Effective Implementation of Agile Practices

Ingenious and Organized Theoretical Framework

Veerapaneni Esther Jyothi
Lecturer, Department of Computer Applications,
V.R. Siddhartha Engineering College,
Kanuru, Vijayawada – 520 007, Andhra Pradesh, India.
nejyothi@hotmail.com

K. Nageswara Rao
Professor and Head, Department of Computer Science and Engineering,
P.V.P.Siddhartha Institute of Technology,
Kanuru, Vijayawada – 520 007, Andhra Pradesh, India.
drknrao@ieee.org

Abstract—**Delivering software in traditional ways is challenged by agile software development to provide a very different approach to software development. Agile methods aim at fast, light and efficient than any other vigorous method to develop and support customers business without being chaotic. Agile software development methods claim to be people-oriented rather than process-oriented and adaptive rather than predictive. Solid Determination and Dedicated efforts are required in agile development to overcome the disadvantages of predefined set of steps and changing requirements to see the desirable outcome and to avoid the predictable results. These methods reach the target promptly by linking developers and stakeholders.**

The focus of this research paper is two fold. The first part is to study different agile methodologies, find out the levelheaded difficulties in agile software development and suggests possible solutions with a collaborative and innovative framework. The second part of the research paper concentrates on the importance of handling traceability in agile software development and finally proposes an ingenious and organized theoretical framework with a systematic approach to agile software development.

Keywords-Traceability; requirements; agile manifesto; framework

I. Introduction

Over the years many different software methodologies have been introduced and used by the software engineering community. Developers and users of these methods have invested significant amount of time and energy in improving and refining them. The method they choose for software development depends on the type of organization, the type of project and the type of people involved. Agile software development methodologies have been greeted with enthusiasm by many software developers because work is done at different levels in parallel [9]. We can use our creativity on the design level you can also have the fun of implementing the design in a smart way.

Handling unstable and volatile requirements throughout the development life cycles and delivering products in short time frames and under budget constraints when compared with traditional development methods are the two most significant characteristics of the agile approaches. Successful agile traceability processes incorporates the carefully devised planning stage to determine when, how, where and why each traceability link will be created.

This paper is organized as follows; section II presents the comparative study of different agile methodologies. Section III Surveys the major worthwhile risks associated with agile software development and suggests possible solutions. Section IV explains the importance of traceability in agile. Finally section V proposes ingenious and organized theoretical framework.

II. Comparitive Study of Agile Methods

Agile methods are a family of development techniques designed to deliver products on time, on budget, with high quality and customer satisfaction [15]. This family includes several and very different methods. The most popular include:

- eXtreme Programming (XP)
- Scrum
- Dynamic Systems Development Method (DSDM)
- Adaptive Software Development (ASD)
- The Crystal family

XP is a deliverable and disciplined approach to agile software development [1]. XP is successful because it stresses customer satisfaction and allows the software developers to confidently respond to changing software requirements even late in the lifecycle. The business culture affecting the development unit is another focal issue in XP.

Scrum approach has been developed for managing the systems development process [8]. It is an empirical approach applying the ideas of industrial process control theory to software development resulting in an approach that reintroduces the ideas of flexibility, adoptability & productivity.

Scrum concentrates on how the team members should function in order to produce the system flexibly in a constantly changing environment [8].

The fundamental idea behind DSDM is that instead of fixing the amount of functionality in a product, and then adjusting time and resources to reach that functionality, it is preferred to fix time and resources, and then adjust the amount of functionality accordingly. DSDM is a non-profit and non – proprietary framework for rapid application development (RAD) maintained by the DSDM consortium.

Adaptive software development is a lightweight software development method that accepts continuous change as the norm. The method follows a dynamic lifecycle, Speculate-Collaborate-Learn, instead of the traditional, static lifecycle, Plan-Design-Build. ASD emphasizes continuous learning. It is characterized by constant change, reevaluation, peering into an uncertain future, and intense collaboration among developers, testers, and customers [16]. ASD was designed for projects that are characterized with high-speed, high-change, and uncertainty.

