Literature Review
A Case Study: Refining the SDLC Method, Improving the Quality and Accelerating the Software Development at BRI
By Irfan Syukur

1. Introduction

PT. Bank Rakyat Indonesia (BRI) is the most profitable bank and the largest banks in Indonesia that specializes in Micro, Small and Medium Enterprises (MSME). The official website of BRI [1] reported, as of December 2014, BRI serves its customers through more than 10,000 outlets (divisions, main branch, sub branch, micro units), 150,000 electronic outlets (ATMs, CDMs, EDCs) and 120,000 professional employees that spreads all over Indonesia. Most of its customers; the recent data showed that BRIs customers reach approximately 55.7 million; are lower-middle income people such as land workers, fishermen, factory workers, small businesses and government employees. As one of the state-owned banks and a bank that has the widest network in Indonesia, BRI is always chosen by the government as an agency to distribute social aid funds and enacting social programs of the government in economic empowerment. To provide the best service for citizen that spread all over Indonesia, no doubt, that technology—infrastructure and systems— plays a very important role. With powerful and good technology provided by BRI, it ensures that all goals, as a business company and a government agent, will be achieved.

The Information Technology Division as one entity of BRI's organizational structure is responsible for providing, executing, monitoring and maintaining the whole technology required for all business and operation activities. In connection with the great and the high complexity of

BRI, it requires a tremendous effort to provide technology that cost-efficient, effective, and of high quality. A methodology or procedure is needed to help and ensure the stakeholder to design, control and monitor every project of technology being executed to achieve predetermined objectives. The methodology or procedure is known in IT industry as a System Development Life Cycle (SDLC). The SDLC method /model is basically a project management tool that is used to plan, execute, and control systems development projects and it is also important to understand that these are just models; they do not represent the total system [6, pp. 2]. Therefore, the purpose of this article is to review literature related to the SDLC methods. While no document could cover all the varieties of SDLC methodologies, this article tries to categorize the predominant practices in use today, so as give a practical understanding of various forms of SDLC in use and provide an alternative approach in selecting and implementing SDLC in a particular project. As a case study, this article focuses on BRI's system software development projects.

2. The Important of SDLC Methods

As defined by [6, pp. 2], System development life cycle (SDLC) is an approach to develop an information system or software product that is characterized by a linear sequence of steps that progress from start to finish without revisiting any previous step. The SDLC methodologies all focus on the common goal of defining steps or phases and processes from the beginning of a project through its successful completion and beyond. The SDLC aims to produce high quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates. As described by [2, pp. 1], the SDLC provides a structure so that team members and all project stakeholders understand the current state of the project. Additionally, it supports

visibility and predictability while enabling project teams to make specific choices that achieve the business goals and constraints.

Many SDLC models have been proposed so far and each of them have some advantages as well as some disadvantages as shown in Table 1 and Table 2. A few important and commonly used SDLC methods are waterfall (it was and still is, the foundation for all methodologies), incremental, iterative / evolutionary, V-shaped, rapid application (RAD) and spiral. Actually there are only three basic methods—waterfall, incremental and iterative model. The other methods are only the derivative, enhancement or mixture from those basics. From [2][4][5] and [6], it can be concluded that most SDLC's could be divided into several phases or steps, which are : planning and requirement analysis, design, implementation, testing and maintenance and operational.

Many software developments have evolved over the years and It is important to understand that one of SDLC method is not necessarily suitable for use by all projects. Each of the available methods is best suited to specific kinds of projects [3, pp. 1]. Methods, as explained by [6, pp. 2], reflect the structure of the organization, its management style, the relative importance it attaches to quality, timeliness, cost and benefit, its experience and its general ability levels, and many other factors. There should not be any standard method because companies are unique. The project team must select a suitable SDLC method for the particular project and then adhere to it. [8, pp.1] claimed that without using of a particular method, the development of a software product would not be in a systematic and disciplined manner; it would lead to chaos and project failure.

