Edith Cowan University
Research Online

Australian Information Warfare and Security Conference

Security Research Institute Conferences

2010

Success of Agile Environment in Complex Projects

Abbass Ghanbary

Julian Day
Consensus

Originally published in the Proceedings of the 11th Australian Information Warfare and Security Conference, Edith Cowan University, Perth Western Australia,30th November - 2nd December 2010
This Conference Proceeding is posted at Research Online.
http://ro.ecu.edu.au/isw/31

Success of Agile Environment in Complex Projects

Abbass Ghanbary (PhD), Julian Day
Consensus
abbass.ghanbary@gmail.com
Julian.day@consensus.com.au

Abstract

This paper discusses the impact of agile methodology in complex and modular interrelated projects based on the authors' practical experience and observations. With the advancement of Web technologies and complex computer systems, business applications are able to transcend boundaries in order to fully meet business requirements and comply with the legislation, policies and procedures. The success of software development as well as software deployment of these complex applications is dependent upon the employed methodology and project management. This is so because employed methodology plays an important position in capturing and modeling of business requirements and project management helps to ensure delivery. Agile methods are rapidly becoming popular in the software development industry. This paper examines this crucial role of agile methodology in a software development and deployment environment.

Keywords

Agile, Methodology, Business Requirements, Roles, Risk, Validation, Verification and Technology, Project Management.

INTRODUCTION

Business applications are systems that are today typically accessed via Internet technologies. As per Ghanbary (2006) the technology has increased the connectivity all around the globe and organisations are globalising faster than ever before. This advancement has raised the expectations of people in their work conditions and living standards.

These business applications are designed, developed and deployed based on the business logics that are both compliance related by external entities such as Government and internal entities manufactured by the business.

Business plays an important and ongoing role in system development projects through the provision of domain validation and feedback. Regular involvement of stakeholders and domain experts helps to ensure the right system is built for the need of business. Also, business develops better understanding of their requirements as the system evolves. Changes to the requirements will change the scope, time and budget of the projects. The employed methodology, including project management techniques, in software projects must embrace all these issues and concerns from a development and deployment prospective. From a commercial perspective, the employed methodology needs to be flexible to comply and access each change request and ensure that these requirements are scoped, accessed and managed in co-operative fashion.

This paper evaluates the capability of agile development issues in order to evaluate the flexibility of this methodology for complex and collaborative projects.

DEFINITION OF AGILE

According to Unhelkar (2010), agile methods are developer-centric, starting with the developer writing part of the code and demonstrating it to the user in an effort to engage them in a conversation. This interaction through the code further clarifies and illustrates the requirements that are built-in in the code.

Based on Badr's (2006) agile approach to software development is the process of rapidly creating the end application by introducing efficiency measures. Beck (1999), further describes Extreme Programming (XP), as part of agile approach that has been adopted for various software development approaches. Agile as an iterative and incremental approach to software development is performed in a highly collaborative manner to produce high quality software that meets the changing needs of its stakeholders (Ambler, 2006).

Agile approach to development of software differs from the more traditional approaches in a number of ways. The rapid development lifecycle can be said to encompass an 'Agile' approach. Thus, while the agile approach is not a lifecycle, it provides a suite of key principles on which software development and maintenance can be based as stated by (Abrahamsson, 2002) and (Ambler, 2006). The following principles outline the basis of agile software development:

- The highest priority is to satisfy the client through delivery of valuable software.
- Agile development welcomes changes in support of the client's competitive advantage.
- Deliver working software frequently.
- Business people and the development team work together on a daily basis.
- Build projects around motivated people, give them the environment and support they need and trust them to deliver.
- The most efficient 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 within a project should be able to maintain a constant pace indefinitely.
- The best architectures, requirements and designs emerge from self-organising teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Although agile methods differ in their practices, they share a number of common characteristics, including iterative development, a focus on interaction, communication, and the reduction of resource-intensive intermediate artifacts (Cohen et al., 2004).

REQUIREMENT DEFINITIONS AND AGILE

Agile methodology emphasises on production of working source code by reducing the appropriate documentation of business requirements. The concentration on source code production and less documentation in agile process means that project knowledge is stored mentally by all team members. This has been exercised many years ago (in the early stages of computer engineering), however, the question is the practicality of agile method in today's environment with complex projects including complicated business requirements that must be compliant with internal factors such as policies and procedures while is fully compatible with external factors such as Government legislations.

