Third 21st CAF Conference at Harvard, in Boston, USA. September 2015, Vol. 6, Nr. 1
ISSN: 2330-1236

A Software Project Management Innovation (SPM) Methodology: A Novel Method for Agile Software Development

Fahad Abdulaziz Aleid
Royal Saudi Air Force
Communications and Information Technology
Riyadh, Saudi Arabia

Abstract

This paper seeks to define and describe a new and effective Agile Software Development Method, which comes under the auspices of the Software Development Life Cycle (SDLC). The method has been satisfactory experimented by the Royal Saudi Air Force (RSAF). This paper called the method *"Software Project Management (SPM)"* which gives the development team to control and manage software project resources more effectively by increasing team collaboration and productivity thus decreasing the amount of time needed to complete the project. The methodology, therefore, allows clients more time to produce a dedicated and well-integrated application by applying a refined systematic and structured process of continuous refinements within a defined period. This paper fills a gap and contributes knowledge towards Agile Software Development methodologies by providing a new novel method that has been tested, applied, and modified during managing RSAF's software projects.

Keywords: Software project management, SPM, Agile Software Development, SDLC.

Introduction

In the last decade, Software Development Life Cycles (SDLC) has undergone a major revolution in the Information Systems (IS) arena. Numerous software development methods have appeared in this field. Developmental approaches originated in the field of IS but the risk facing both in-house and outsourcing software teams is to develop applications that meet customer satisfaction within a given timeframe.

This paper describes some of the agile methods. Then discusses the new methodology, which is the topic of this research.

The study is presented in five sections. The following section, reviews some agile software development concepts, their values, principles and characteristics. The second section describes some example agile methods, such as, SCRUM, Unified Process (UP) and Feature-driven development (FDD). The third section covers Software Project Management Methodology (SPM), which is the central feature of this study. The final section summarizes the research by describing its contribution to knowledge, its implications and its scope for further work in the field.

Literature Review

An overview of Agile Software Development

The purpose of this section is to present an overview of Agile software development. To begin with, Agile software development (ASD) is a set of software development methodologies that is an alternative to the traditional waterfall approach. ASD focuses on finding a way of flexibility, close collaboration between the development team and business side, keeping code simple, shorten the development time-frame, respond to predictable and unpredicted change, do more tests on small releases of the system, and delivering each release as soon as it is ready. It consists of a group of software development methods that share the same characteristics.

Agile software development comprises different development methods but teams agree upon the need for collaboration, e.g. programmer teams and business experts in order to deliver a particular software project. It encourages group planning, evolution, many deliveries during the particular project, continuous improvement and flexibility to change (Agile Alliance, 2013).

Agile software development is grounded in techniques developed by James Martin (Martin, 1991) and James Kerr et al (Kerr & Hunter, 1993). In 2001, representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development and Pragmatic Programming met to consider lightweight development methods and produced an Agile Software Development Manifesto (Agile Alliance, 2013). At the manifesto's publication the authors said,

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions	*Over*	*Processes and tools*
Working software	*Over*	*Comprehensive documentation*
Customer collaboration	*Over*	*Contract negotiation*
Responding to change	*Over*	*Following a plan*

That is, while there is value in the items on the right, we value the items on the left more" (Agile Alliance, 2013).

Agile Software Development, therefore, considers self-organization, interactions and motivations are more important than process and tools. It also considers working software more important than presenting documents to clients in meetings. Thirdly, it values customer collaboration to fulfill better their requirements because during the initial stages of the project customers cannot always precisely define them. Therefore, customer collaboration/ involvement is very important. Finally, this type of development believes that software

projects should not stick to a fixed plan but be capable of making a quick response to change. Highsmith and Cockburn (2001, p. 122) said,

> *"What is new about agile methods is not the practices they use but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability."* (Highsmith & Cockburn, 2001)

There are 12 principles or rules in the Agile Manifesto as follows (Agile Alliance, 2013):