3. Selecting an SDLC Method

From a study [6, pp. 1], improving the quality and reducing the cost of products are fundamental goals of software process method selection. While selecting the right SDLC methodology is challenging, the challenge is not insurmountable. With a clear understanding of the business and a framework for guidance, selecting a fitting SDLC can be readily achieved. [2, pp. 264] believed that a proper methodology in a maturing environment will enable the business to build software that assists the business in realizing its value proposition. The method being selected, must give control over complexity (feature, content and architecture), sizing (point solution, departmental or enterprise), service level (ad hoc, reliable and mission critical) and delivery (buy vs. build decisions) [4, pp. 11). These are not simple challenges and the selection of a method to accommodate these demands is critical to the success of the projects. The main issues in selecting an SDLC method is to take the position that organizations can benefit from following a formal process for identifying and selecting projects [6, pp. 1] and only few organizations know what criteria to use in selecting a method to add value to the organizations [4, pp. 11]. Regarding to those issues, this article presents two different approaches in selecting an SDLC method.

3.1. D3 Cube (Decision Cube)

With value propositions linked so tightly to SDLC method, the organizations have to select and utilize the method that aligns with their predetermined objectives. [4, pp. 13] recommended The D3 Cube (Decision Cube)—as shown in Figure 2— as a framework that provides a common approach to selecting an SDLC method. As shown in Figure 1, there are three criteria—functionality, budget and time. Each of which map to a specific element of business value and, when combined, form the basis for selecting an SDLC method to optimize value to

the business. To use the Decision Cube simply plot the project or organizational project style along the appropriate axis for each of the three criteria. The resultant quadrant is the recommended method. Unfortunately, there will always be anomalies on those projects. When a selected method is considered less meet predetermined criteria, the organizations can shift the recommended method into the nearest quadrant of the recommendation [4, pp. 17]. The point is, for any change in the method, it allowed moving only one quadrant in any direction, as shown in Figure 3.

3.2. Project Characteristic

From time to time, many software process models have been developed in order to maintain reliability and quality of software. To achieve those objectives, another approach in selecting an SDLC method was suggested by [6], which is "Project characteristic". This approach divides categories into three project characteristics—project team, user community, and project type and risk. Each of them is measured in 0-10 rating. Further, comparison tables are designed on three project Characteristic categories as shown in Table 3, 4 and 5. Moreover, based on observation, comparison and experience that described in those tables, suggested methods are prepared for each characteristic category as shown in Table 6, 7 and 8.

4. The Implementation of SDLC Method at BRI

As declared by [9, pp. 2], to support the development of information technology at BRI, it needs a guideline that could be used as reference upon the development of IT systems. The development of BRIs systems uses an SDLC methodology. In the methodology, the stage of development of IT systems is divided into initiation, planning, requirements definition, design,

programming, testing, implementation, post-implementation review, maintenance, and disposal if the system will not be used again. The previous SDLC guidelines had been implemented since January 2006. Regarding to the change of the organizational structure of the IT Division and the development of technology and information systems used in BRI, and the new regulations of the Central Bank, the IT system development guidelines need to be adjusted in order to remain applicable and auditable. Although it is not precisely defined, the new SDLC method adopts the waterfall method with adding some iteration at certain stages.

In line with the business and operational needs, it requires many new and the current systems that have to be developed or enhanced. However, because of the limitation in human resources, the tight of time schedule, the lack of management and the lack well communication between the users (business and operation divisions) and IT Division, many of the development projects could not meet the criteria expected, in term of the time schedule and the system quality. With those facts, it appears that the waterfall method being implemented by BRI is insufficient and ineffective to cope with the trends. Considering to those issues, this article's objective is to provide an alternative solution that could be used as a reference based on the literature reviews.

