released into the production environment.

Another important artefact in the Scrum method is the Burndown Chart.

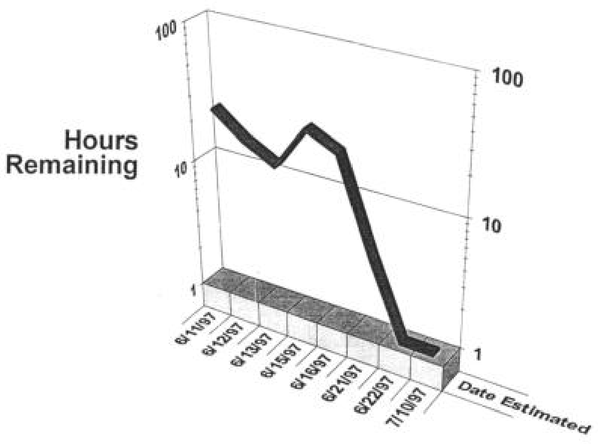

Figure 2: Scrum Burndown Chart

This chart essentially shows the amount of work that is estimated to remain to be done. It is almost an inverse 'progress chart', not showing the work, time and effort completed (that is now irrevocable history), but the amount work, time and effort remaining. Given an agreed upon project completion date, then this chart clearly shows the likelihood of completing all of the requirements by that time. There can also be a sprint burndown chart, showing the likelihood of completing all the selected tasks in that sprint.

Overall, agile development methods can be characterised by the slogan 'Welcome Change'. Other 'slogans' that are applicable to agile methods include 'Fail early', 'Deliver early and often'. These are all indolent of the idea that agile methods allow frequent face-to-face communication between the project parties, and continual communication and update of the project status, including new and changed requirements. Problems are surfaced almost immediately they are encountered, and cannot be dismissed or deferred, often increasing in size, intensity and risk.

Given the high importance of continual communication between project parties, it is considered that undertaking projects in an agile manner, with remotely located and distributed teams, sometimes distributed across almost all time zones, therefore requiring almost 24/7 access to current project information, is an especial problem.

It is suggested here that the development of a project ontology, that allows seamless updating, promulgation of current information, and ready access (to and by authorised personnel), is an ideal approach to project administration.

5. SUPPORTING THE PROJECT - A PROJECT ONTOLOGY

In recent times there has been considerable growth of interest in ontologies as knowledge structures. One especially germane example of an ontology, for the present purposes, is to be found at www.seontology.org , which is purported to be the world's first Software Engineering Ontology. This Software Engineering Ontology defines the shareable software engineering knowledge, which includes specific project information [30, 31]. However, this ontology, like most other ontologies, is fundamentally static, or has a passive structure. Like many knowledge bases (such as the Internet itself) searching that knowledge base is effective and useful provided, first, that the searcher knows exactly what they are searching for, and second, that the knowledge is coded in such a way that search results are relevant and address the real question. For example, the oft-quoted search on the keyword of 'Jaguar' may return many hits on natural history information, endangered species and habitat destruction, when it is information about a famous marque of motor vehicle is intended.

Like any other repository of information, an ontology must be maintained for currency, relevance and completeness. In a project team, this will need to be done by all or any of the players who will be authorised to do so. In a distributed and remotely located team situation this is a problem exacerbated by distance geographically and in time.

For these various reasons, the project players will need active support.

The question then is, How can we provide this active support to the project players; the end users of the project support system? How do we make it easier for them to find the information that they need, and how to provide them with meaningful information.

Further, given that almost without exception all participants in an agile project activity will need to be able to update, modify, change, add to, remove from the information in the ontology, how do we rank requirement requests and requests for changes to previously accepted requirements. Ranking here implies the level of authority or influence that any given player has. This is a manifold problem, not just restricted to some access verification 'firewall'. The right of any given player to propose a change to requirements, the expectation that such a request is authoritative, the acceptance of the priority of the change; these are all subjective factors that need to be addressed within the ontology, because access to management decision making may not be an option given the remoteness of the suggester.

These are matters that are given elevated status in an agile development project. Given also the 'need for speed' that is inherent in the agile development decision process, together with the fact that the developers are fundamentally self-organised, and need constant access to up-to-date information on progress, problems, and prioritised requirements.