1. Customer satisfaction through the continuous delivery of useful software is important.
2. Changing requirements is welcomed and acceptable even if it comes late during development.
3. The software is delivered within a couple of weeks for each cycle.
4. Developers and clients cooperate and work together on a daily basis.
5. Projects are built around motivated and individuals who are trusted by customers. These individuals should facilitate customers' needs and prepare the appropriate software environment.
6. Face-to-face conversation is the best method to understand clients' requirements.
7. The best measurement of progress is the working software itself.
8. Promote sustainable development and all participants in the project should maintain a constant pace.
9. Continuous attention to deliver fast quality software with a high potential for user acceptance and at the same time it should possess the ability to adapt to changing business requirements.
10. Simplicity is important. This can be achieved by reducing the amount of work required to operate the software without loss to its functionality.
11. A self-organizing team is essential because it is the key to achieve best requirements, design and architecture.
12. The team should hold regular discussions about their practices and seek ways to become more effective.

Characteristics of Agile Software Development

There are nine characteristics of Agile Software Development as follows (Agile Alliance, 2013):

1. Modularity: It is breaking the process of development into different modules called activities.
2. Iterative: The development process comes in short cycles to achieve good results with respect to verifications and corrections.
3. Time-bound: For iterations there is a time limit, which makes the project easy to plan and develop, and lower the risks every four weeks.
4. Parsimony: Agile processes attempt to remove unnecessary activities to reduce risks and heighten success.
5. Adaptive: Agile process adapts to any new risks.
6. Incremental: Agile processes do not deliver the entire work at once but divide the system into increments. This helps to teams to work in parallel within different time scales.
7. Convergent: Agile processes apply all techniques to reach goals.
8. People-Oriented: Agile processes favor people over process and technology.
9. Collaborative: Agile processes encourage team members to communicate and collaborate with each other for quicker integration in large projects and to be able to work in parallel.

Reviewing some of Agile software development methods:

This section reviews some examples of the previous studies of different methods of the Agile Software Development such as SCRUM, Unified Process (UP) and Feature-driven development (FDD). A brief description about some of these methods follows.

SCRUM

Scrum is an agile software development method that manages software product development, which was developed in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka in the New Product Development Game (New Product Development Game, 1986). It is a flexible strategy that allow development teams to work as a unit in order to reach the software's goals. The process is described,

"Scrum is a management and control process that cuts through complexity to focus on building software that meets business needs." (Scrum.org, 2015).

In other words, Scrum is

"a simple framework for effective team collaboration on complex software projects" (Scrum.org, 2015).

The main principle of SCRUM is during a software project, the owner of the project has the ability to change requirements even in the middle. This cannot be achieved easily if traditional development methods, e.g. a sequential process, such as, the waterfall model are used. The following diagram presents SCRUM's process framework.

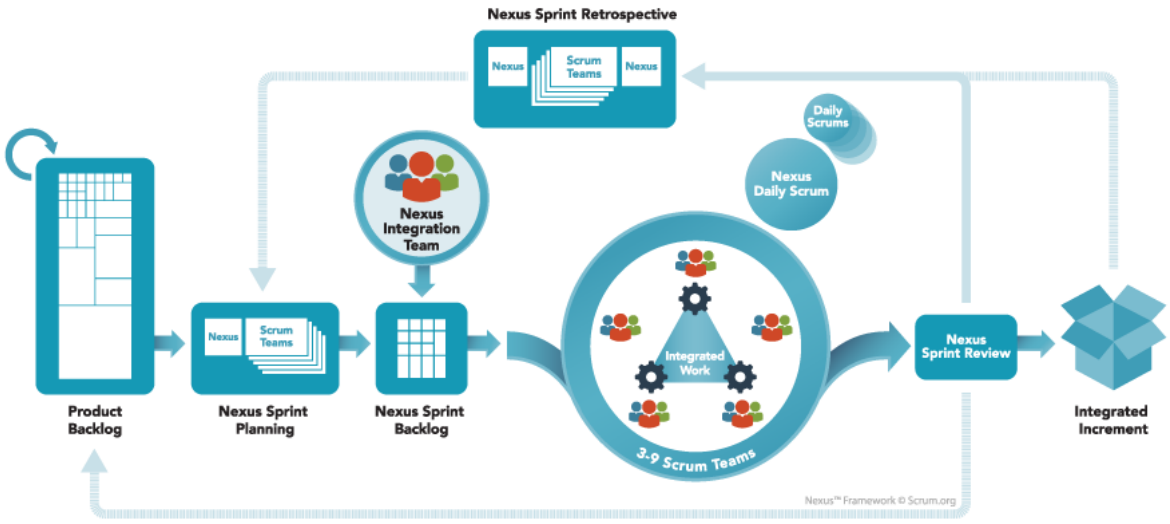

Figure 1: The SCRUM method lifecycle (Scrum.org, 2015).