5. Waterfall and Agile Methods

The main reason why BRI chose the waterfall as its SDLC method was that as a bank, public and state company, "auditable" and well documented is an obligation. BRI has to responsible and report on all its activities. Waterfall method provides those benefits, as shown in Table 1. The classical waterfall model is an idealistic one since it assumes that no development error is ever committed by all the team project members and the users during any of the SDLC phases.

However, in practical development environments, it is impossible to avoid such circumstances. Because of its inflexibility [Table 1], less adaptability [8, pp. 9-10][6, pp. 6] and misassumptions related to the method [7], this 'waterfall' could not follow the development of the ongoing trend at BRI.

Base on these issues, another SDLC method is proposed, which are the agile methods. Agile Methods are a reaction to traditional ways of developing software and acknowledge the "need for an alternative to documentation driven, heavyweight software development processes" [11]. Agile development is not a methodology in itself. It is an umbrella term that describes several agile methodologies—for example : Scrum, XP, Crystal, FDD, and DSDM [10].

As illustrated by [12], Agile methods are based on adaptive software development methods, while traditional SDLC models (waterfall model, for example) are based on a predictive approach. In traditional SDLC models, teams work with a detailed plan and have a full list of characteristics and tasks that must be completed in the next few months or the entire life cycle of the product. Predictive methods completely depend on the requirement analysis and careful planning at the beginning of the cycle. Any change that is to be included will go through a strict change control management and prioritization. The agile model uses an adaptive approach where there is no detailed planning and only clear future tasks are those related to the characteristics that must be developed. The team adapts to dynamic changes in the product requirements. The product is frequently tested, minimizing the risk of major faults in the future. Interaction with the clients is the strong point of agile methodology and open communication and minimal documentation are typical characteristics of the agile development environment. Or in other

words, agile methods are adaptive rather than predictive and people-oriented rather than process-oriented.

6. Conclusion

The waterfall method that has been used by BRI, less relevant to the ongoing trend and has to be exchanged or enhanced. As recommended by the Decision Cube approach, the waterfall only could move to Incremental or Spiral method. However, according to Table 1 and Table 2, these methods also have weaknesses that still insufficient to cope with the trends. The Agile methods come up and provide an alternative solution; the advantages and the weakness of these methods explained in Table 9. Unfortunately, considering to the scale and the organizational complexity of BRI, the movement of the SDLC method from the waterfall to the agile would give a tremendous cost, efforts and implementation time to BRI. As an alternative, the waterfall and the agile methods could be combined, optimizing the advantages of each of the methods and minimizing the weaknesses of each of them. The final objectives are to get an method that auditable, well documented, flexible and adaptable.

As a literature review, this article presents and offers an alternative approach related to the current issues faced by BRI on its software development projects. However, it still needs further in-depth research to ensure the right method for BRI.

References:

[1] "PT. BRI Corporate Profile", [online] 2015, http://phx.corporate-ir.net/phoenix.zhtml?c=148820&p=irol-homeProfile, (Accessed: 3 April 2015)

[2] Ms Namrata Jain and Anurag Jain, "Software Development Life Cycle: A Detailed Study", *International Journal of Advanced Research in Computer Science*, vol. 2, no 3, pp. 261-264, May 2011

[3] Center for Medicare & Medicaid Services. (2008, Feb. 27), *Selecting a Development Approach*. [Online]. Available: http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

[4] Alan E.Dillman. (2008), Understanding and Selecting Systems Development Life Cycle (SDLC) Methodologies. [Online]. Available: http://www.menyaltd.com/Portals/7/Understanding%20and%20Selecting%20SDLC%20Methodologies.pdf

[5] Neha Budhija and Satinder Pal Ahuja, "Study of Software Process Model Selection", *International Journal of Advanced Research in Computer Science*, vol. 2, no 6, pp. 279-282, Nov./Dec. 2011.