The dynamic change requirements (faced by any projects) can be very problematic in agile methodology due to the lack of documentation. According to Cohen (2004), agile methodology is not very suitable for systems that have high criticality, reliability and safety requirements. The organisation's culture, people and communication have major impact on the success of the agile projects while project size is the most important factor, as size grows, face-to-face communication becomes more difficult. Therefore, agile methods are more suitable for projects with small teams.

Extreme Programming (XP) as part of agile methodology has evolved from the problems caused by the long development cycles of traditional development models. Kent Beck wrote the first book about XP in 1999 (Beck, 1999) and expanded it with a second edition in 2005. XP is a deliberate and disciplined approach to software development; it is designed to empower the developers to confidently respond to changing customer requirements, even late in the project life-cycle. This methodology emphasises team work and implements a simple and effective way to enable groupware style development (Wells, 1999). XP improves the software project in four essential ways; communication (between programmers and their customers), simplicity (by keeping a simple and clean design), feedback (by testing the software starting from day one), and courage (XP programmers are able to courageously respond to changing requirements and technology) (Wells, 2000). Beck and Fowler (2001) stated that XP is a new way of building the right software, and building it fast, to a high quality and explain how it has been tried on real projects that proved it work with a small development team.

Figure 1 shows how the rules and practices must support each other and work together to form a development methodology (Maharmeh & Unhelkar, 2008).

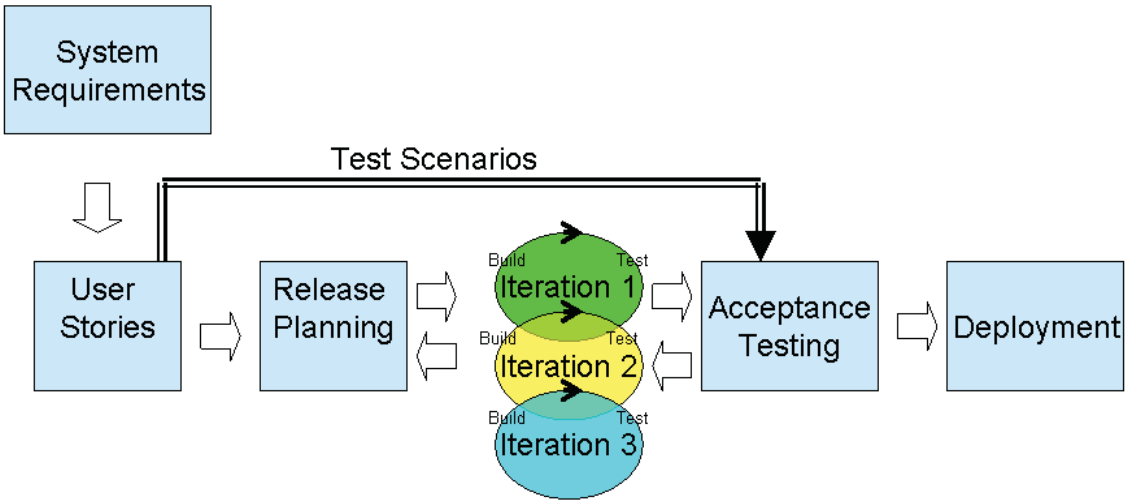

Figure 1: Example of eXtreme Programming Process (Maharmeh & Unhelkar, 2008)

According to Beck (1999), the XP process life-cycle consists of five phases:

Exploration phase: In this phase, the customer writes out the story cards that they wish to be included in the first release, and the project team familiarise themselves with the tools, technology and practices they will be using in the project.

Planning phase: In this phase, the project team sets the priority order for the stories and an agreement of the contents of the first small release. They also estimate how much effort each story requires and agrees on a schedule.

Iterations to Release phase: This phase includes several iterations of the system before the first release. The schedule that has been set in the planning phase is broken down to a number of iterations, the first iteration creates a system with the architecture of the whole system, the functional tests that have been created by the customer are run at the end of every iteration and at the end of the last iteration the system is ready for production.

Production phase: In this phase extra testing and checking for the system performance before it can be released to the customer.