Before describing the main roles in Scrum, this study presents and explains the terms *"Product Backlog"* and *"Sprint Backlog"*. To begin with, product backlog is an ordered requirement list for the product. The product owner has the main responsibility for it. It can be changed during the stages of the project. It lists the requirements, functions and features of the product. It has items like description, order, estimated time and value. The advantage here is that requirements can be changed while project is still in development. Product backlog refinement usually takes no more than 10% of the capacity of the development team. The more important items in the product backlog should be more detailed compared to those of lower importance.

On the other hand, sprint backlog is a set of items for each sprint that is selected from the product backlog. During sprint, the sprint backlog is updated after each Daily Scrum.

After describing the above terms, this study presents the three main roles in SCRUM. The first role is the product owner who is a person represents the customer (client) by writing the customers' needs (user's story) and ensuring that the development team delivers the correct

and accurate product to the business. He/she is responsible for ranking and prioritizing the requirements and adding them to the product backlog.

The second role is the development team in which the team is made between three to nine people. These people are a mixture of analysts, developers, designers, testers and writers of documents. They are responsible for developing and delivering the parts of the final product at the end of each sprint.

The third role is the scrum master who is responsible for eliminating risks and removing obstacles during the project. This helps the team to achieve their goals. The scrum master, however, is not a project manager but a person who facilitates teamwork and ensures that the Scrum process is used as originally proposed.

Sprint is the core key of Scrum. Each sprint last between 1-4 weeks. During each sprint, no one can make a change but the scope can be negotiated for clarifications. This helps the team to learn more about the issues under work. This approach helps to limit risk to one month of the costs.

Scrum proposes four events in order to adapt (Scrum.org, 2015). First is the sprint Planning event which plans what needs to be delivered during this sprint and how much time is required in order to achieve the work. The maximum time for this stage is eight hours in case the sprint is one-month long. Second event is the fifteen minutes daily scrum meeting which held between members of the development team to assess what has and needs to be achieved during the next workday. The rule of the scrum master is to ensure that the development team has attended the meeting within the fifteen minutes' period. Third event is the sprint review. This event has a maximum of four hours. This meeting takes place at the end of each sprint to assess what was achieved during the sprint and compare it to the Product Backlog. The product owner explains what has been and not been done in the Product Backlog. The development team presents their work and problems that arose during the sprint and how they propose to solve them. The forth event is the sprint retrospective. This event is a maximum of three hours long. The meeting is a good chance for the team to plan improvements for the next Sprint.

Unified Process (UP)

It is a simple and easy approach for developing business software using agile techniques (Scott W. Ambler + Associates, 2006). The Unified Process captures the nature of Agile in its four phases (Scott W. Ambler + Associates, 2006). The first phase is the inception which identifies the initial scope of the project and the system's potential architecture. Then the second one is the elaboration phase which verifies the architecture of the system. The third phase is the construction. This phase builds the software on a regular and incremental basis. The final one is the transition phase that validates and deploys the application into the production environment.

