Chapter 7

Achieving Agility through BRIDGE Process Model: An Approach to Combine the Agile and the Disciplined Software Development

7.1 Introduction

Many Software Development Life Cycle (SDLC) process models and approaches have been introduced till date i.e. Waterfall model, Spiral model, Prototype Model, Evolutionary development etc (19). But, rarely any of these models exactly fits as-it-is for modern software development projects (130). Hence, often instead of a single process model, most industry follows a sandwich or hybrid model i.e. mix-up of different models on demand basis (131). Despite of this, very often the customers are unhappy with the end product and many the projects even had to fail! Agile software development philosophy came out to be a successful approach to increase the project success rate up to significant extent while reducing the development time and cost. Agile software development (132) philosophy has been proved to be quite successful to increase the project success rate up to a significant extent. Despite of its success, many authors have criticized the agile approach as it often violates the basic theories, principles and practices of traditional software engineering. But the ability to meet client needs and the delivery of quality software products within estimated time is the significant benefits of agile development and is the key to its survival. Some of the additional benefits of agile process are increased productivity, expanded test coverage, improved quality,

Based on the publication in the "*Innovations in Systems and Software Engineering (ISSE): A NASA Journal*", ISSN: 1614-5046, 11(1), 1–7, 2014. [DOI: 10.1007/s11334-014-0239-x]

fewer defects, reduced development time and costs, delivery of better understandable and maintainable code, improved morale, better collaboration, and higher customer satisfaction as pointed out by Vijayasarathy (133). But as said by Boehm et. al (134, 135), neither the agile nor the traditional disciplined approach alone provide the ultimate approach. Further Hashmi et. al (136) says, there are ongoing debates whether the quality of the products of the agile approaches are satisfactory. In addition, Fritzsche et.al (137) and Theunissen et.al (138) mentioned, some projects e.g. safety-critical projects require standards to be followed when developing software. On the other hand, the two seem contradicting (139, 140), but several researchers agree that a software project needs both agile and discipline (134, 141, 142). New SDLC models are introduced at regular basis as new technology and new research results are needed to be accommodated over the time for modern project needs and suitability. This was the primary philosophy behind the introduction of BRIDGE software development life cycle process model by Ardhendu (62).

7.2 Scope of This Study

In this chapter we have just considered the theoretical aspects for the purpose of analysis and as evident. We didn't consider any real development situations or data for this instance. Presently we are running three experimental projects in our department in parallel to discover the impact of the BRIDGE process model. But, as it shall take more time to complete the projects and to evaluate the successfulness, we simply skipped this for the time being. The real development situations will be disclosed in the future work when the projects will be completed and the experimental results shall be available to us.

7.3 From Disciplined SW Development Approach towards Agile

7.3.1 Limitations of The Traditional Development Models

We have many traditional SDLC models i.e. Classical Waterfall Model, Iterative Waterfall Model, Spiral Model, RAD model etc. in our hand by the time, but a very few are really used in practice exactly as it is. There exist several criticisms about these traditional models. According to Nandhakumar and Avison (143), traditional methods

are too mechanistic to be used in detail. Truex et al. (144) pointed out that traditional SDLC models are more dogmatic and claim that traditional methods are merely unattainable ideals and hypothetical "straw men" that provide normative guidance to utopian situations. Further, Wiegers (145) noticed that, industrial software developers have become skeptical about "new"solutions that are difficult to grasp and thus remain not used. Baskerville et al. (146) claim that "to compete in the digital economy, companies must be able to develop high quality software systems at "Internet speed"—- that is, deliver new systems to customers with more value and at a faster pace than ever before". The primarily perceived limitations of the traditional development models are:

- There are too much work associated with documentation
- They are too sequential
- They requires too much of planning activities
- It does not show results until the end
- It engages stakeholders too late
- Delay in project delivery
- Increased Project Cost

For these reasons, the traditional software development approaches are rarely used in industries these days as they lack suitability for the purpose.

7.3.2 Agile Development : The Origin and its Necessity

Agile principles evolved to address the above criticisms and primary limitations of the traditional software development. Agile software development is neither a set of tools nor a single methodology. Rather, agile is a philosophy appeared as recommendations in 2001 with an initial 17 signatories. While the publication of the "Manifesto for Agile Software Development"(132) didn't starts the move to agile methods, which had been going on for some time, it did signal industry acceptance of agile philosophy. Further, Kieran Conboy (147) in his paper has brilliantly derived the functional definition of agile as "the continual readiness of an information system development method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment".

