Issues Identified in the Software Process due to Barriers found during Eliciting Requirements on Agile Software Projects: Insights from India

N.Ganesh
Research Scholar, Faculty of Computer Science and Engineering, Anna University, Coimbatore, India

S.Thangasamy
Dean, Research and Development, Kumaraguru College of Technology, Coimbatore, India

ABSTRACT
Nowadays, many of the Agile based project development teams are distributed geographically across the globe. The teams are divided into onshore and offshore that work together to achieve on a common target. The teams in turn work in small iterations to minimize the effect of change in software requirements and at the same time developing regular communication between them. However different factors such as physical distance and lack of proper communication between the onshore and offshore teams become a hurdle between them leading to misunderstandings about software requirements. Though there are many advanced way of communication like video conferencing, voice chatting is available, it certainly has several disadvantages in getting the task completed. This paper gives an insight about these challenges faced in many of the software industries and it would allow different stakeholders within agile based onshore /offshore setting to better understand these challenges in eliciting their requirements.

General Terms
This article can be classified under the problems faced and the solutions achieved in the software life cycle management.

Keywords
Agile, Requirements elicitation, knowledge sharing, onshore / offshore

1. INTRODUCTION
The major challenge found for better understanding of Software requirements is due to lack of communication between the onshore and offshore site distributed teams. Shorter iterations at the offshore site require more communication with the onshore site. The language problem seems to exist only when both the onshore and offshore site teams who are non-English speakers communicate in English. Regular long distance meetings would help in better understanding of software requirements. Previous domain and product knowledge is helpful in better understanding of software requirements.

In developing a software, the requirement gathering ways are broadly classified into two groups as internal (team members) and external (customers, partners) to the software project. In such a type of projects, Agile is the best answer where you work in small iterations and deliver working software at the end of iteration. The diverse collection of customers along with tough competition with its competitors puts a lot of pressure on the software requirements of a particular software product. Agile methodologies are very helpful in this regard as it welcomes changes, especially late changes [1].

The core concept in agile development is that the team can be increasingly effective when the time and cost of moving information and decisions between people is reduced [1]. For this purpose the development team should be co-located thereby increasing communication and there should be a close collaboration with customer and other stakeholders. Another aspect of agile is decrease in documentation and relying only on just enough documentation. Within global software development the activities are divided into two areas: onshore activities and offshore activities [2].One of the vital underlying fact why the Software projects are deployed at offshore is that by taking advantage of cheaper labour, facilities and talented workforce [2].

The biggest advantage of offshore development is reduced cost of development [3]. The significant challenge of offshore development is hurdles in communication between the onshore and offshore team. It is more difficult for the client and onshore team to communicate with the offshore project team because of increase in distance that leads to difficulty in face to face meeting (one of primary principles of agile) and also differences in time zones. Both of which can increase the probability of developing the incorrect functionality as misunderstandings and interpretations occur over software requirements [4]. Some of the main issues and challenges in global software development are: inadequate communication, knowledge management, cultural diversity, time differences [5].

The utilization of agile projects in offshore projects could help achieve the advantages of agility (flexibility) and off shoring (maximize cost savings). However agile extremely focuses on communication and feedback thereby demanding a development team that is physically co-located and in close collaboration with its customers or other important stakeholders. It is not easily achievable in distributed setting [4].

2. RELATED WORKS: Literature study

2.1 What is Agility?
Agility is dynamic, context-specific, aggressively change embracing, and growth-oriented. Agility is dynamic, context-specific, aggressively change embracing, and growth-oriented. The core concept in agile is quick response to change [4]. Any methodology should be quicker and cheaper to implement changes in requirements where you cannot lock or freeze requirements in earlier stages [6]. The focus of agile is always on the team and concludes that to effectively respond to

change one have to reduce the cost of moving information between people along with reduction in the elapsed time between making a decision to seeing the outcomes of that decision [4].

Table 1: Agile manifesto versus waterfall model

Agile	Waterfall
Individuals and interactions	processes and tools
Working software	comprehensive documentation contract
Customer collaboration	negotiation Follows a
Responding to change	plan

2.2 Why go for Agile?

The core concept in agile development is that the team can be increasingly effective when the time and cost of moving information and decisions between people is reduced [4]. For this purpose the development team should be co-located thereby increasing communication [3]. For example in case of XP if the development team (mainly programmers) is scattered in two rooms it creates problems for successfully implementing it [7]. Also included are close collaboration with customer and other stakeholders. Another aspect of agile is decrease in documentation and relying only on just enough documentation [3]. In software development there is a continuous flow of requirements. The aspect of welcoming late changing of software requirements in agile [4] makes it suitable for the demanding needs of software development.