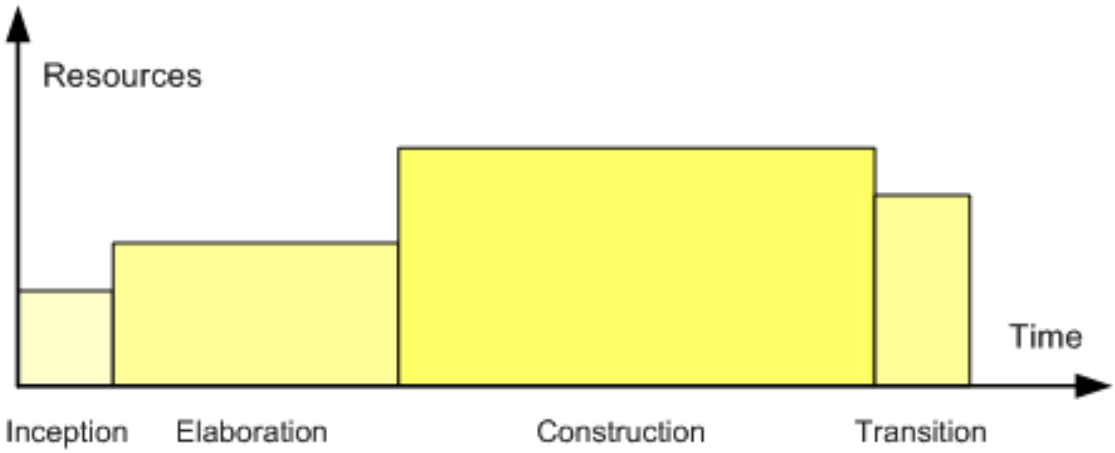

Figure 2: The four phases of the Unified Process. (Scott W. Ambler + Associates, 2006)

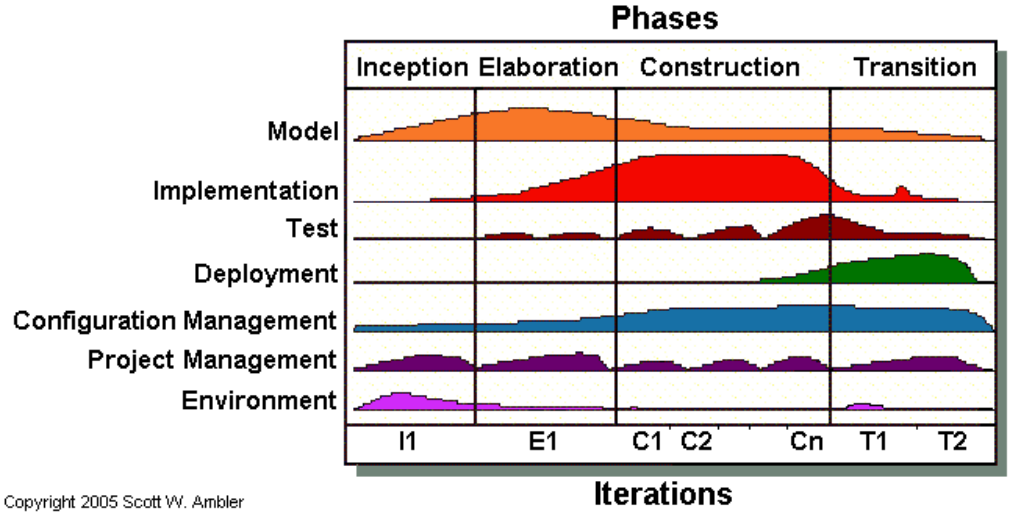

Figure 3: The Agile Unified Process (AUP) lifecycle (Scott W. Ambler + Associates, 2006).

As seen in the above figure, disciplines are performed in an iterative manner (Scott W. Ambler + Associates, 2006). To begin with, model aims to understand the business model of the organization and identifies the problem, and finds the best solution. Then, implementation which transforms the model into an executable programming code. Then, the test that evaluates the system for quality purposes. This includes finding bugs and ensuring that requirements are met. Next is deployment discipline which aims to deliver the system to end users. Then configuration management which manages access to the projects' artifacts. This includes versioning and managing changes to the system. Then, project management which directs project activities. For example, it manages risks, assigns tasks to people and tracks progress to make sure that it delivers on time and within budgetary limits. Then, the environment discipline that ensures that processes, standards, guidelines, software, and hardware are available to the team.

UP produces the system for release in different versions (v1, v2…etc) but before that it delivers development releases at the end of iterations into pre-production servers.

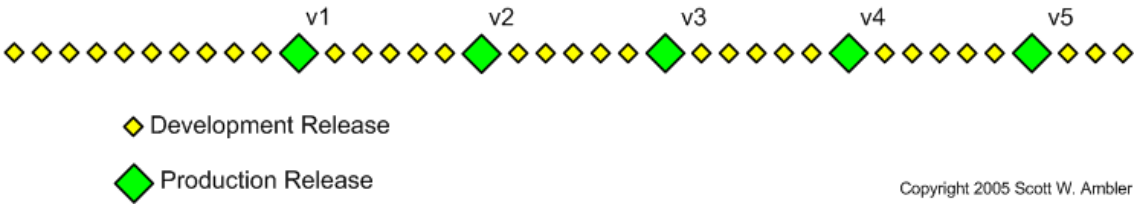

Figure 4: Incremental releases over time (Scott W. Ambler + Associates, 2006).