The crystal family of methodologies includes a number of different methodologies for selecting the most suitable methodology for each individual project. Besides the methodologies, the crystal approach also includes principles for tailoring the methodologies to fit the raring circumstances of different projects. A team using crystal clear should be located in a shared office-space due to the limitations in its communication structure. Both the crystal clear as well as crystal orange suggest the following policy standards:

- incremental delivery on a regular basis
- progress tracking
- direct user involvement
- automated regression testing of functionality
- two user reviewing per release

TABLE I. TEAM SIZE IN CRYSTAL FAMILY

Methodology	**Team (n° people)**
Crystal Clear	2-6
Crystal Yellow	6-20
Crystal Orange	20-40
Crystal Red	40-80

TABLE II. COMPARISON OF AGILE METHODS

Concept	**XP**	**SCRUM**	**DSDM**	**CRYSTAL**	**FDD**
Team size	3-16	5-9	2-6	4-8	6-15
Number of teams	1	1-4	1-6	1-10	1-3
Volatility	high	high	low	high	low
Team distribution	no	no	yes	yes	yes

III. LEVELHEADED DIFFICULTIES IN AGILE SOFTWARE DEVELOPMENT AND THE SUGGESTED SOLUTIONS

Collaboration between customers and development team is important factor of agile software development. Even though the agile methods are meant for fast delivery under budget constraints there are certain levelheaded difficulties in agile software development.

Below are such difficulties which are worthwhile to be considered:

- Need high-quality collaboration between customers and agile development team.
- Need a high-level of customer involvement.
- Lack of long-term detailed plans.
- Producing a lower level documentation.
- Misinterpreted as unplanned and undisciplined.

Requirements are the base of all software products and their elicitation, management, and understanding are very common problems for all development methodologies [2]. In particular, the requirements variability is a major challenge for all commercial software projects. Five of the eight main factors for project failure deal with requirements incomplete requirements, low customer involvement, unrealistic expectations, changes in the requirements, and useless requirements.

TABLE III. MAIN CAUSES OF PROJECT FAILURE

Problem	**%**
Incomplete requirements	13.1
Low customer involvement	12.4
Lack of resources	10.6
Unrealistic expectations	9.9
Lack of management support	9.3
Changes in the requirements	8.7
Lack of planning	8.1
Useless requirements	7.5

A. Quality facilitator

Figure1 portrays the responsibilities of the Quality Facilitator in different aspects of the agile software development process. QF should be responsible of the effective management of changing requirements which is an important factor to be concentrated on in order to maximize stakeholder ROI. QF along with the agile team should as well address the issues of maintenance and support so as to maintain high discipline and good engineering principles. QF is responsible in the following aspects of different issues which are discussed.

Agile Project Management is an iterative method of determining requirements for software and for delivering projects in a highly flexible and interactive manner [4]. It requires empowered individuals from the relevant business with supplier and customer input. Agile project management takes the ideas from agile software development and applies them to project management. Agile methodologies generally promote a project management process that encourages stakeholder involvement, feedback, objective metrics and effective controls [11].

Software configuration management is a process for developing and maintaining concurrency of the product’s performance, functional, and physical attributes with its requirements, design and operational information throughout its life [9]. Hence both the agile teams and the quality facilitator should focus on the configuration management.

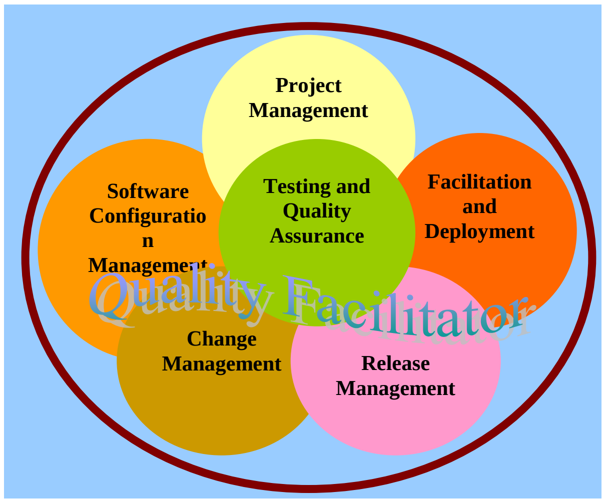

Project Management
Software Configuration Management
Testing and Quality Assurance
Facilitation and Deployment
Quality Facilitator
Change Management
Release Management