6. PROPOSED ONTOLOGY PROJECT

It is intended to undertake a project to develop such an ontology. The intention is to develop a social network based recommender approach which will provide active support and recommendations for remotely located project players and distributed team members. The intention is that they will be able to effectively and efficiently access and share project knowledge in an agile software development and project situation. This approach integrates the software engineering ontology and recommender approach through the utilisation of social network agents. This is considered to be a new and innovative approach for remotely located and distributed software development teams. Of particular usefulness is that the software engineering ontology enables an active ecology of social network agents to convey, utilise and act on project information at least semi-autonomously, according to explicit project domain knowledge. An innovative characteristic is that recommendation techniques are used to recommend useful project information and tentative solution(s) for project issues that are raised by team members.

A key feature is that it is a social network-based system integrated with another key emerging technology - semantic web. It is an innovative social network based system supporting remote collaborative applications. The problem areas to be explored are as follows. First is the development of a new social network based recommender system architecture that facilitates meaningful communication, discussion, negotiation and information exchange through collaborative agents interacting with the ontology. A major aim of our approach is to enable the integration process from syntactic interoperability to semantic interoperability. Another aim is to develop an open knowledge platform for sharing project artefacts, expert opinions, progress information, project document and standards, etc. An important requirement that has guided the design of an open knowledge platform is the support of software development teams which are geographically distributed. An open knowledge platform enables provision of expertise and coordination of project information within a team; especially if it allows new members or new teams to catch up with the current project. The open knowledge platform provides the best opportunities for sharing and retrieving knowledge and software project information but also entails some significant techniques in member profile evaluation. A first prototype of the social network based recommender system will be developed using JADE. JADE (Java Agent Development framework) is a software framework that simplifies the implementation of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems [32, 33]. Given the distributed nature of the JADE based multi-agent systems, teams can be distributed connected usually through the Internet and situated in different parts of the globe. Each system platform can be distributed on different computation nodes and it is connected to a web server where social network agents reside, allowing direct interactions with the members. Field-testing, evaluation and benchmarking will be carried out to validate the practical value and usability of the open knowledge platform. Evaluation processes include the industry partners making use of the prototype, use of surveys and focus groups in order to find out the issues with the prototype, and forming case studies in order to probe in-depth issues.

7. CONCLUSION

Agile, iterative development, which can be described as a knowledge-based, self organising software development approach, and software project management approach, requires extensive face-to-face communication, personal collaboration and communication, and frequent sharing of knowledge and information, between all players in a project (client, developer, project 'owner' etc.). A further major feature of agile development is the acceptance of continual change requests, and the management and prioritisation of those change requests, through a continually updated and administered Project (or Product) Backlog.

These characteristics of such an approach are significantly problematic when considering remotely located and distributed development teams. This is considered to be considerably more so than in the tradition, rigorously pre-planned development approaches, where all requirements and planned activities are stated 'up front', and ostensibly will not change (except through a further rigorous 'change control' system).

A project ontology, encompassing all of the 'knowledge' of the project, and providing active support and recommendations to project players, is envisaged, and will be the subject of a research and development project that will be undertaken at The Digital Ecosystems and Business intelligence Institute (**DEBII**) at Curtin University of Technology.

DEBII is a University *Tier one Research Centre of Excellence* incorporating Research Centres of Frontier Technologies for Extended Enterprises (CEEBI) and AoRE (Area of Research Excellence) (http://www.debii.curtin.edu.au/).

8. REFERENCES

[1] Advanced Development Methods Inc, Home Page, http://www.controlchaos.com/, accessed October 2007

[2] Schwaber, Ken & Mike Beedle, *Agile Software Development with Scrum*, Prentice Hall, 2001, ISBN 0130676349

[3] Cockburn, Alistair, Crystal Clear: *A Human-Powered Methodology for Small Teams* (The Agile Software Development Series), Addison-Wesley, 2004, ISBN 0201699478

[4] Cockburn, Alistair, *Agile Software Development: The Cooperative Game*, Addison-Wesley, 2006, 2nd edition, ISBN 0321482751