2.3 Welcoming Change

In waterfall software development model the software requirements are frozen before moving to the next stage and following the same freezing principle when commencing to the next stage. Agile on the other hand allows the freedom of making changes in software requirements specification even late during the software development [6]. Satisfying the customer at delivery of the software and not during the initial phases has now taken precedence [8]. It means that requirements change during different phases of software development and thus requires flexibility and accommodating change. Requirements cannot be specified absolutely correctly in the specifications.

Iterative development of agile methodologies helps in giving a better understanding of software requirements by the project team. During early iterations changes in software requirements occur that will require its reassessment. These early iterations will remove the ambiguities in software requirements and will minimize the chances of implementing software requirements that will prove costly when changing them later in the software development lifecycle [9].

2.4 Why do Software Projects fail?

According to [10] software projects fail due to a number of reasons. Some of which are mentioned below:

i. Not clearly communicating the requirements

ii. Business problem is not solved by the requirements

iii. Changing nature of requirements

iv. Incorrect requirements

v. Committing resources before fully understating the requirements

For these issues to resolve in context of agile methods it requires regular communication with end-user and other stakeholders involved in the project [4]. It is important because there is less documentation and more face to face communication. This communication between different stakeholders brings them to a common understanding of software requirements. It becomes more important in case of market driven software products where you have many stakeholders and there is the issue of bringing them on a common understanding about software requirements. These different stakeholders have different backgrounds and thus interpret requirements in different ways which requires regular communication and feedback meetings. The technical knowledge of these stakeholders may not be at the same level making face-to-face communications more important for example a software requirement that comes from marketing department is more abstract and when is made concrete or understandable to the development team, its meaning and context may change. Thus marketing department then have to be brought into the communication process to confirm that these software requirements which are now in a new shape have maintained the same meaning. This activity cannot be performed without communicating with them. In case the technical team does not communicate this early on then it can create difficulties for the development team to accommodate these changes later on. By difficulties it is meant that increase in development time and resources spent on the product.

2.5 Requirements Engineering Process

A requirements engineering is a structured set of activities which are followed to derive, validate and maintain a systems requirements document. Process activities include requirements elicitation, requirements analysis and negotiation and requirements validation.

2.5.1 Requirements Elicitation

Requirements Elicitation is about finding requirements through consultation and communication with different stakeholders, studying already existing system documents and using domain knowledge. In other words it also identifies what the system should do and not so. Particular questions that need to be answered are: what activities come inside the scope of the system and that had to be put as software requirements and what activities are outside the scope of the system.

2.5.2 Requirements analysis and negotiation

According to [11] requirements analysis and negotiation are "activities which aim to discover problems with the system requirements and reach agreement on changes to satisfy all system stakeholders". The main goal of requirements analysis is to identify possible conflicts, overlaps, dependencies, inconsistencies or omissions when software requirements are elicited and specified. It is an ongoing process in which all the stakeholders come together to arrive on a concrete set of requirements [12]. Some analysis of software requirements also takes place during the elicitation phase. This is the case when problems with software requirements can be identified as soon as they are expressed. However the extended analysis takes place after the initial version of software requirements document produced [11]. Requirements negotiation can also be included during the analysis phase. Different stakeholders give importance to a software requirement from their own perspective. They also have different levels of power over the decisions being made about particular requirements [11]. There are different ways to analyze software requirements for

example prototypes, mock-ups and test cases that can be used for analyzing and refining the requirements. One of the most important activities of this analysis phase is to ensure that all stake holders from customers to engineers and developers have the same understanding of software requirements. In agile based development the customer is involved in all the phases of iteration.

2.5.3 Requirements validation

Validation is performed to approve that the software requirements are acceptable to be implemented. During this validation phase conflicts between stakeholders are also resolved [11]. It is important to note that execution of software requirements validation process should be done appropriately to save costs that are going to occur later on during the implementation of incorrect requirements. Requirements reviews and inspections can be used to validate software requirements. In reviews a group of people read and analyze requirements, identify any problems and issues and then to find a solution for them [11]. In agile methodologies requirements validation is performed through regular review meetings and acceptance tests. Customers can use the developed software and determine which functionality is implemented and what further needs to be developed. It allows the whole team including the customer to know strength and weaknesses of the design [3].