[6] Transseed Group. (2015), System Development Life Cycle. [Online]. Available: http://www.transseed.com/downloads/TRS%20-%20Systems%20Devlopment%20Life%20Cycle.pdf

[7] InfoQ.com. (2015). [Online]. Available: http://www.infoq.com/resource/articles/scaling-software-agility/en/resources/ch02.pdf

[8] NPTEL. (2015). [Online]. Available: http://www.nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Soft%20Engg/pdf/m02L03.pdf

[9] Nokep: S. 217-DIR/TSI/10/2011. (2011, Oct). *Prosedur Siklus Pengembangan Sistem TI PT. Bank Rakyat Indonesia (Persero), Tbk.*

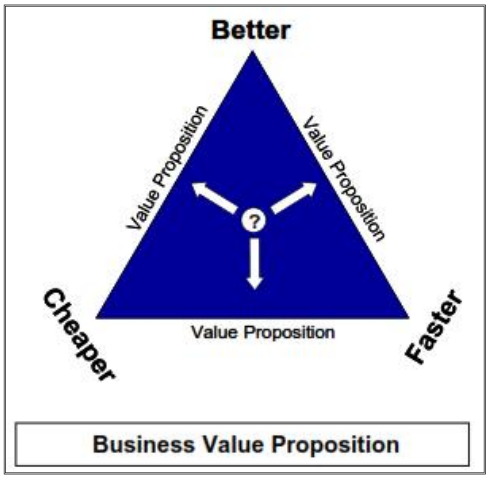
[10] Monjurul Habib. (2013, Dec 30). *Agile software development methodologies and how to apply them* [Online]. Available: http://www.codeproject.com/Articles/604417/Agile-software-development-methodologies-and-how-t

[11] Beck K., Cockburn A., Jeffries R., Highsmith J., "Agile manifesto", http://www.agilemanifesto.org, 2001, 12-4-2002.

[12] Marian STOICA, Marinela MIRCEA and Bogdan GHILIC-MICU, "Software Development: Agile vs. Traditional", *Informatica Economica*, vol. 17, no 4/2013, Apr. 2013.

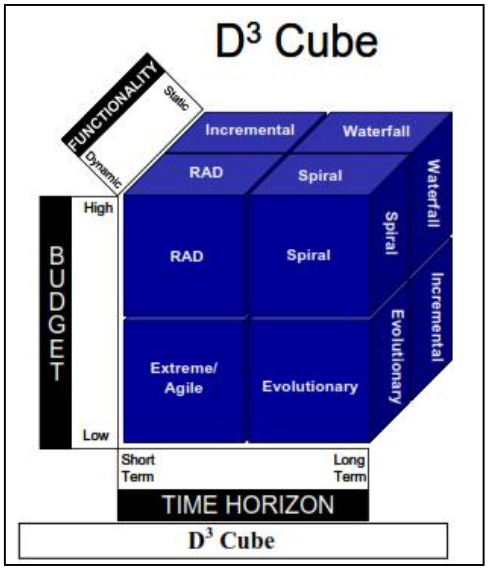
[13] Sheetal Sharma, Darothi Sarkar, and Divya Gupta, " Agile Processes and Methodologies: A Conceptual Study", *International Journal on Computer Science and Engineering (IJCSE)*, vol. 4, no 5, May. 2012.

Figure 1. Business Value Proposition [5].



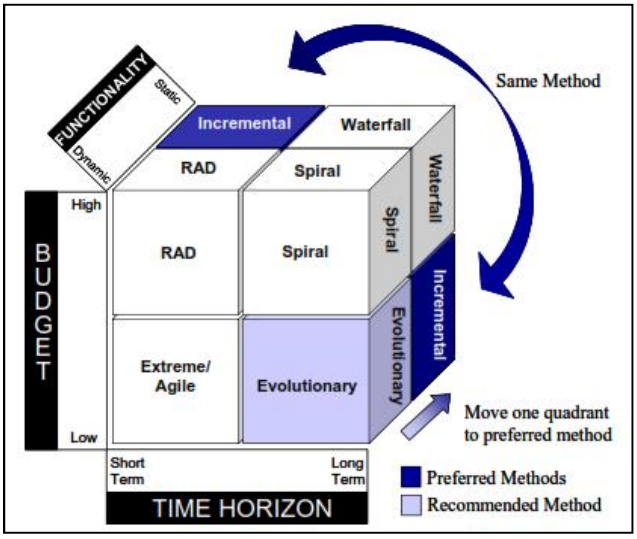
There are the goals to be achieved by all business projects [5].