Maintenance phase: This phase requires production support efforts for customer support tasks. Therefore, the development activities may decelerate after the system is in production.

Death phase: This phase is reached when the customer no longer has any stories to be implemented. In this phase system documentation is finally written as no more changes to the architecture, design or code are made.

XP is aimed for small and medium sized teams for better communication of the business requirements. Any resistance against XP practices and principles on behalf of project members, management or customer may be enough to fail the process. The physical environment is also important in XP as communication and coordination between project members should be enabled at all times.

PROJECT COMPLEXITY AND AGILE

This paper evaluates the capability of agile development issues in order to weigh up the flexibility of this methodology for project risks that could affect the development project. It also evaluates the most appropriate project management expertise required. Planning for the system architecture including product functionality and requirement delivery (where the problem is divided into a set of separate however interrelated tasks) must be planned based on current requirements and assessed risks.

Agile methodology as an iterative, incremental approach might assume that all requirements are not known in advance; therefore the system architecture may not fully support the functionality of the developed product. Agile project knowledge is stored mentally by team members. This is so because, in agile approach, there is no or little proper modeling of the software system beforehand.

The management team must determine that the product is deliverable based on the user requirements and the complexity of the interrelations of the internal modules of the system is covered. This architectural design of internal modular complexity must be extended if the system is collaborating with other systems that are functioning independently internal or external to the institution.

AGILE RISK MANAGEMENT

Based on the previous section, the success of complex projects in an agile environment can be fully dependent upon validation and the verification of the system.

Validation is ensuring the right system is produced. Effective validation is a crucial aspect of ensuring a successful software development project. Verification is performed as part of development iteration and includes techniques such as functional testing and code review.

Validation and verifications are two intrinsic aspects of software development such that:

- Validation involves the right system is built, while verification involves ensuring the system is built right.

- Validation is an essential activity that should be undertaken throughout the entire development effort. If validation is not effectively undertaken it can result in a system that does not address the concerns of stakeholders or the needs of the organisation.

- Validation is typically documented in functional test cases that describe the correct behaviour of the system. Verification is formally undertaken as part of iteration and can include unit testing, functional testing and regression testing.

Risk management is a critically important area to identify threats, estimate risk, and manage risk review. The most important stage of a risk analysis is to identify threats that are facing the undertaken project. These threats may be Human (team members change and illness or death), Operational (distribution of resources), Reputational (organisation reputation), Procedural (internal systems and control), Project (cost and time), Financial (internal and external factors), Technical (advances in technology), and Political (compliance change)

PEOPLE ROLES IN AGILE

Managing people in Agile environment requires special mechanism since the project manager needs to ensure that all the people involved in the projects have the same level of understanding hence the development is taking place with minimum amount of the documentation. Types of people commonly involved in software systems are as follows:

Account Manager: Account Manager is responsible for managing the client relationship and addressing any concerns the client may have.

Business Analyst: Business Analyst is responsible for capturing the business requirements and documents them in a way that is understandable by all parties in the project.

Application Architect: An Application Architect is responsible for designing components within the broader system architecture.

Domain Expert: A Domain Expert is someone who has extensive knowledge of the business domain related to the system. Domain experts provide system validation and are an essential component of success in software development.

Project Manager: The Project Manager within a software development project is responsible for delivery planning. The Project Manager is the person to consult for information on how the project is progressing against schedule and budget. Project Manager is also responsible for managing risks and issues within the development effort.

Quality Coordinator: A Quality Coordinator is responsible for planning and coordinating the ongoing assessment of a system including verification and testing.

Software Engineer: A Software Engineer is someone who is trained and experienced in the field of software development. A Software Engineer should have strong understanding of software engineering first principles

and be experienced in multiple programming language while having a great understanding of business requirements.

Solution Architect: A Solution Architect is usually involved in medium to large scale projects and is responsible for designing the overall architecture of the system. The architecture of a system is a fundamental factor in determining the system's quality properties such as extensibility, flexibility, etc.

System Testers: A System Tester tests the systems to ensure that it is suitable for use before the delivery.
Based on Beck (1999) the team shall be between 6-20 people in agile environment to ensure that communication within team members are managed and controlled.