Agile was a significant departure from the heavyweight document-driven traditional software development methodologies in general used at the time. Agile software development stresses rapid iterations, small and frequent releases, and evolving requirements facilitated by direct user involvement in the development process. In this way, development of agile methods could be seen as cumulative methods built on existing traditional methods where the *good* parts are kept and the *bad* parts are omitted or modified.
As per the recent survey report on state of agile by Versionone (148), shows that 88% of the respondent organizations are practicing agile development, 52% of the projects are being developed following agile.
From the Agile Manifesto (132, 149) and the definition of agility cultivated by Conboy (147) we may extract and combine the following primary feature of agile methods to be put on focus:

- Individuals and interactions
- Working software delivery
- Customer collaboration
- Rapid system change incorporation i.e. responding to change
- Economic development
- Quality product development
- Simplicity
- Enhance knowledge from change incorporation.

In the next section we give the list of principles of agile development philosophy with brief outline.

7.4 Principle of Agile Development

There were twelve basic principles of agile software development as highlighted in the Agile Manifesto (132). The detail discussions of these principles are beyond the scope of this work. But for the purpose of justification and comparison, we combine and highlight these principles from Agile Manifesto (132) and as extracted from Conboy's (147) definition here in brief:

1. **Customer Satisfaction :** The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. **Incorporation of Rapid System Change :** Agile methodology welcomes changing requirements even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Frequent Working SW Delivery :** Deliver working software frequently- from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Continuous Cooperation of Client and Developer:** Business people and developers must work together daily throughout the project.
5. **Motivated Trusted Individuals :** Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. **Continuous Improvement-Arrangement of Face-to-Face Conversation:** The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. **Progress Measurement :** Working software is the primary measure of progress.
8. **Sustainable Development :** Agile processes promote sustainable development. The stakeholders, developers, and users should be able to maintain a constant pace indefinitely.
9. **Attention to Technical Excellence :** Continuous attention to technical excellence and good design enhance agility.
10. **Simplicity :** If the design and implementation are simple, testing is easier and more effective.
11. **Self-organizing Teams :** The best architectures, requirements, and designs emerge from self-organizing teams.
12. **Internal Assessment for Knowledge Enhancement :** At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

13. **Quality Assurance :** The organization must ensure the quality the system developed following the standard quality improvement practices.

14. **Economic Development :** Economic development should be achieved through optimum utilization of the resources through lean system development (147).

Some authors (149, 150) have done studies about scaling agile methods in regulated environments. But, practicing agile in regulated environments will imply many constraints to the agile process that may be either against the philosophy of agile or it could be equivalent to a traditional development approach. As pointed out by Fitzgerald et al (149), "Some of the essential characteristics of agile approaches appear to be incompatible with the constraints imposed by regulated environments". They further mentioned in the same paper that agile software development methods are faced with some fundamental challenges in regulated environments as a core characteristic of regulated environments is the necessity to comply with formal standards, regulations, directives and guidance. Further, as mentioned by Turk et al (151) that agile methods and regulated environments are often seen as fundamentally incompatible. Thus it may lead the agile process to be unlike a traditional process that is not the objective of agile always. The objective should not be to go away from the agile philosophy rather to be adhered to the Agile while achieving the advantages of disciplined approach. It is better not to tailor the agile methods to achieve the discipline necessary in regulated environments, but to optimally achieve the objectives of agile through some disciplined approach in regulated environment.
In the following section we review the BRIDGE model as for reference and then explain how to achieve agility through this model.

7.5 Introduction to BRIDGE Process Model

The BRIDGE process model was introduced and presented by Ardhendu (62) in IEEE IACC, 2009. Although, detailed discussion of the BRIDGE model is beyond the scope of this chapter, but we give the schematic diagram in Figure 6.2 of the model for the sophisticated readers just for reference with its primary properties.