The figure above shows that production release only comes after many development releases. Moreover, the first production release is normally longer than the later ones. For example, if the first production release takes 6 months, then the next production release may be delivered in a shorter time, e.g. 4 months.

UP Agile philosophies are based on five principles (Scott W. Ambler + Associates, 2006). The first principle is that teamwork is oriented. They may need some high-level guidance from time to time. The second one is the simplicity in which a few papers not thousands explain everything briefly. The third principle is the agility which follows the principles of the Agile Alliance. The forth one is the focus on high-value activities. As stated in the characteristic of Agile, it attempts to remove unnecessary activities to lighten risks and be successful. It focuses, therefore, on activities that actually count. Final principle is the tool independence in which any tools are welcomed while applying Agile UP, such as, any open source tools.

Feature-driven development (FDD):

FDD claims to be one of the lightweight Agile methods (DeLuca, 2007). It is motivated from a client-valued feature perspective in certain time. The method was developed by Jeff De Luca in 1997 for a development project in Singapore (DeLuca, 2007). The project involved a fifteen-month period and fifty software specialists. Jeff De Luca used five processes to develop.

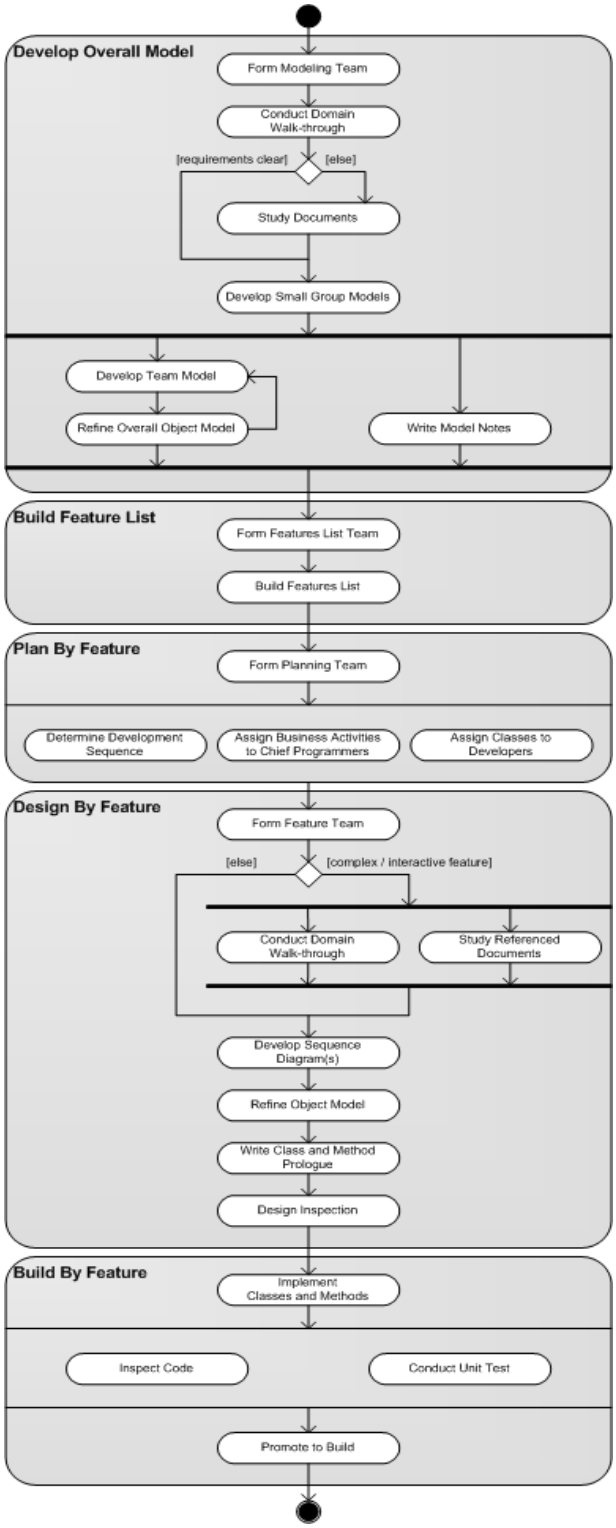

Figure 5: FDD process model (Nebulon Pty. Ltd, n.d.).