2.5.4 Requirements Management

A basic requirements management should contain the following activities [11]:

i. Change management of agreed requirements.

ii. Management of interrelationships between those requirements

iii. Management of dependencies between requirements document and other documents produced during the overall software engineering process.

Requirements traceability is a vital activity within requirements management without which it cannot be performed effectively. This traceability information is used to identify what other requirements are/might be affected by making these propose changes [11]. In agile methodologies software requirements are written on index cards or maintained in a product backlog or feature list. The main difference with traditional requirements management is the level of detail in which the software requirements are specified [3].

3. RESEARCH APPROACH

Case studies are used to a larger extent by social science as both a research and a teaching vehicle, especially in IS (information systems) research. The research is based on an interpretive case studies conducted in select Indian organizations engaged in Information Technology (IT) outsourcing to global clients. The case study research strategy is useful for investigating phenomena that are under-researched, complex or difficult to extract from their underlying contexts. We have adopted an interpretive approach since it is through the multiple, inter-subjective views of actors working within the IT cultural enclave environments that concept, theories, and rich insights about the phenomena can emerge. The study was exploratory in nature with the aim to teach from the case study participants about the context of the phenomenon to provide high quality software development as given in agile manifesto. This context dependent knowledge can prove useful in gaining expertise of understanding a practical Indian setting, an outcome relevant to the research objectives. There are three types of studies using the case study method namely intrinsic case study: researcher wants a better understanding of the particular case; instrumental case study: a particular instance is examined to provide insight into an issue or refinement of theory; Collective case study: Researchers may jointly study a number of case studies in order to inquire into the phenomenon, population, or general condition. Since it is a phenomenon of the usage of scrum in software developmental projects, we have adopted the instrumental case study approach to provide better insights into the issue.

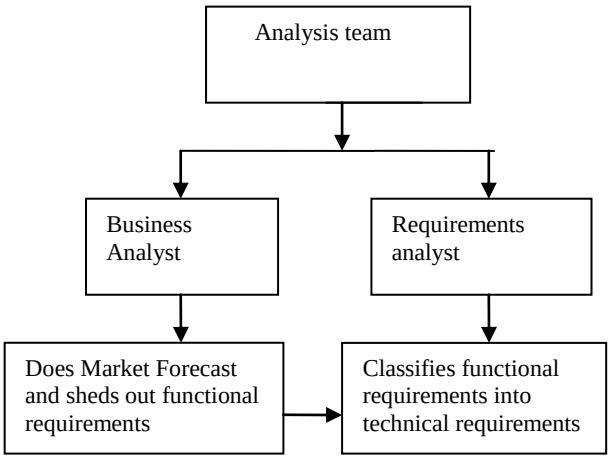
4. EXPERIMENTAL RESULTS

4.1 CASE DESCRIPTION – THE BACKGROUND

The case study is framed based on my involvement in an agile software project that has been developed in a software company located in Chennai, India. The software company which I am referring to is one of the pioneers in handling projects on image and video optimization. As an extension of this work the Software Team has developed logic and is developing the beta version of the product to compress the image and retrieve it back without any loss on the original image. They are also leaders in creating solutions for correcting the online competitive examinations. The company has its offices in various locations across India and has its global operations in countries like USA, Japan and Philippines. As per the policies of the Organization, the company name and the Project names are kept to be anonymous.

4.1.1 THE ANALYSIS TEAM

The schema below shows the members in the analysis team.



