CSCI 5608 – SOFTWARE PROJECT MANAGEMENT

PROFESSOR MEGAN R. MACMILLAN, PHD.

SPRING 2016

AGILE CHANGE MANAGEMENT

Pallavi Madasu

University of Colorado, Boulder

Contents

1 Abstract

Nowadays, most of the projects are run in agile fashion as the requirements keep changing and evolving very often, to be competent enough and maintain speed to market. Agile principles encourage team involvement throughout the development life cycle and it is a very cooperative and collaborative approach. This provides excellent visibility for key stakeholders and risks can be identified and handled at the earliest stage, hence, there is a need for a streamlined and flexible approach to requirements change management. Handling the changes should be smooth and natural without any glitches and one of the best ways to do is prioritizing the tasks to be implemented. This not only helps in delivering high-quality end product but also maximizes the stakeholder Return on Investment.

The goal of this research is to examine and understand agile change management methodologies and processes in order to identify the problems and issues that may arise in essential change management and requirements improvement. I want to gain knowledge about various requirements management frameworks in agile like Scrum, Disciplined Agile Delivery (DAD) and their prioritization strategies. As part of this study, I have interviewed project managers from two companies, who develop software projects using agile methodologies and principles, to comprehend the challenges of requirements change management and different processes used to address them in the real world.

2 Introduction

The problem of managing software changes has gained a lot of attention in recent years, especially during the past ten to fifteen years, there is a rapid increase in the development of software for electronics. Every day the needs of the users keep changing which led to the creation of smart phones, tablets with thousands of applications being released to support them. This growth and change in the industry has led to the adoption of effective software engineering processes. [1]

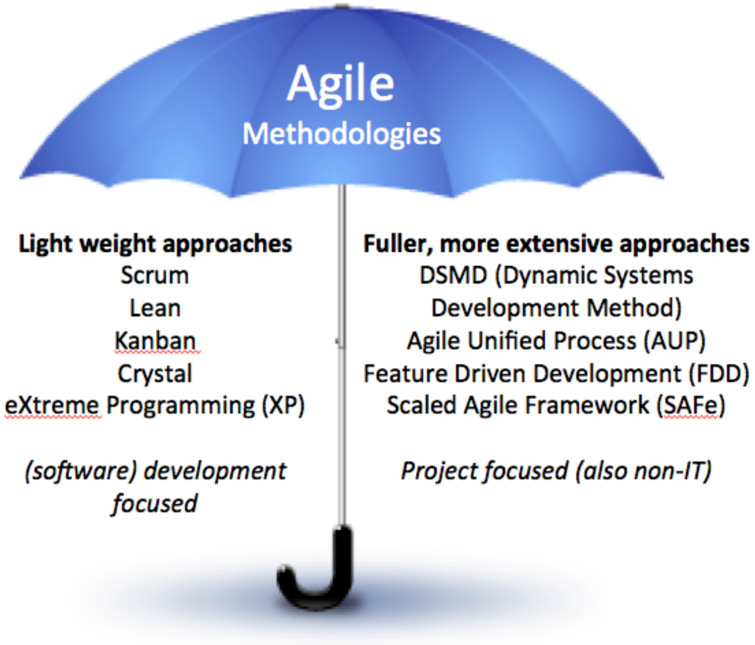
More than one-third of major project risks for software and system development projects arise from late changes, either in a customer's needs or due to incorrect or incomplete requirements [2][7], which usually throws software development managers under increased pressure to deliver the software within stipulated time, though the budgets are being cut down significantly, Business environments continue to change at a rapid pace, despite the fall of economy resulting in software development firms striving to keep pace with change. These circumstances have led to an increased need in agile software development methodologies as they guarantee the delivery of quality software with flexibility and rapidity. During the entire software development life cycle, agile methodologies such as extreme Programming (XP), Feature-Driven Development strive to slash the cost due to changing requirements. For example, XP uses swift development cycles and follows iterative planning to deliver the most priority functionalities of the system at the earliest possible. XP ensures rigorous testing from day one leading to early defect detection and resolution, in addition to the continuous feedback received from customer, resulting in a high quality software as the outcome of the process. [3]

A consistent change management process is needed to address the risks in the projects following iterative development model. The objective of the change management process is to record and track change requests like reprioritization of requirements according to the needs of the customer and incorporating them in the development iteration cycle. [2]