RELEASE MANAGEMENT IN AGILE METHODS

Deployment as the last phase of most methodologies is the most important factor for the success of the project. The project manager needs to ensure that tasks undertaken by different developers has been validated and verified. This is the most complex issue in agile development; yet again due to the lack of documentation and proper modeling of the system before development.

Work units could be performed iteratively meaning that the same activity may be repeated in one or more iterations during the actual deployment. These work units are the components used to model the operations performed by the producers to develop work products, such as tasks, technique and activity in various stages.

Based on the following identified characteristics by Miller (2001) on agile methodologies, the author will evaluate the risk of each characteristic:

- **Incremental:** small software releases, with rapid development cycles.
    - **Associated Risk:** The impact of each software release on the other part of the system hence there is no proper models to identify the impacts before actual release.
- **Cooperative:** the customer and developers work constantly together with close communication.
    - **Associated Risk:** To ensure all the communications and captured requirements are actually implemented in the system as the projects grows bigger, requirements change and business rules are changed.
- **Straightforward:** the method should be easy to learn and to modify, as well as document.
    - **Associated Risk:** Very useful for high level understanding of business requirements but detailed analyses are mainly discussed during the development. This will reduce the chance for project managers to ensure all detailed requirements are actually implemented.
- **Adaptive:** the method should allow for making changes up to the last moment.
    - **Associated Risk:** This issue will have impact on other internal module of the system (in complex interrelated projects) hence there is yet again no proper modeling to evaluate the change of the business requirements and business rules on the other parts of the system.
- **Development:** Development cycles are short and iterative which enable fast verifications and corrections.
    - **Associated Risk:** This mainly involved risk to ensure no requirements are forgotten (missed) as a result of this fast development.
- **Iteration:** Iteration cycle's time-bound is from one to six weeks.
    - **Associated Risk:** Convergent approach that minimizes the risks.

PROJECT MANAGEMENT IN AGILE DEVELOPMENT

The key overriding feature of agile development is the project management capability of the team members and the lead developers. Managing many iterative phases is a challenge to most individuals, let alone a team of individuals whose work has to be coordinated and merged in some instances into the final resultant system. Engagement and involvement of all team members and the users is crucial to ensuring an optimum outcome.

CONCLUSION & FUTURE DIRECTION

This paper outlined the impact of agile methodology from different perspectives such as capturing the business requirements; its effect on complex projects, risk analyses, people's roles, and management of the release. This paper facilitates the practice of large, medium and small software development projects and their relevant activities to be more successful, resulting in high quality software applications that are also produced within time and budget while all the business requirements are captured and implemented in agile environment.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J., 2002, Agile software development methods. Review and Analysis. Espoo. VTT Publications 478.

Ambler, S. W., 2006, Agile Software Development. Agile Modeling, 2006. Viewed 10 June 2006.

Badr, I., 2006, Rapid Development through Agile Modelling. Telelogic AB, 2006. Viewed 04 June 2006. < http://www.telelogic.com >

Beck, K., Extreme programming explained: Embrace change. Addison-Wesley, 1999. ISBN 0-201-61641-6.

Beck, K. & Fowler, M., Planning Extreme Programming (XP Series). Addison Wesley, 2001.

Cohen, D., Lindvall, M., & Costa, P., An introduction to agile methods. In Advances in Computers (pp. 1-66). New York: Elsevier Science, 2004.

Ghanbary, A. (2006)." Collaborative Business Process Engineering across Multiple Organisations" proceedings of ACIS 2006. Adelaide. Australia

Charette, R., Why Software Fails. IEEE Spectrum, Sep 2005.

Maharmeh, M. & Unhelkar, B. (2008). Investigation into the Creation and Application of a Composite Application Software Development Process Framework. Proceedings of ITNG Conference. ITNG 2008. Las Vegas, NV. 7-9 April 2008 Page(s):1286-1286

Miller, G. G., The Characteristics of Agile Software Processes. The 39th International Conference of Object-Oriented Languages and Systems, Santa Barbara, CA, 2001

Unhelkar, B., (2010), Agile in Practice- A Composite Approach,. Cutter Consortium., Vol 11, No 1. © Cutter Consortium Executive Report.

Wells, J. D., (1999), Extreme Programming: What is Extreme Programming,. Viewed 14 June 2006, < http://www.extremeprogramming.org/What.html >