Figure 2. D^3 Cube



These are three selection criteria—functionality, budget and time— each of which map to a specific element of business value and, when combined, form the basis for selecting an SDLC methodology to optimize value to the business [5].

Figure 3.



Any proposed changes in methodology, it is advisable to shift only one quadrant from the current methodology [5].

Table 1. The strengths and weaknesses for the most important and popular software development methods (SDLC) [3].

Methods	Strengths	Weaknesses
Waterfall	• Ideal for supporting less experienced project teams and project managers, or project teams whose composition fluctuates. • The orderly sequence of development steps and strict controls for ensuring the adequacy of documentation and design reviews helps ensure the quality, reliability, and maintainability of the developed software. • Progress of system development is measurable. • Conserves resources.	• Inflexible, slow, costly and cumbersome due to significant structure and tight controls. • Project progresses forward, with only slight movement backward. • Little room for use of iteration, which can reduce manageability if used. • Depends upon early identification and specification of requirements, yet users may not be able to clearly define what they need early in the project. • Requirements inconsistencies, missing system components, and unexpected development needs are often discovered during design and coding. • Problems are often not discovered until system testing. • System performance cannot be tested until the system is almost fully coded, and under-capacity may be difficult to correct. • Difficult to respond to changes. Changes that occur later in the life cycle are more costly and are thus discouraged. • Produces excessive documentation and keeping it updated as the project progresses is time-consuming. • Written specifications are often difficult for users to read and thoroughly appreciate. • Promotes the gap between users and

		developers with clear division of responsibility.
Iterative	• "Addresses the inability of many users to specify their information needs, and the difficulty of systems analysts to understand the user's environment, by providing the user with a tentative system for experimental purposes at the earliest possible time." • "Can be used to realistically model important aspects of system during each phase of the traditional life cycle." • Improves both user participation in system development and communication among project stakeholders. • Especially useful for resolving unclear objectives; developing and validating user requirements; experimenting with or comparing various design solutions; or investigating both performance and the human interface. • Potential exists for exploiting knowledge gained in an early iteration as later iterations are developed. • Helps to easily identify confusing or difficult functions and missing functionality. • May generate specifications for a production application. • Encourages innovation and flexible designs. • Provides quick implementation of an incomplete, but functional, application.	• Approval process and control is not strict. • Incomplete or inadequate problem analysis may occur whereby only the most obvious and superficial needs will be addressed, resulting in current inefficient practices being easily built into the new system. • Requirements may frequently change significantly. • Identification of non-functional elements is difficult to document. • Designers may prototype too quickly, without sufficient up-front user needs analysis, resulting in an inflexible design with narrow focus that limits future system potential. • Designers may neglect documentation, resulting in insufficient justification for the final product and inadequate records for the future. • Can lead to poorly designed systems. Unskilled designers may substitute prototyping for sound design, which can lead to a "quick and dirty system" without global consideration of the integration of all other components. While initial software development is often built to be a "throwaway", attempting to retroactively produce a solid system design can sometimes be problematic. • Can lead to false expectations, where the customer mistakenly believes that the system is "finished" when in fact it is not; the system looks good and has adequate user interfaces, but is not truly functional. • Iterations add to project budgets and schedules, thus the added costs must be weighed against the potential benefits. Very small projects may not be able to justify the added time and money, while only the high-risk portions of very large, complex projects may gain benefit from prototyping. • Prototype may not have sufficient checks and balances incorporated.
Incremental	• Potential exists for exploiting knowledge gained in an early increment as later increments are developed. • Moderate control is maintained over the life of the project through the use of written documentation and the formal review and approval/signoff by the user and information technology management at designated major milestones.	• When utilizing a series of mini-Waterfalls for a small part of the system before moving on to the next increment, there is usually a lack of overall consideration of the business problem and technical requirements for the overall system. • Since some modules will be completed much earlier than others, well-defined interfaces are required.