As seen in the figure above, there are five processes (Highsmith J. A., 2002). The first process is the develop overall model. It started with a high-level of understanding of the system. Then, many detailed models were created. Each model consisted of a small group. Each model (or combination of models) was selected to become the model for a certain domain area. At the end, these domain area models were gradually merged to create the overall model. The second is to build feature list. After completing the first process (develop

overall model) and after gaining enough knowledge, the team identifies the list of features. Each area covers some of the business activities and forms the categorized feature list. The feature list contains the features that are important to the user (client-valued functions). Each feature takes no more than two weeks. If not, then it should be broken into smaller one. The third process is to plan by feature. This process comes after completing the feature list described above since it starts with developing the plan for each feature by assigning each feature as a class (each class has an owner) to the programmers. The forth process is to design by feature. In this process, the senior programmer selects some of the related features that can be finished in no more than two weeks. He/she prepares the diagram for each feature enhancement for the overall model. Then, after writing the class, a design review is held. The fifth process is to build by feature. In the previous process, client satisfaction was understood. This step is to produce a completed feature by developing the code for each class. Then each unit is tested to be able to integrate it into the main building up of the software.

After reviewing agile concepts and some of its methods, the next section presents and explains the innovated SPM methodology.

Software Project Management methodology (SPM)

This paper innovated the first version of the SPM. Therefore, it stretches and enriches this type of software development and overcomes some of the difficulties encountered when applying other agile methods. It provides a good approach to software project management and solution delivery. SPM is an iterative and incremental approach that uses the principles and characteristics of agile software development and the IIFO method, which prioritizes project items that need to be delivered. The following section presents and discusses SPM in more detail.

Definition of SPM

SPM refers to *"Software Project Management" method* which is a management process that builds systems that is flexible, easy to amend during the development process to achieve customers' needs and their requirements within a specified time-scale and deliver the work incrementally and iteratively with a high percentage of success.

The paper verifies that SPM is simple to understand, easy to manage and is a lightweight method. In order to apply SPM, there are some sequence of processes involved that perform the methodology.

The first stage has many processes. To begin with, client and developers' team should agree on a business case that captures the reasoning for the project and presented in a semi-structured written document. This document can be modified during the project's time-frame. Therefore, the business case helps developers and clients (end-users) in defining the main aim of the project and its initial objectives. Once the business case is captured, the product vision will be rich and will sets the direction and guidance to both developers and end-users. This results in defining and describing the scope of work (SOW) which covers the milestones, deliverables, time-line and the software product that is expected to be produced. The previous processes construct the initial understanding of the project requirements. Once these processes are defined, both parties can complete the project charter and signing the agreement. Then both parities participate in preparing the initial planning which helps to define and build the developers' team.

The second stage takes no more than two weeks. This stage reviews the documents and label the required products and services. These requirements should be understood and documented but they can be changed during the project. The documentation includes the initial requirements' list. At the end of each session, the team briefly repeats what they have understood to clear up any misunderstanding Depending on these products and services, the

team is build. The team has a project's manager who direct the software team, minimize risks, offer a healthy environment for the team and make sure all roles are defined and tasks are completed within time-scale. The developers team has one or more sub-teams which consist of programmers, analysts, documenters, testers and customer's representative (end-user). Each sub-team consists of no more than five specialists. The leader of the sub-team is a senior programmer or analyst and contains a end-user representative who can understand users' needs such as an analyst or a chosen representative from the client's side can hold this position.

After building the right team and its sub-teams, the project moves forward to the third process which called the prioritized requirements process. once the team has understood the scope of the project and documentation and the initial requirements activities, the team moves to make a new list that contains the important products' categories by applying Important-in-First-out (IIFO) for organizing and manipulating the product categories. The output for this process is a prioritized list of requirements activities. This process should not take more than 4 hours.