The Primary Properties of BRIDGE Model
The primary properties of BRIDGE model are enlisted below (62):

1. It involves the client over the entire development life cycle activities.

2. It keeps continuous communication among the development team, project management team and client.
3. It enforces explicit verification of individual phases.
4. Supports Components Based Software Development (CBSD).
5. It enforces on standard coding practices.
6. It considers configuration management as a separate activity.
7. It forces to specify all the phase deliverables.
8. Separate software architecture design phase.
9. Separate system deployment phase.
10. Separate on-site system testing phase.
11. It explicitly instructs to validate the system.

7.6 Agile Development with BRIDGE Process Model

For detailed discussion about the BRIDGE process model and its primary properties introduced by Ardhendu (62), one may refer to Chapter 6 of this thesis. One may refer to Figure 6.2 for the schematic diagram of the BRIDGE process model.
In this section we are going to explore how the principles of agile can be achieved through BRIDGE process model. We consider all the principles of agile and discuss in brief how these are supported and can be achieved through BRIDGE model.

- **Achieving Customer Satisfaction :** The agile principles proposed customer collaboration for increasing customer satisfaction. In BRIDGE model this is achieved through continuous client interaction i.e. the left base pillar of the model.
- **Accommodation of Requirement Change :** The initial system requirements may not be mature enough at the very inception of the project. As the developers and client understand the system more and more over the development process, the requirements get refined, even may get modified over the time. As the client is engaged over the entire process, as soon as new requirement is discovered, it is accommodated within the same design or necessary modification is made in the

design and its subsequent phases by means of phase iteration. The iterative nature of the BRIDGE process model with the focus on component based software development facilitates accommodation of requirement changes easily in the system. Promoting system architecture design and component assembly based system development methodology (115), it is comparatively easy following BRIDGE process model to discard, add or modify system requirements.

- **Frequent Working SW Delivery :** As BRIDGE is an iterative process, hence we may start with the initial system requirements and design the initial working software with limited capabilities and deliver to the client at earliest. As the time moves, more and more requirements can be accommodated and delivered to the client in the subsequent software versions and variants.

- **Continuous Cooperation of Client and Developer :** This is one of the best features of the BRIDGE model. The client remains as an active member of the development process from the very inception till the end of the entire process. Hence, unlike agile in this model also there exist a close continuous cooperation between the client and developers.

- **Motivated Trusted Individuals :** Alike the client, the management unit also remain as a continuous part over the BRIDGE process model that enables the management to monitor the individual developers. Often, if needed the management may keep on motivating the developers to give their best and may take necessary actions to make them enough trust worthy for the organization. As a result, over the time the developers become more motivated trusted individuals for the organization. The management may investigate regularly about the need and working environment that can be allocated optimally so that the job can be done within time and budget while maintaining the quality.

- **Continuous Improvement-Arrangement of face-to-face conversation :** We know face-to-face conversation is the best way to convey and share information. Being an integral part of the process the management may arrange face-to-face conversation among the developers and even with the client for continuous improvement. As in BRIDGE process model management team is associated with the development over the development period, the management team may organize such events to promote continuous improvement easily.

- **Progress Measurement :** In general, before starting any phase it must satisfy some phase entry criterion. As each phase of the BRIDGE model has to go through a strict verification activity before starting the immediate next phase, hence all phase has to satisfy the phase exit criteria too. The phase entry and phase exit criterion are nothing but some intermediate milestones in addition to some other desirable constraints. These milestones may be even partial working software. Through these intermediate milestones, both the development and management team may assess and measure the progress periodically. Following the way, in BRIDGE model we may measure the progress of the project at regular interval.

- **Sustainable Development :** Continually evolving, growing, and changing is a natural phenomena of any organism- none of the other organisms can survive without it. The same thing can be applied to software also, just for analogy. Very few software is written once, installed, and then never changed over the course of its lifetime. As per Lehman's first law (126) regarding software, a software product must change continually or become progressively less useful. New requirement will get discovered over time, some old requirements may need to be modified or discarded for the products survivals and its enhancements. Hence, the system has to be designed and developed keeping the view of anticipating the future changes in mind. Following such type of development is what called sustainable development. But, unfortunately it is a rare practice as it may increase the product cost and other development burdens. Maintaining and enhancing software to cope with newly discovered problems or new requirements can take more time than the initial development of the software. Sustainable development requires a singular focus on a form of technical excellence that regularly provides useful software to customers while keeping the cost of change minimal. Under the umbrella of effective management, the project stakeholders can maintain consistent work pace and speed promoting sustainable development following the BRIDGE model as all of the stake holders works together in this model.