	• Stakeholders can be given concrete evidence of project status throughout the life cycle. • Helps to mitigate integration and architectural risks earlier in the project. • Allows delivery of a series of implementations that are gradually more complete and can go into production more quickly as incremental releases. • Gradual implementation provides the ability to monitor the effect of incremental changes, isolate issues and make adjustments before the organization is negatively impacted.	• Difficult problems tend to be pushed to the future to demonstrate early success to management.
V-model	• Simple and easy to use. • Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model. • Proactive defect tracking – that is defects are found at early stage. • Avoids the downward flow of the defects. • Works well for small projects where requirements are easily understood.	• Very rigid and least flexible. • Software is developed during the implementation phase, so no early prototypes of the software are produced. • If any changes happen in midway, then the test documents along with requirement documents has to be updated.
Spiral	• Enhances risk avoidance. • Useful in helping to select the best methodology to follow for development of a given software iteration, based on project risk. • Can incorporate Waterfall, Prototyping, and Incremental methodologies as special cases framework, and provide guidance as to which combination of these models best fits a given software iteration, based upon the type of project risk. For example, a project with low risk of not meeting user requirements, but high risk of missing budget or schedule targets would essentially follow a linear Waterfall approach for a given software iteration. Conversely, if the risk factors were reversed, the Spiral methodology could yield an iterative Prototyping approach.	• Challenging to determine the exact composition of development methodologies to use for each iteration around the Spiral. • Highly customized to each project, and thus is quite complex, limiting reusability. • A skilled and experienced project manager is required to determine how to apply it to any given project. • There are no established controls for moving from one cycle to another cycle. Without controls, each cycle may generate more work for the next cycle. • There are no firm deadlines. Cycles continue with no clear termination condition, so there is an inherent risk of not meeting budget or schedule. • Possibility exists that project ends up implemented following a Waterfall framework.
RAD	• The operational version of an application is available much earlier than with Waterfall, Incremental, or Spiral frameworks. • Because RAD produces systems more quickly and to a business focus, this approach tends to produce systems at a lower cost. • Engenders a greater level of commitment from stakeholders, both business and technical, than Waterfall, Incremental, or Spiral frameworks. Users are seen as gaining more of a sense of ownership of a system,	• More speed and lower cost may lead to lower overall system quality. • Danger of misalignment of developed system with the business due to missing information. • Project may end up with more requirements than needed (gold-plating). • Potential for feature creep where more and more features are added to the system over the course of development. • Potential for inconsistent designs within and

	while developers are seen as gaining more satisfaction from producing successful systems quickly. • Concentrates on essential system elements from user viewpoint. • Provides the ability to rapidly change system design as demanded by users. • Produces a tighter fit between user requirements and system specifications. • Generally produces a dramatic savings in time, money, and human effort.	across systems. • Potential for violation of programming standards related to inconsistent naming convention and inconsistent documentation. • Difficulty with module reuse for future systems. • Potential for designed system to lack scalability. • Potential for lack of attention to later system administration needs built into system. • High cost of commitment on the part of key user personnel. • Formal reviews and audits are more difficult to implement than for a complete system. • Tendency for difficult problems to be pushed to the future to demonstrate early success to management. • Since some modules will be completed much earlier than others, well-defined interfaces are required.

Table 2. The advantages and disadvantages of the most and popular SDLC methods [4].