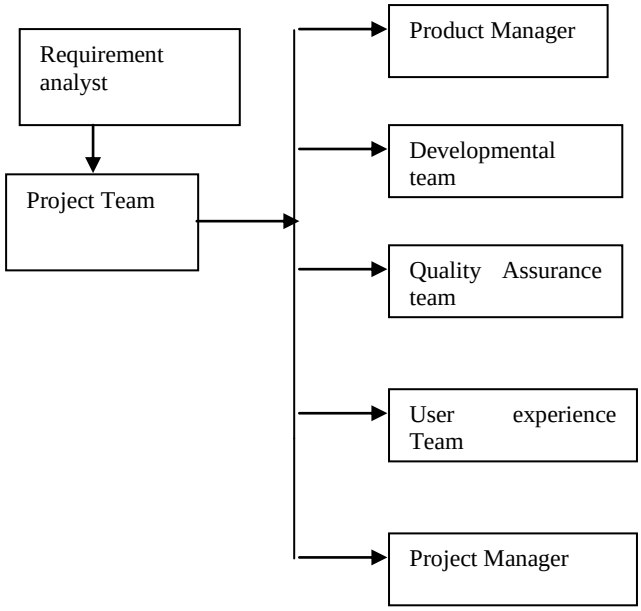
The duty of the business analyst is to forecast the market and to bring in business. He is also responsible for getting the functional requirements. The requirement analyst converts the functional requirements in to the technical requirements. In our company, the delivery head acts as the requirement analyst. He allocates the person and schedules the tasks for

the employees and also responsible for fixing the deadlines for frequent deliveries.

The major difference between a traditional requirement analyst and the agile requirement analyst is that the latter insist on continuous delivery. When continuous delivery is given, continuous feedback from the customer is mandatory. When continuous delivery and continuous feedback go hand in hand, then continuous integration should be done by the developers in a team. Each team should have atleast three to four integration per day. This is the way we work in our Organizational team.

4.1.2 THE PROJECT TEAM

The below illustration shows the Agile cross-functional development team includes a product manager, developmental team, Quality Assurance team, User Experience team, and a project manager.



The project manager supervises the release schedule and helps resolve logistical issues, but otherwise does not participate in the development process. The developmental team, consisting of three to eight developers, implements and delivers to QA a feature or set of features per Sprint based on a prioritized backlog determined by product management. QA tests the features in that Sprint and identifies issues. If any issues are found, engineering team fixes the features in the current Sprint. The sprints that they adopt are two weeks in length, with seven of the ten working days of the Sprint dedicated to implementation by engineering and the remaining three days to Quality Assurance for testing.

4.1.3 THE TEAM

The cross-functional development team for the project consisted of the product manager, five development engineers, two QA engineers and two members of the User Experience team. I was one of the members of the requirement analyst team, who shares the requirements obtained from the client to the cross functional development team, exclusively to the user experience team. The User Experience team was responsible for user research and testing. Two other members were responsible for design and implementation, and regularly participated in Sprint planning and daily Scrums. All members of the team were centrally located in the corporate office at Chennai, India.

4.1.4 USER EXPERIENCE PARTICIPATION

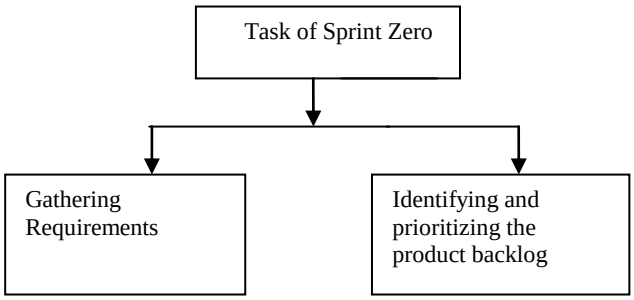
As the requirements and target audience were only partially defined during the initial stages, the product management engaged the User Experience team to research and gather user requirements. The research results obtained enabled the project team to identify the product feature set in preparation for the start of development. The results of user research and testing were instrumental in prioritizing the product backlog. User Experience involvement in the Sprints was a key factor in successfully focusing the cross-functional development team's efforts on the requirements of the users.

4.1.5 USAGE OF SPRINT ON THE PROJECT

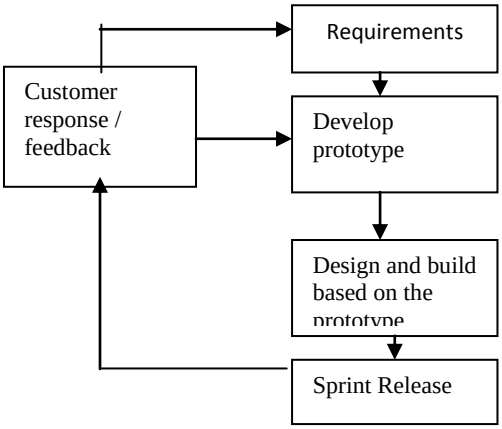
The engineering team and product management were invited to view user testing sessions, discuss findings and review preliminary designs during the sprint session. This helped the cross-functional team to share their ideas, and enabled the engineering team to prepare for the next Sprint. At the end of the Sprint, final designs and specifications were reviewed with the cross-functional team, and then handed off to engineering for development. User Experience participation in the daily Scrums was a crucial factor in the development process. The daily scrum became the convenient forum to share the results of user research and testing, and to clarify any misinterpretations of the designs.