The iterative and incremental process is the forth process of the SPM. The team break each category into tasks. Then combine them together into segments. Each segment should not take more than three weeks to deliver. Each completed segment should go through testing process and pass through quality assurance process (QA), functional tests and then user's acceptance. Once the segments pass these steps, they can be integrated into the beta version (development environment) of the software. This process is iterative and incremental.

The final process comes after the last segment. In this process, the complete beta version go under a complete testing process (QA, functional tests and through the client's acceptance process) as a piece of completed integrated software. This means that the beta version is ready to release. Therefore, at the end of this process, the client signs the acceptance form and start working on the software under the support agreement. This process takes no more than two weeks.

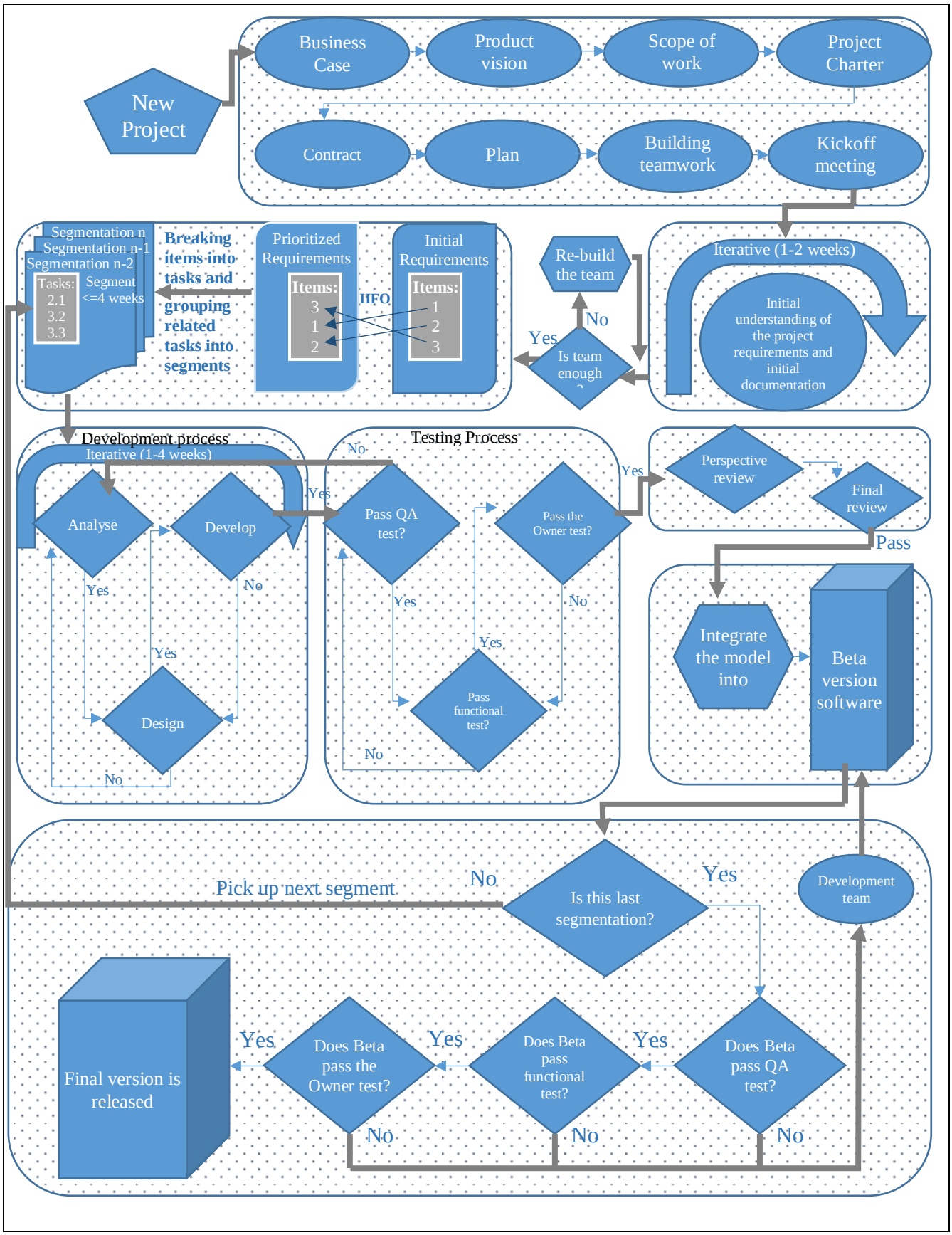

Figure 6: SPM methodology

Explains the SPM from the perspective of Agile

This section explains the SPM's method from the perspective of Agile. To begin with, SPM complies with the Agile principles. SPM involves customer satisfaction; therefore, changing requirements are welcomed. Software is delivered frequently in small pieces, developers and clients work together in self-organizing teams (face-to-face conversations are essential as stated in 4.1). The methodology is simple, easy to apply and understand. Moreover, SPM contains the characteristics of Agile, i.e. the project is broken into small modules of short development cycles. This generates fixed periods to achieve development, prioritizes requirements, minimizes risks, is incremental, people oriented and encourages team collaboration.

Research contribution to knowledge

The research findings helped deepen the understanding of the different Agile methods. The study is distinctive in identifying and presenting a new agile method that can be a solid ground for further improvement. SPM has improved sequentially inside RSAF's environment during many in-house and outsourcing software projects. The study's findings, therefore, are contributed to the knowledge a new approach that match Agile concepts with better understanding and a good explanation in regards to Agile methods.

Implications of the study

This research has theoretical, practical and methodological implications from which academics, governments and firms can benefit. To begin with, this study has a theoretical implication related to its research design through using the frameworks developed in this study and these can be applied as an initial framework for future studies.