Figure 1. Responsibilities of Quality Facilitator

Agilists want to develop software which is both high-quality and high-value, and the easiest way to develop high-value software is to implement the highest priority requirements first. The activities contributing to effective change management are motivating change, creating vision of change, developing political support, managing the transition of change and sustaining momentum [11].

Teams practicing Agile Software Development value working software over other artifacts. A feature from the release plan is not complete until you can demonstrate it to your customer, ideally in a shippable state. Agile teams strive to have a working system ("potentially shippable") ready at the end of each iteration [11]. Thus Release Management should be easy for an ideal agile team, as agile teams, in theory are ready to release at regular intervals, and the release management aspect is the customer saying "ship it!."

Deployment is the point where an application starts to provide a return on the development investment. Delivering software doesn't stop once the application is written. It needs to be built, assembled and deployed, and historically this has been a non-trivial, manual and an error-prone task.

Agile testing does not emphasize testing procedures and focuses on ongoing testing against newly developed code until quality software from an end customer's perspective results. Agile testing is built upon the philosophy that testers need to adapt to rapid deployment cycles and changes in testing patterns [13].

Agile development processes place both added importance as well as special demands on your software quality assurance (SQA) practices. As an integrated part of the agile team, testers participate in the full life-cycle from requirements through release [9]. While many of the principles and practices of software quality assurance apply, an agile approach requires some new ways of viewing testing activities in the development process. Successful outsourcing projects are the ones that strike a good balance between testing and quality assurance throughout the lifecycle of the project [13].

B. Collaborative and innovative framework

To generate a significant impact on the productivity of the project, companies should adopt a hybrid approach that is using scrum and adding XP engineering practices.

Unlike XP, projects are wrapped by scrum, they becomes scalable and can be run simultaneously by distributed teams[1]. XP can be gradually implemented within scrum framework. Scrum is a project management approach, whereas XP is a methodology for project development [6]. Scrum only focuses on the managing side that is on what needs to be done rather than how to do it. XP on the developer side uses test driven development, pair programming, refactoring, etc. which are very essential to build good quality software but are missing in scrum [5].

Since Scrum doesn't have any engineering practices and XP doesn't have any management practices, XP with scrum projects allows better value metrics process for measuring and managing initiative ROI [10].

Planning includes the definition of the system being developed and the definition of the project team, tools and other resources, risk assessment, controlling issues, training needs and verification management approval.

The requirements can originate from the customer, sales and marketing division, customer support or software developers. The requirements are prioritized and the effort needed for their implementation is estimated. The product backlog list is constantly updated with new and more detailed items, as well as with more accurate estimations and new priority orders. Sprint backlog lists product backlog items selected to be implemented in the next iteration.

Unlike the product backlog, sprint backlog is stable until the sprint is completed. A new iteration of the system is delivered after all the items in the sprint backlog are completed.

Figure 2 is the framework, a hybrid approach in which XP engineering practices are implemented in the scrum sprint [5]. Sprints are iterative cycles where the functionality is developed or enhanced to produce new increments. Each Sprint includes the traditional phases of software development: requirements, analysis, design, evolution and delivery phases. These phases are implemented using extreme programming methodology. The functional tests created by the customer are run at the end of each iteration.

Here the key characteristics of XP are included such as refactoring - restructuring the system by removing duplication, improving communication, simplifying and adding flexibility without changing its functionality, pair programming – two people write the code at one computer which is great for complex and critical logic [11], collective code ownership – code belongs to the project not to any individual engineer, continuous integration – a new piece of code is integrated into the code-base as soon as it is ready. Thus the system is integrated and built many times a day [7]. All tests are run and passed for the changes in the code to be accepted.