4.1.6 SPRINT ZERO

Sprint zero takes place prior to the start of development.



Sprint zero is used by the cross-functional team to review requirements and create initial user stories based on the backlog items. I (the first author of this article) made a knowledge transfer of the requirements to my team which utilized sprint zero to better understand the users' needs, to explore their context and identify their goals for the project as a whole. The data obtained from the initial user research effort is used to negotiate the priorities of the first Sprint and to communicate the users' expectations to the cross functional team. User scenarios and the features were developed in subsequent Sprints. Designs for features engineering plans to implement in the first Sprint are also created and tested.



The remaining sprints are used in refining the earlier and the subsequent sprints.

4.1.7 Reasons for offshore software development in agile projects

The main reasons for our Organization to move offshore are to reduce cost of development due to lower wages, skilled labor and round the clock development [5]. The different reasons for moving offshore are given below:

i. Improvement in Telecommunications

ii. Favourable Government policies

iii. Skilled and very cheap man power

iv. Less cost of setting offshore site

In convention the offshore projects prefer plan driven approach. Detail requirements and design specification are sent to offshore site where they can develop the software. This approach is useful to minimize the communication hiccups that are a basic product of offshore development. In offshore setting the principles of agile can be successfully executed. Most of the agile principles are based on giving space for decision making to the software development team. This will create more confidence particularly in the offshore team. Also more thinking will be done at offshore bringing in more understanding about the software requirements increasing ownership of the software product. Also more communication takes place between the onshore and offshore teams resulting in much better job performance [13]. The Agile project puts great focus on communication with the end-user or customer. This communication can help in developing trust amongst both the onshore and offshore teams [13].

5. CHALLENGES IN UNDERSTANDING SOFTWARE REQUIREMENTS IN AGILE PROJECTS

The major challenge in understanding the software requirements lies through effective communication and knowledge sharing, domain knowledge, less software requirements documentation. A detailed explanation of the above said features are as follows:

5.1 Effective communication and knowledge sharing

Agile puts more focus on team effort where the teams are physically co-located. But when teams are geographically distributed especially with large time zone differences then it becomes difficult to co-ordinate. The primary reason is that they can no longer talk face to face every day. Although telephones and teleconferencing allow communicating synchronously but still it takes longer to resolve an issue when teams are at a distance from each other [15]. A common problem in communication is that team at one location is waiting for a response from the team at other location creating misunderstandings and irritation with each other [14].

As there is less focus on documentation and more focus on communication then more of the knowledge is un-documented. Communication is not the same as physically co-located teams compared to on and offshore teams therefore knowledge sharing becomes more difficult. Although any kind of software requirements specification acts as a mean of knowledge sharing but not everyone in the team will clearly understand the specification and will have questions to pose.

5.2 Domain knowledge and product knowledge

Most of the development is done offshore and the role of offshore team has much greater importance. According to agile principles every team member has a say and a much greater role than the waterfall model [4]. In agile there is less focus on documentation and more on developing code therefore a lot of knowledge that is developed walks out with the person who leaves the company, which is said to be a major flaw in the methodology.

5.3 Less software requirements documentation

Less documentation in agile puts more focus on the real implementation of software requirements. This reduces the time spent on software requirements specification that can be utilized in doing software development. But less documentation puts more on communication within and between agile teams. If the team is physically co-located questions regarding the software requirements could be easily resolved. Due to less detail of software requirements, it would be difficult for the quality assurance team to perform major software testing at the offshore site.

5.4 Challenges in conducting reviews

Yet another challenge that we faced in our Organization is that in not getting the clients to review early builds. Thus offshore team has to wait for a long time to get a response from the client. Transfer of tacit knowledge from the business (which is onshore) to the offshore team which acted as a challenge as details of this tacit knowledge that is not present in the software requirements specification. Validation of software requirements by the offshore team due to change in business priorities becomes a challenge. It could be possibly that not knowing the full business context leads to difficulties in software requirements validation by the offshore team.