The practical implications of this study can help firms and governments to apply SPM in their software projects. It provides an explicit and solid ground roadmap for private and public sectors in their software projects.

Finally, this research provides a methodological implication. It provides a good and new innovated agile method that provides solid ground for theoretical discussion and improvements.

Further work

Further work may build upon this research by improving the method and its framework. This study suggests conducting SPM methods in different environments, cultures, circumstances and time scales to extend the research findings.

Acknowledgment

I am very grateful to Royal Saudi Air Force (RSAF) for giving me the opportunity to apply SPM methodology into their software projects. My thanks also go to my colleagues at the Directorate of Communication and Information Technology (DCIT) for their support and for the valuable suggestions and discussions. In addition, I want to express my gratitude to the clients for their participation by offering their valuable time, which helped me conduct this research.

References

Agile Alliance. (2013, 6 8). *What is Agile Software Development?* Retrieved 6 28, 2015, from http://www.agilealliance.org/the-alliance/what-is-agile/

DeLuca, J. (2007, April 01). (S. Roock, Editor, & It-agile) Retrieved July 20, 2015, from http://www.it-agile.de/fileadmin/docs/FDD-Interview_en_final.pdf

DeLuca, J. (n.d.). *version 1.3 of the Feature Driven Development processes*. Retrieved July 20, 2015, from version 1.3 of the Feature Driven Development processes: http://www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

Highsmith, J. A. (2002). *Agile software development ecosystems.* Boston: Addison-Wesley.

Highsmith, J., & Cockburn, A. (2001, Sep). Agile software development: the business of innovation. *Computer, 34*(9), 120-127.

Kerr, J., & Hunter, R. (1993). *Inside RAD: How to Build a Fully Functional System in 90 Days or Less.* McGraw-Hill.

Martin, J. (1991). *Rapid Application Development.* Macmillan.

Nebulon Pty. Ltd. (n.d.). *FDD Process Model*. Retrieved July 20, 2015, from Feature Driven Development: www.featuredrivendevelopment.com/files/FDD%20Process%20Model%20Diagram.pdf

New Product Development Game. (1986, Jan 01). New Product Development Game. *Harvard Business Review*, 137-146.

Scott W. Ambler + Associates. (2006, 01 01). *The Agile Unified Process (AUP)*. Retrieved July 15, 2015, from Effective Practices for Software Solution Delivery: http://www.ambysoft.com/unifiedprocess/agileUP.html

Scrum.org. (2015, Jan 01). *What is Scrum?* Retrieved July 15, 2015, from Scrum.org: www.scrum.org/resources/what-is-scrum