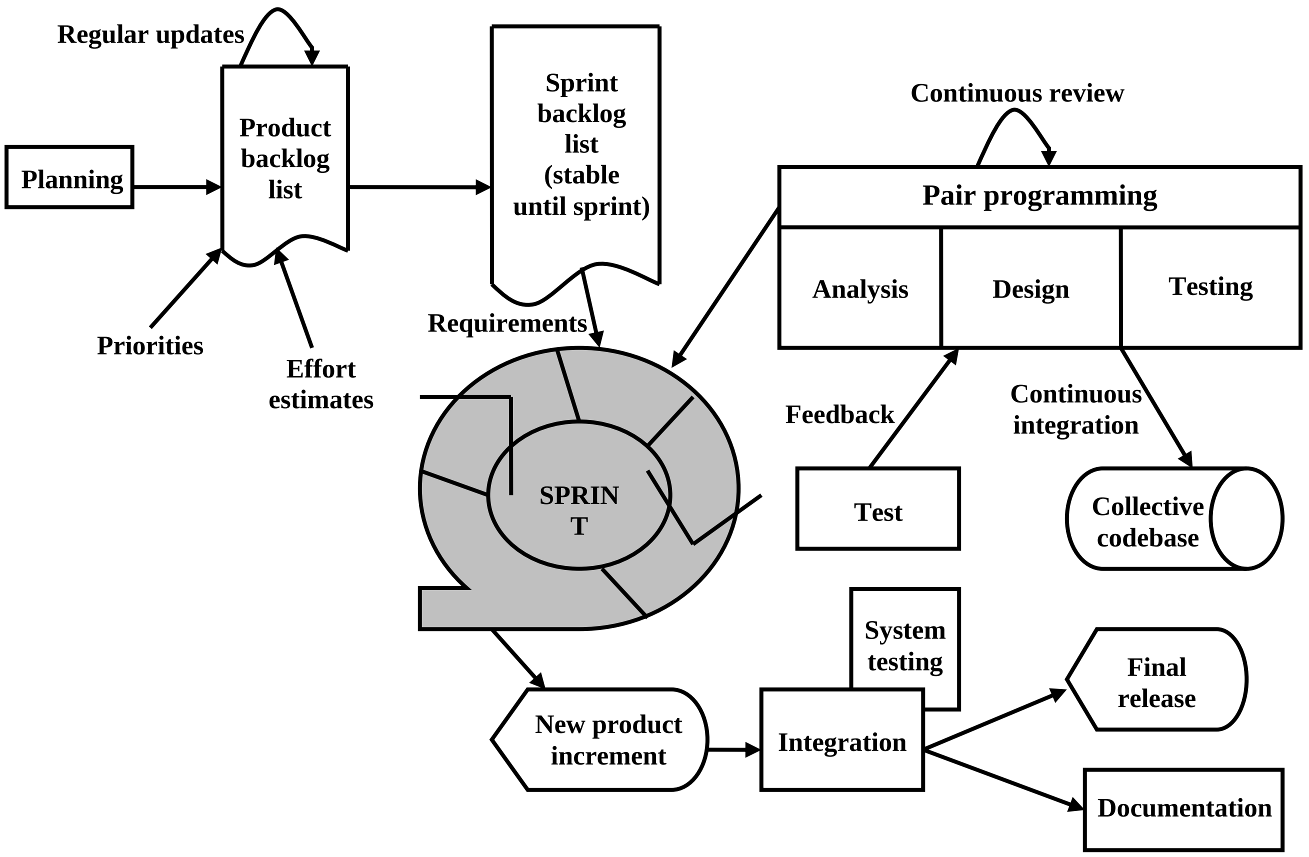


Figure 2. A Collaborative and innovative framework

Extra testing and checking of the performance of the system before the system can be released to the customer. At this stage, new changes may still be found and the decision has to be made if they are included in the current release. The postponed ideas and suggestions should be documented for later implementation [9].