6. CONCLUSION AND SOLUTION

This article focused on the challenges caused due to lack of communication, language problem, cultural differences and difficulties in knowledge sharing due to physical distance.

The solution of the article is to have regular communication between the onshore and offshore site to make transfer of knowledge easier. This regular communication is useful to resolve any type of conflicts and misunderstandings about software requirements. In shorter iterations lack of communication is bigger problem than in longer iterations due to physical distance between on and offshore teams and lack of time to implement software requirements. As a solution either the iteration duration could be increased or software requirements could be decreased for iteration to deal with lack of communication and difficulties in knowledge sharing.

7. ACKNOWLEDGMENTS

My sincere thanks to my Organization who gave me an opportunity to enrich my software knowledge without which I would have not written this article.

8. REFERENCES

[1]. Tony Gorschek, Claus Wohlin, "Requirements Abstraction Model", Springer-Verlag London, dated: 26-11-2005.

[2]. Muhammad Faisal Nisar, Tahir Hameed, "Agile development handling offshore Software Development

issues", Multitopic conference proceedings of INMIC 2004. 8th International, dated: 2004-Dec-24-26.

[3]. Frauke Paetsch, Dr. Armin Eberlein, Dr. Frank Maurer, "Requirements Engineering and Agile Software Development", Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE,,03), dated: 2003.

[4]. Alistair Cockburn, Jim Highsmith, "Agile Software Development: The People Factor", Software Management, dated: 2004.

[5]. Ernest Ferguson, Clifton Kussmaul, Daniel D. McCracken, Mary Ann Robbert, "Offshore Outsourcing: Current Conditions & Diagnosis", Technical Symposium on Computer Science Education, Proceedings of the 35th SIGCSE Technical symposium on Computer Science Education, Norfolk, Virginia, USA, pp: 330-331, ACM Press New York, Ny, USA, dated: 2004.

[6]. Renee McCauley, "Agile Development Methods Poised to Upset Status Quo", Vol. 3, Iss. 4, ACM SIGCSE Bulletin, dated: Dec-2001

[7]. Kent Beck, "Embracing change with extreme programming", Computer Journal, Vol. 32, Iss. 10, pp: 70-77, IEEE, dated: 1999.

[8]. Jim Highsmith, Alistair Cockburn, "Agile Software Development: The Business of Innovation", Computer, Vol. 34, Iss. 9, pp: 190-127, IEEE, dated: Sep-2001

[9]. Phillip g. Armour, "Agile…and Offshore", Communications of the ACM, Vol. 50, Iss. 1, ACM Press New York, NY, USA, dated: Jan-2007.

[10]. Lowell Lindstrom, Ron Jeffries, "Extreme Programming and Agile Software Development Methodologies", Vol. 21, Iss. 3, pp: 41-52, Information Systems Management, dated: 2004.

[11]. Gerald Kotonya, Ian Sommerville, *"Requirements Engineering Processes and Techniques"*, John Wiley & Sons Ltd, England, dated: 1998

[12]. Colin Potts, Kenji Takahashi, Annie I. Anton, "Inquiry-Based Requirements Analysis", Vol. 11, Iss. 2, pp: 21-32, IEEE Software, dated: March-1994

[13]. Ajay Danait, "Agile Offshore Techniques – A Case Study", Proceedings of Agile Development Conference, pp: 214-217, IEEE, dated: 24-29-Jul-2005

[14]. Keith Braithwaite, Tim Joyce, "XP Expanded: Distributed Extreme Programming", Springer-Verlag, pp: 180-188, Berlin, dated: 2005.

[15]. Joachim Sauer, "Agile practices in offshore outsourcing – an analysis of published experiences", Proceedings of the 29th Information Systems Research Seminar in Scandinavia, Helsingborg, Denmark, dated: Aug-12-15-2006

[16]. Marjanovic 2009. "Inside Agile Processes: A Practitioner's Perspective", 42nd Hawaii International Conference on system Sciences, 2009 (HICSS '09). 05 – 08 Jan 2009, Sydney, Australia, IEEE, Pp. 01 – 10.

[17]. Bin Xu (2009), "Towards High Quality Software Development with Extreme Programming methodology: Practices from real software projects, IEEE Xplore