- **Attention to Technical Excellence :** In addition to design phase, the BRIDGE model has a specific software architectural design phase. Further, being the customer an integral part of the development process, technical excellence can be monitored by both the development and project management team. Continuous attention to architectural design, low level design and technical excellence of BRIDGE model enhances the agility.

- **Simplicity :** The notion of simple is very subjective and giving practical guidelines what is simple and how to accomplish that is impossible. At the beginning, requirements seem to be quite complex, but after the analysis and as the project development progresses they becomes clear. The developers should only implement features that have been agreed upon with the customers, nothing more. The art of maximizing the amount of work not done–is essential. Cockburn and Glass (152) warn that simplicity should not mean neglecting design by starting programming as soon as possible. This principle most strongly supports the design of high quality architecture and implementation. This principle further acknowledges that in programming it is more difficult to make simple design than cumbersome solutions. Following a disciplined and systematic approach makes any process simple. The system requirements should be simple and must specify the scope of the project very specific and clear. If the design and implementation are simple, testing becomes easier and effective. As in BRIDGE process model all the phases are distinctly well defined, it promotes simple design and implementation of the system that makes the other subsequent activities easier, simple and more effective.

- **Self-organizing Teams :** By self-organizing team we mean that the team members share a common goal and belief that their work is interdependent and collaboration is the best way to accomplish their goal. Rather than having a manager with responsibility for planning, managing and controlling the work, the team members share increasing responsibility for managing their own work and also share responsibility for problem solving and continuous improvement of their work processes. Hence, the empowered team members' reduce their dependency on top management as they accept accountability. The team structure places ownership and control close to the core of the work. The primary role of the project management is to build individual stakeholder a dedicated responsibility centre which is easy to establish through BRIDGE process model. By developing individual responsibility centers, the team may become the self-organizing team.

  Advantages of Self-Organizing :

  1. People in a self-organized team are able to make decisions themselves and accordingly adapt to changing situations.
  2. Self organized teams do a much better job of utilizing the talents of the team because more minds are involved in any activity.
  3. Self organized teams have much more communication between team members.

4. The best way to learn is to have actual responsibility and opportunities to do new things.
5. A self organized team is collectively aware of the upcoming work and much better able to bootstrap themselves with new work when they complete their existing task.
6. Self organized teams spread knowledge around much better and make decisions together. That makes each team member more effective because they have much more background on the "why" of the coding assignments.
7. A command and control team member often lacks an understanding of why a decision was made because they weren't involved with that decision. This may hamper their ability to follow a design or approach which restricts productivity.

As in BRIDGE model, the project management has consistent involvement with the development team, so if needed then the top management can facilitate the development team at any moment to be self-organized.

- **Internal Assessment for Knowledge Enhancement :** In association to project management the individual stakeholders may carryout internal assessment at regular intervals. Through the internal assessment result, the progress may be measured too. Further from the internal assessment the team may identify the various bottlenecks and upon rectifying and adjusting become more effective and plan the future activities efficiently. Being project management team with the development team in the BRIDGE process model, internal assessment becomes much easier.

- **Quality Assurance :** In BRIDGE model quality is ensures through implementation of multi level quality improvement methods through phase-wise verification, unit level, integration, and system testing, and system validation. Further, during maintenance, discovered errors if any may be rectified. Additionally, being management team working together with the development team, the entire process and individuals remain under proper control and monitoring of the management teams. Overall, the integrated project development environment of this process model ensures the system quality to achieved and improved.

- **Economic Development :** Economic development is achieved through optimal utilization of the resources and restricting misuse of resources. The optimal resource management is done through management authority with individual's care

and concern. In BRIDGE, all the stakeholders works together with better coordination and concern ensuring economic system development.

7.7 Conclusion

Despite of the criticism towards traditional software development process, the goodness of agile process is questionable. Agile may not be the good choice for some projects always. Often, traditional process model proves to the better than agile for some types of projects. Hence, our objective should not be to criticize the traditional process models over agile, rather we must carry out research to accommodate the good attributes of agile process in traditional process models. In this chapter we have shown that the philosophy of agile may also be achieved through the BRIDGE process model which follows the principle of traditional software development too. Hence, we recommend BRIDGE process model to be practiced by industries for modern software development projects.