Communication and coordination between project members should be enabled at all times. Any resistance against XP practices from project members, management or customers may be enough to fail the process. Ultimately better results can be obtained by tailoring some of the scrum principles such as the daily scrum meeting to keep track of the progress of the scrum team continuously and they also serve as planning meetings [8] [13].

Developing software on time and within budget is not good enough if the product developed is full of defects and customers today are demanding higher quality software than ever before. Now-a-days the software market is mature enough and users want to be assured of quality [5]. Due to time-to-market pressures or cost considerations, the developer may limit the software quality assurance function and not choose to conduct independent reviews.

IV. TRACEABILITY IN AGILE

This section of the research paper explains the importance of traceability in agile methods, how much level of traceability should be added and how to add traceability to agile methods.

A. *The Importance of Traceability*

Traceability is an important part in agile software development. Today there are many conflicts on whether the traceability is important in agile methods. During the study we have considered several opinions of different people of agile teams. Majority of people articulated that if traceability is supposed to be added in agile methods it should see that it will not place much administrative overhead to the team and also it should be considered on how the team will be benefited from the gathered information.

Traceability done in agile methods should help keep all the information gathered, organized and easy to find. It should ensure that the teams involved in development should find a way to look at how different artifacts are linked. Also it is important for teams to be capable of tracing the information and the decisions that were made during the entire process [14].

To generate a significant impact on the productivity of the project, companies should adopt a hybrid approach that is using scrum and adding XP engineering practices as discussed in section III. The tracing practices that apply to adding traceability to Scrum can be applied to XP and vice-versa.

B. *Levels of AddingTtraceability*

Some level of tracing is always considered to be useful for all the projects to be developed. There are several situations where it is good to add traceability in the agile methods. Some projects are suitable to add traceability completely where as some are suitable to add traceability to the minimum.

Adding traceability to a project depends on the life span of the project. The organization should consult the customer or stake holder on how long the product will be used. The level of traceability to be added depends on the life span of the product. If the life span is too short it is better to avoid traceability in agile methods.

If the life span is about one year there is no need of adding all the traceability and the documentation. If the life span for the project is about to change then the traceability should be added from the beginning of the development itself. If the life span last long it is better to add full tracing. The customer or stake holder is uncertain about the life span of the project it is better to add traceability to a certain level.

Adding traceability to a project depends on the size of the project [12]. If the size of the project is small it is difficult to know how much the level of traceability is added. When Scrum or XP agile methods are used to develop small projects the level of traceability can be quite minimum. If the size of the project is large and complex, the team should document their work from the beginning of the development. Hence the level of traceability to be added should be maximum for such projects.

C. The Tracing Practices in Adding Traceability

In order to add traceability in agile methods there are several tracing practices which can be used individually or combined with each other. Below are several tracing practices:

1) *Trace the requirements to stakeholder.*
2) *Trace the requirements to problem description.*
3) *Trace the requirements to product backlog.*
4) *Trace the requirements to sprint backlog.*
5) *Trace the requirements to code.*
6) *Trace the requirements to database tables.*
7) *Trace the requirements to test.*
8) *Documentation.*

V. INGENIOUS AND ORGANIZED THEORETICAL FRAMEWORK

Agile Alliance formulated their ideas into values and further to twelve principles that support those values. Values of Agile Manifesto are as follows:

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

The above values are realized in the principles of Agile Manifesto[3].