Practicing agile methodologies in the real world poses more challenges in spite of its early success. There are some factors that hinder the widespread adoption of agile. One of the biggest issues with following agile is to get support from senior management for implementing what seem like dramatic changes in the application development. This needs all the participants including stakeholders, managers and developers to be on the same page and requires usage of frameworks and tools very different from the traditional development model. Agile software development teams must and do embrace change, accepting the idea that requirements will evolve throughout a project. As Agile appreciates that requirements evolve gradually, there is a risk that any early investment in planning or documenting would be unfruitful. In short, agility strives to truly manage change, not to prevent it. Thus, an efficient and effective process is needed to manage changes in requirements and to deliver the high-value end product. [3]

3 Agile

"Agile is a time-boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end" [11]. Agile is a collection of software development practices and methods based on the principles and values behind the agile manifesto. The below diagram of agile umbrella summarizes the most popular development methodologies under agile. [21]

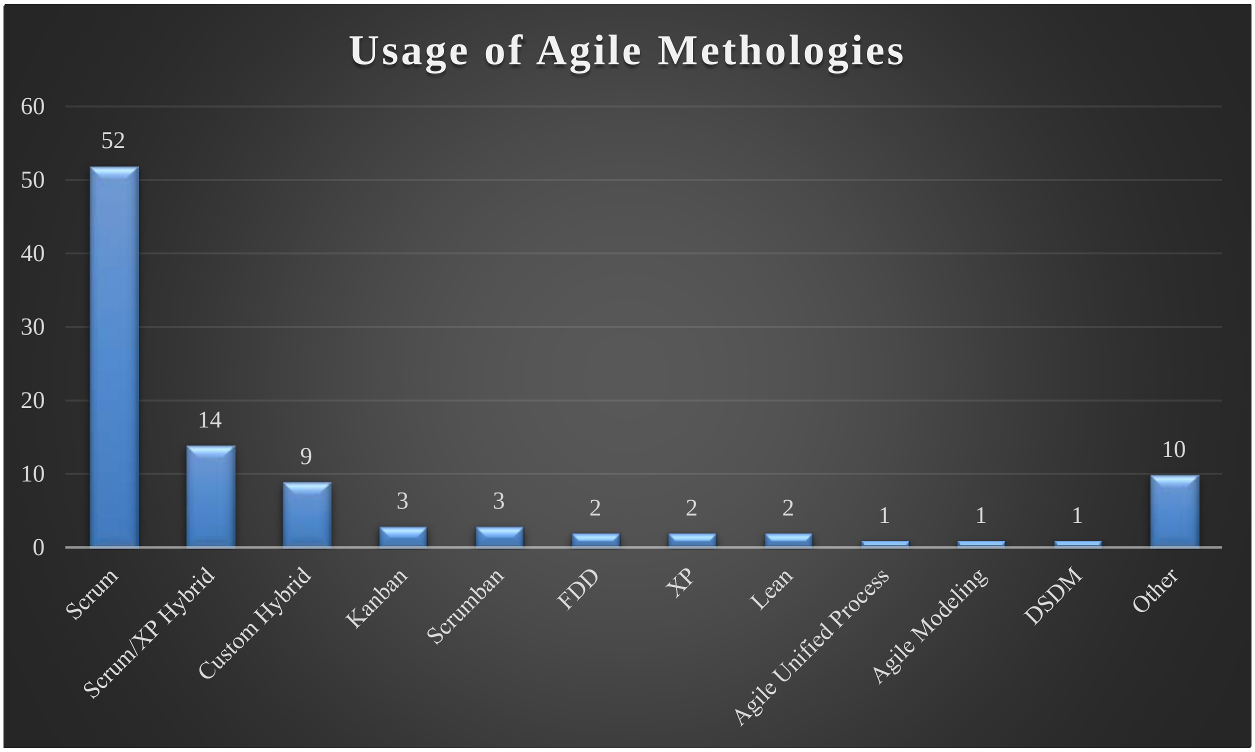

Agile
Methodologies
Light weight approaches
Scrum
Lean
Kanban
Crystal
eXtreme Programming (XP)
(software) development focused
Fuller, more extensive approaches
DSMD (Dynamic Systems Development Method)
Agile Unified Process (AUP)
Feature Driven Development (FDD)
Scaled Agile Framework (SAFe)
Project focused (also non-IT)

The crux of Agile model is incremental and iterative development. [10] Though the philosophy behind agile is the same, there are different agile methodologies mentioned below, each has its own recipe of practices, tactics and terminology from implementation perspective. Let's discuss each of the software development focused light weight agile approaches in detail.