[5] Crystal Methods Main Foyer,

alistair.cockburn.us/ index.php/

Crystal_methodologies_main_foyer, Accessed October 2007

[6] Feature Driven Development Portal, www.featuredrivendevelopment.com/, accessed October 2007

[7] Palmer, Stephen R. & John M. Felsing (2002*), A Practical Guide to Feature-Driven Development* (The Coad Series), Prentice Hall, , 2002, ISBN 0130676152

[8] DSDM Consortium, www.dsdm.org/, accessed October 2007

[9] Stapleton, Jennifer (2003), *DSDM: Business Focused Development*, DSDM Consortium, Second Edition, 2003

[10] Poppendiek.LLC Home Page, www.poppendieck.com/, accessed October 2007

[11] Poppendieck, Mary & Tom Poppendieck, *Lean Software Development: An Agile Toolkit*, Addison-Wesley, 2003, ISBN 0321150783

[12] Poppendieck, Mary & Tom Poppendieck, *Implementing Lean Software Development: From Concept to Cash,* Addison-Wesley, 2006, ISBN 0321437381

[13] Agile Manifesto, http://agilemanifesto.org/, Accessed October, 2007

[14] Evans, Ian (2006), "Agile Delivery at British Telecom, Methods & Tools", Summer 2006:20, www.methodsandtools.com/ mt/download.php?summer06

[15] Ambler, Scott (2003a), *Agile Database Techniques: Effective Strategies for the Agile Software Developer*, Wiley Application Development Series

[16] Ambler, Scott (2003b), "Something's Gotta Give", Dr Dobbs Architecture & Design, March 1st, 2003, http://www.ddj.com/architect/184414962

[17] Jim Highsmith 'bio' @ http://www.adaptivesd.com/about.html, accessed October 2007

[18] Orr, Ken (2002), "Agile Requirements", Agile Project Management Advisory Service, Executive Report, Vol. 3, No. 12, Cutter Corporation

[19] Mahanti, A. (2006). "Challenges in enterprise adoption of agile methods - A survey". Journal of Computing and Information Technology **14**(3): 197-206.

[20] Melnik, G. and F. Maurer (2005). "Agile methods: A cross-program investigation of student's perceptions of agile methods". Proceedings of the 27th international conference on Software engineering ICSE'05, ACM Press, IEEE Computer Society.

[21] Qumer A., B. Henderson-Sellers, (2007), "An Evaluation of the Degree of Agility in Six Agile Methods and its Applicability for Method Engineering", Information and Software Technology, 2007

[22] Gilb, T. (1988), *Principles Of Software Engineering Management*, Addison-Wesley

[23] Boehm, Barry (1986), "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes, August, 1986.

[24] Boehm, Barry, "A Spiral Model of Software Development and Enhancement", IEEE Computer, vol.21, #5, May 1988, pp 61-72.

[25] Beck, Kent (2004), *Extreme Programming Explained: Embrace Change, 2nd Edition*, Addison-Wesley, 2nd edition, ISBN 0321278658)

[26] Alhir, Sinan Si, "The Agile Unified Process (AUP)", home.comcast.net/~salhir/ TheAgileUnifiedProcess.PDF, Accessed March 1st, 2007

[27] Agile Alliance (2007), www.agilealliance.org/show/1641

[28] www-306.ibm.com/ software/rational/), ootips.org/rup.html, accessed October 2006.

[29] VersionOne and APLN, 2nd Annual Survey "The State of Agile Development", conducted June-July, 2007

[30] Wongthongtham, P., A methodology for multi-site distributed software development, PhD Thesis, Curtin University of Technology, 2006)

[31]Wongthongtham, P, Chang, E, Dillon, T & Sommerville, I 2006, 'Ontology-based multi-site software development methodology and tools', Journal of Systems Architecture, vol. 52, no. 11, pp. 640-53.

[32] Bellifemine, F, Poggi, A & Rimassa, G 2001a, 'Developing multi-agent systems with a FIPA-compliant agent framework', Software Practice and Experience, vol. 31, pp. 103-28.

[33] Bellifemine, F, Poggi, A & Rimassa, G 2001b, 'JADE: a FIPA2000 compliant agent development environment', The fifth International Conference on Autonomous Agents, ACM Press, New York, USA, Montreal, Quebec, Canada, pp. 216-7.