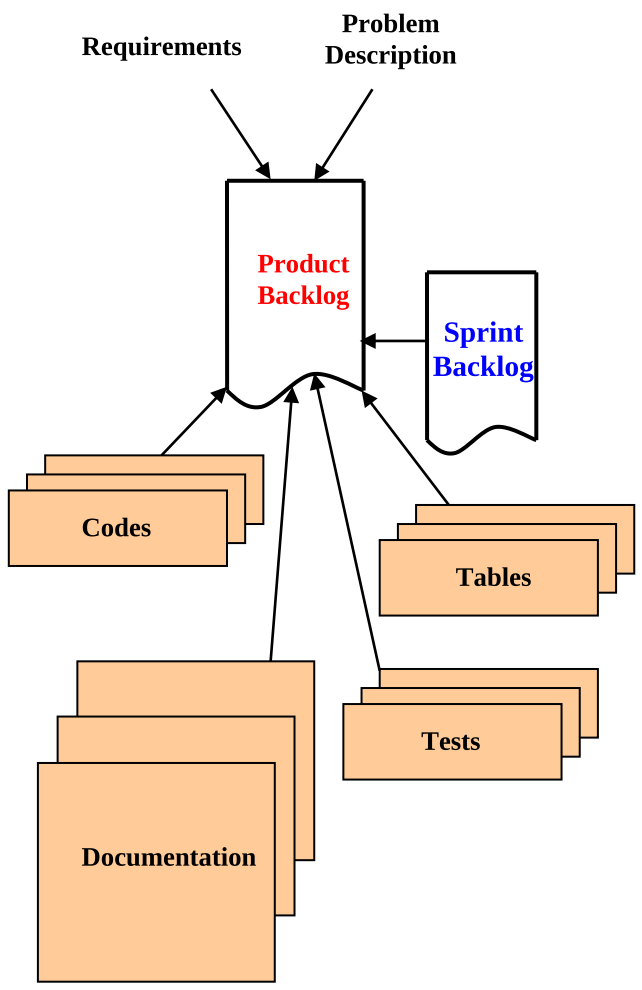


Figure 3. Traceability practices

The eleven principles among the twelve are chosen to model ingenious and organized theoretical frameworks which are as follows:

1) *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2) *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3) *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4) *Business people and developers must work together daily throughout the project.*

5) *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6) *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7) *Working software is the primary measure of progress.*
8) *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9) *Continuous attention to technical excellence and good design enhances agility.*
10) *Simplicity--the art of maximizing the amount of work not done--is essential.*
11) *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

To improve software development and to maximize the ROI (Return On Investment), it is not merely sufficient if the organization just understand the principles of Agile manifesto but indeed implementing them in right way and in right situations is more important. In order to facilitate agile software development, all the aspects of the organization are to be considered. The four different aspects that lead the organization are internal, external, technical and social aspects. Each aspect should be handled very carefully.

The ingenious and theoretical framework given below clearly shows which of the agile manifesto principles are to be followed in each aspect of the system. By doing so working software can be produced using agile software development with faster delivery and within budget. Not all the principles are relevant enough to be followed in all the aspects of the organization. With careful examination and with experience the framework given suggests what principles are to be followed in each and every aspect of the organization and their interdependencies through which working software can be produced which satisfies both customers and the development team.