Methodology & Criteria	Advantages	Disadvantages
Waterfall Budget : High Time : Long Functionality : Static	• Cleary define phases • Assures delivery of initial requirements • Well documented process and results	• Lack of measurable progress with phases • Cannot accommodate changing requirements • Resistant to time and/or budget compression
Incremental Budget : High Time : Long Functionality : Static Or Budget : High Time : Long Functionality : Static	• Early and periodic results • Measurable progress • Supports parallel developments processes	• Demands increased management attention • Can increase resource requirements • No support for changing requirements
Iterative Budget : High Time : Long Functionality : Static	• Supports changing requirements • Minimize time to Initial Operating Capability (IOC), a point in time during the Production & Deployment (PD) Phase where a system can meet the minimum operational (Threshold and Objective) capabilities for a user's stated need. • Achieves economist of scale for enhancements	• Increases management complexity (e.g. number of business units, functions, geographies and layer of managements) • IOC is not complete • Risk of not knowing when to end the project
Spiral Budget : High Time : Long	• Supports changing requirements • Allows for extensive use of prototype, used to allow the users evaluate	• Increased management complexity • Defer production capability to end of the SDLC

Functionality : Static	developer proposals and try them out before implementation • More accurate captures requirements	• Risk of not knowing when to end the project
RAD Budget : High Time : Long Functionality : Static	• Minimizes time to delivery • Accommodates changing requirements • Measures progress	• Increased management complexity • Drives costs forwards in the SDLC • Can increase resource requirements
Agile Budget : High Time : Long Functionality : Static Or Budget : High Time : Long Functionality : Static	• Rapid demonstrable functionality • Minimal resource requirements • Supports fixed of changing requirements	• Not conducive to handling complex dependencies • Creates QA risks • Increased risk of sustainability, maintainability and extensibility

Table 3. Comparison based on project team [6].

	Waterfall	**Spiral**	**RAD**	**Incremental**
New to problem domain	1	9	1	3
New to technology domain	8	9	1	8
New to tools to be used	7	8	1	2
Any training available	2	1	8	9
Comfortable with structure	8	1	2	9
Closely track by manager	8	9	2	9

Table 4. Comparison Based On User Community [6].

	Waterfall	**Spiral**	**RAD**	**Incremental**
Availability of user representative restricted or limited	9	2	2	7
Expert in problem domain	2	9	2	8
Want to track the project process	7	8	1	2
Want to involve in SDLC	2	9	2	8

Table 5. Comparison Based On Project Type and Risk [6].

	Waterfall	**Spiral**	**RAD**	**Incremental**
System integration project	2	8	7	9
Enhancement to an existing project	2	2	9	8
High reliability is must	7	9	8	2

Table 6. Suggested model base on team property [6].

S.N	**Project Team Member**	**Suggested Model**
1.	New to problem domain	Spiral
2.	New to technology domain	Spiral
3.	New to tools to be used	Spiral
4.	Any training available	Incremental
5.	Comfortable with structure	Waterfall

6.	Closely track by manager	Spiral

Table 7. Suggested model based on user community [6].

S.N	User Communicate	Suggested Model
1.	Availability of user representative restricted or limited	Waterfall
2.	User representative new to the system definition	Spiral
3.	User representative expert in problem domain	RAD
4.	User representative want involve in SDLC	RAD
5.	User representative want to track project progress	Spiral

Table 8. Suggested model base on project type and risk [6].

S.N	Project type and risk	Suggested Model
1.	Integration project	Incremental
2.	Enhancement to an existing System	RAD
3.	The funding for project stable	
4.	Project reliability must	Spiral

Table 9. The Advantages and Disadvantages of the Agile Methods [13]

The Advantages	The Disadvantages
Adaptive to the changing environment	Customer interaction is the key factor of developing successful software
Ensures customer satisfaction	Lack of documentation
Least documentation	Time consuming and wastage of resources because of constant change of requirements
Reduces risks of development	More helpful for management than developer