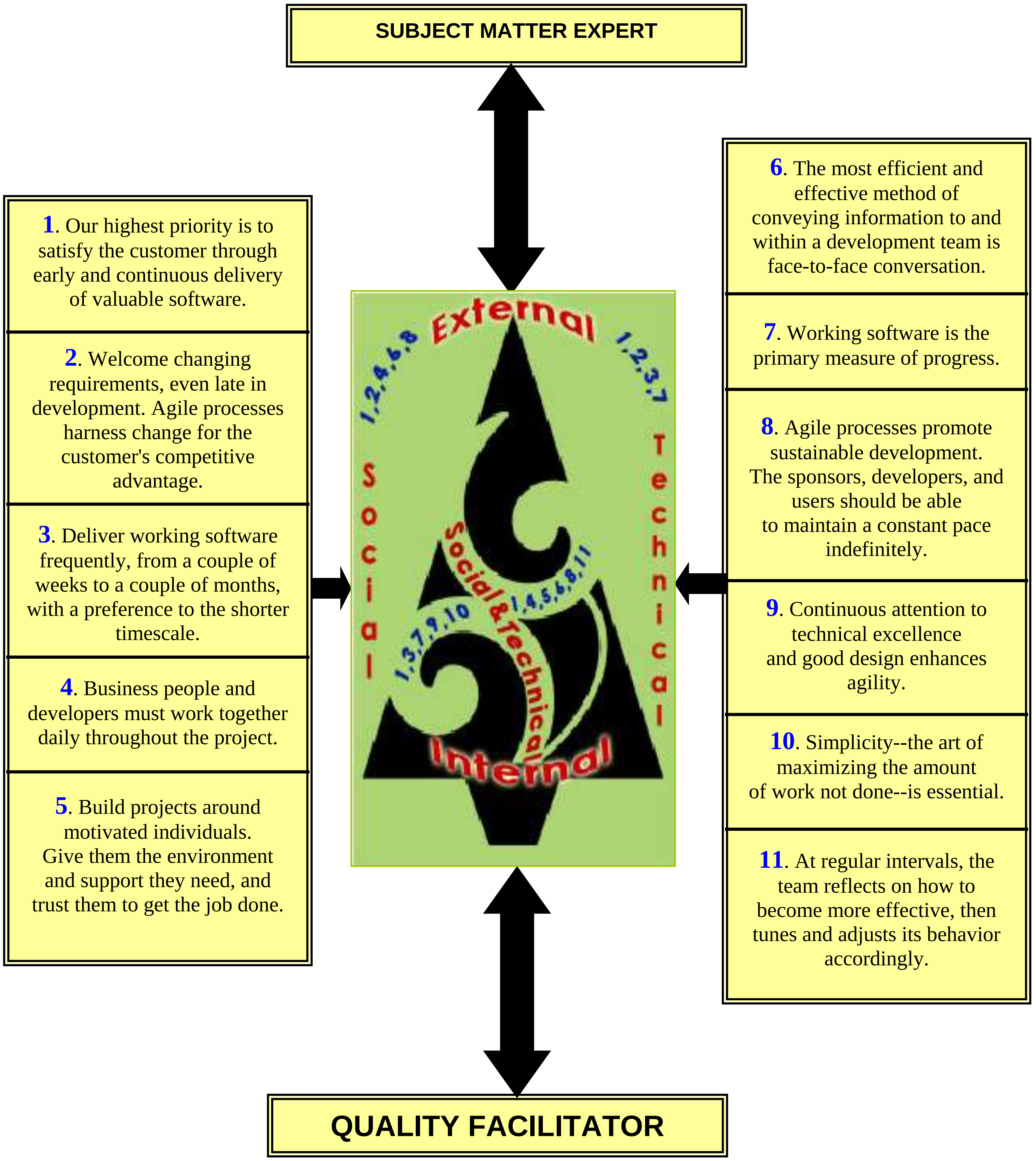
Internal aspects in the organization relate to the development team. External aspects relate to the customers or stakeholders. Technical aspects relate to different stages of development. Social aspects relate to the people, their working style, communication, job satisfaction.

The first principle which tells team to show highest priority as mentioned above relates to all the four aspects of the organization. The second principle relates to external - social aspects and also external – Technical aspects since it deals with the basic goal of agile software development. The third principle relates to technical aspects with external implementation and also to technical aspects with internal implementation. Principle four relates to social aspects with internal and external implementation since it deals with coordination and collaboration between people and developers.

The fifth principle relates to internal –social aspects of the organization, principle six relates to social aspects in coordination with to internal and external aspects since communication is the most important issue. Principle seven relates to technical aspects with external implementation and also to technical aspects with internal implementation which is the ultimate goal of agile software development. The eighth principle relates to social aspects with external implementation and also to social aspects with internal implementation

The ninth principle which is the key strength and success factor of the organization relates to technical and internal aspects, principle 10 also relates to technical and internal aspects, the eleventh principle relates to internal and social aspects.

The involvement of the subject matter expert is very important for smooth running of the project. Since the subject matter experts have the domain knowledge he can help the development team in following the principles and practices.



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

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

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

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

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

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

7. Working software is the primary measure of progress.

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

9. Continuous attention to technical excellence and good design enhances agility.

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

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

Figure 4. An ingenious and organized theoretical framework

CONCLUSION

Most of the agile techniques are repacking of well-known techniques like ignoring documentation of meetings, maintaining reviews, writing test cases before writing code, etc. This research paper suggests that small and mid-sized companies can have a Quality Facilitator who should play an important role to eliminate all the discussed worthwhile risks to the extent possible to produce a defect free product. Since certifications like ISO, CMMI emphasize more on documentation of the activities/work items, we need to concentrate on the documentation work for agile also [10]. So our research suggests a QF role for ensuring the process is followed and the documentation is done properly so that there won't be any quandary for the project while facing the internal/external audits.

The proposed collaborative and innovative framework can be implemented along with the suggested possible solutions for the mentioned levelheaded difficulties. Also the organizations have to take steps in implementing the suggested theoretical framework which can lead in developing successful projects.

REFERENCES

[1] Weisert, C., (2002), The #1 Serious Flaw in Extreme Programming (XP), Information Disciplines, Inc., Chicago. [Online]. Available from: http://www.idinews.com/Xtreme1.html.

[2] B. Boehm and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods," IEEE Computer, vol. 36, no. 6, pp. 57-66, June 2003.

[3] Agile Alliance.2002. Agile Manifesto. http://www.agilealliance.org/

[4] Coram, M., and Bohner, S., "The Impact of Agile Methods on Software Project Management", In Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS), April 2005, pp. 363-370.

[5] V. Esther Jyothi and K. Nageswara Rao, "Effective implementation of agile practices – A collaborative and innovative framework", published in CiiT International Journal of Software Engineering and Technology, Vol 2, No 9, September 2010.

[6] Maurer, F. and Martel, S. (2002a). Extreme programming: Rapid development for Web-based applications. IEEE Internet Computing 6(1): 86-90.

[7] Highsmith, J. and Cockburn, A. (2001). Agile Software Development: The Business of Innovation. Computer 34(9): 120-122.

[8] Schwaber, K. and Beedle, M. (2002). Agile Software Development with Scrum. Upper Saddle River, NJ, Prentice-Hall.

[9] Robert C. Martin, Agile Software Development, Principles, Patterns, and Practices, Prentice Hall, 2002

[10] Baker, Steven W. Formalizing Agility: An Agile Organization's Journey toward CMMI Accreditation. Agile 2005 Proceedings, IEEE Press, 2005.

[11] Alleman, Glen. Agile Program Management: Moving from Principles to Practice. Agile Product & Project Management, Vol. 6 No. 9, Cutter Consortium, September, 2005.

[12] Leffingwell, Dean, Scaling Software Agility: Best Practices for Large Enterprises, Addison-Wesley Professional, 2007.

[13] Anderson, David J., and Eli Schragenheim, Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results, Prentice Hall, 2003.

[14] Poppendieck, Mary and Tom Poppendieck, Lean Software Development: An Agile Toolkit. Addison Wesley Professional, 2003.

[15] Ming Huo, June Verner, Liming Zhu, Mohammad Ali Babar, "Software Quality and Agile Methods", COMPSAC '04, IEEE 2004.

[16] Craig Larman, Victor R. Basili. Iterative and Incremental Development: A Brief History, 0018-9162/03/ © 2003 IEEE

[17] Salem, A. M. (2010). A Model for Enhancing Requirements Traceability and Analysis. International Journal of Advanced Computer Science and Applications - IJACSA, 1(5), 14-21.

AUTHORS PROFILE

Mrs. Veerapaneni Esther Jyothi is a Microsoft certified professional, currently pursuing Ph.D Computer Science from Raylaseema University, Kurnool. She is working as a Lecturer in the department of Computer Applications, Velagapudi Siddhartha Engineering college since 2008 and also has Industrial experience. She has published papers in reputed international conferences recently. Her areas of interest include Software Engineering, Object Oriented Analysis and Design and DOT NET.

Dr. K. Nageswara Rao is currently working as Professor and Head in the Department of Computer Science Engineering, Prasad V. Potluri Siddhartha Institute of Technology, Kanuru, Vijayawada-7. He has an excellent academic and research experience. He has contributed various research papers in the journals, conferences of International/national. His area of interest includes Artificial Intelligence, Software Engineering, Robotics, & Datamining.