1. Scrum
2. Extreme Programming (XP)
3. Lean
4. Kanban
5. Crystal

Though Agile is a blend of accepted and controversial software engineering practices, it is being adopted by many software development companies. Project parameters such as the objective, size of the team, length of the project, schedule, budget, clarity about requirements, stability of the proposed system determine the methodology to be chosen to best suit the needs of the project. Sometimes, a hybrid development model can also be designed for better results. [10]

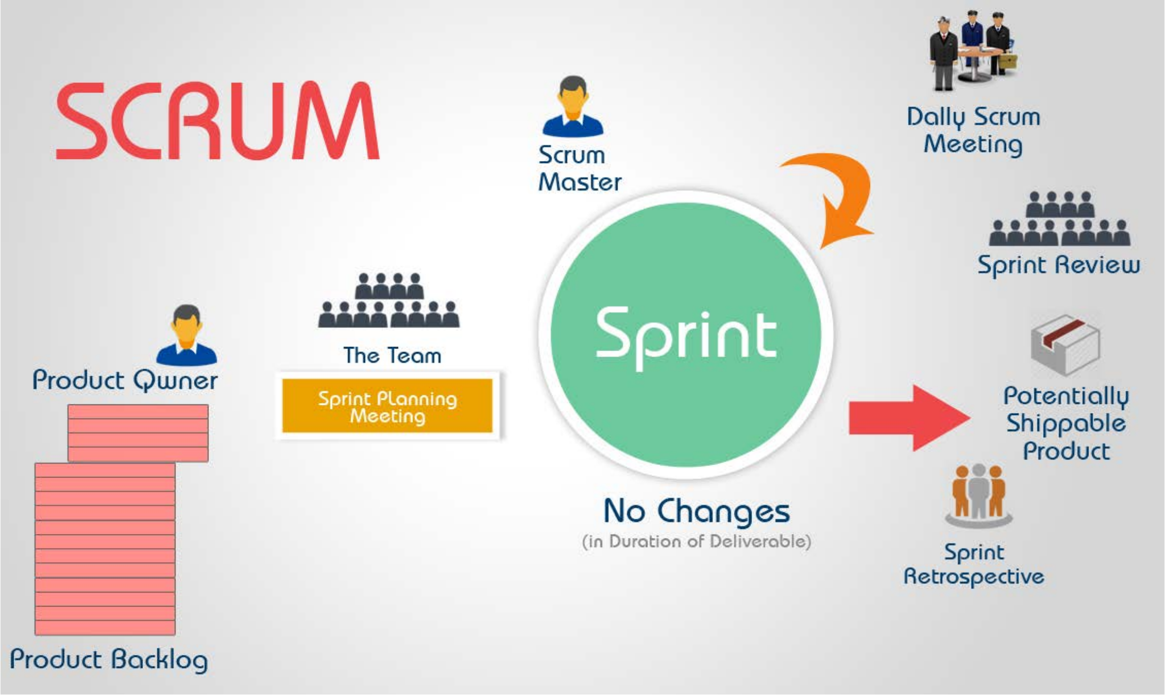
In 2011, Version One conducted Agile Development survey and the conclusion was that the most widely used agile software development methodology is **Scrum**. The below graph shows the usage of various agile methodologies in the real world. [13]



3.1 Scrum

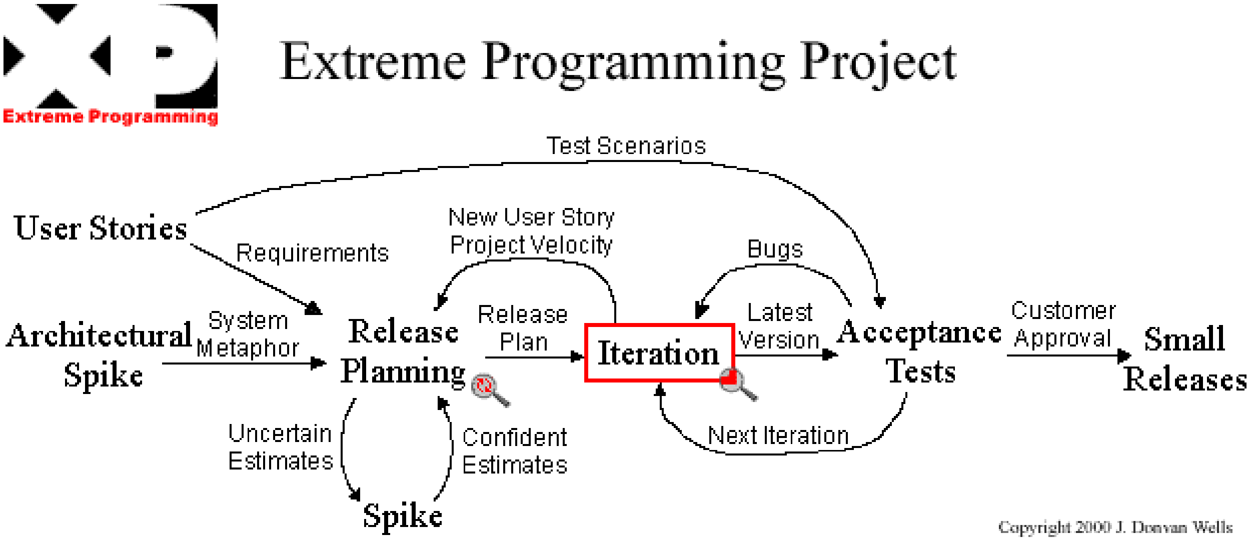
Scrum is an agile framework to manage projects, especially software development. It is one of the most widely practiced agile methodologies, suited for projects with constantly changing or emerging requirements. Work progresses in iterations called sprints. There are 3 main roles in Scrum model, explained in detail below:

- **Product Owner**: The product owner represents the business, customers, stakeholders and guides the team towards developing and delivering the right product. This person builds and manages the product backlog, which is a stack of requirements and defects in the order of priority. Product owner has the final authority to prioritize or reprioritize the requirements. [15] [16]
- **Scrum Master**: A scrum master is a facilitates the team to perform at their best. He / She maintain balance between the product owner, stakeholders and the team. Scrum master is responsible for conducting daily standup meetings, retrospective meetings, removing any impediments team is facing, ensuring that the project is on track and progressing well as per the plan. [15]
- **Team:** Scrum counts on self-organizing and cross functional teams. Self-organizing team means that there will be no boss to dictate and assign the tasks to each person in the team, the tasks are divided by the team as a whole. Cross functional team implies that everyone team member does everything to transform an idea to implementation including design, development and testing. [15] A typical scrum team consists of 5 – 10 people; they work together with "we" attitude to complete the committed work in each sprint.



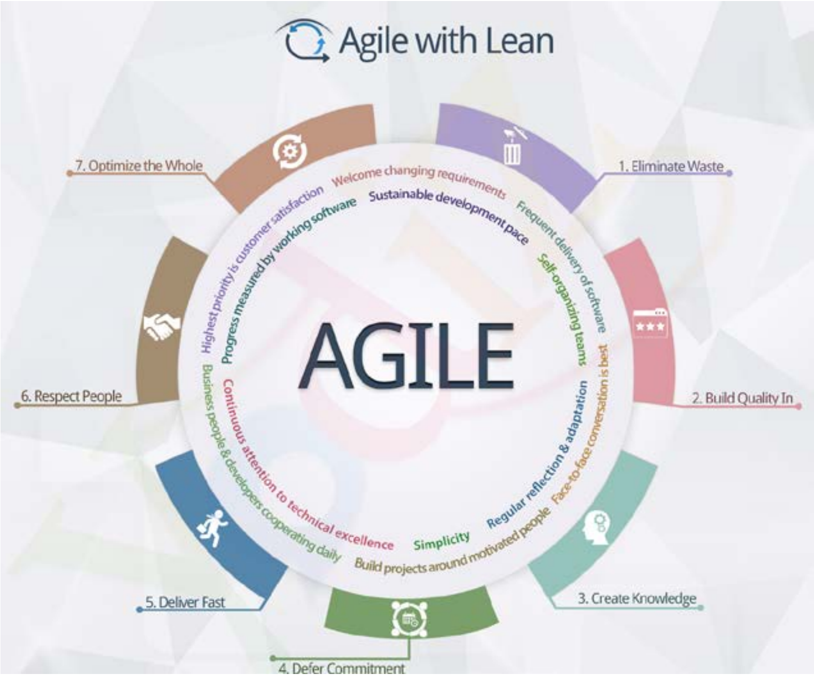
3.2 Extreme Programming (XP)

The focus of Extreme programming is “Customer Satisfaction”. “XP delivers the software you need as you need it”. [18] It empowers the team to be confident to respond to changing requirements, even late in the development life cycle. Extreme Programmers constantly communicate with the customers to get the feedback by testing the product from day one and pair programming. XP improves the software by communication, simplicity, feedback, respect and courage. [18]



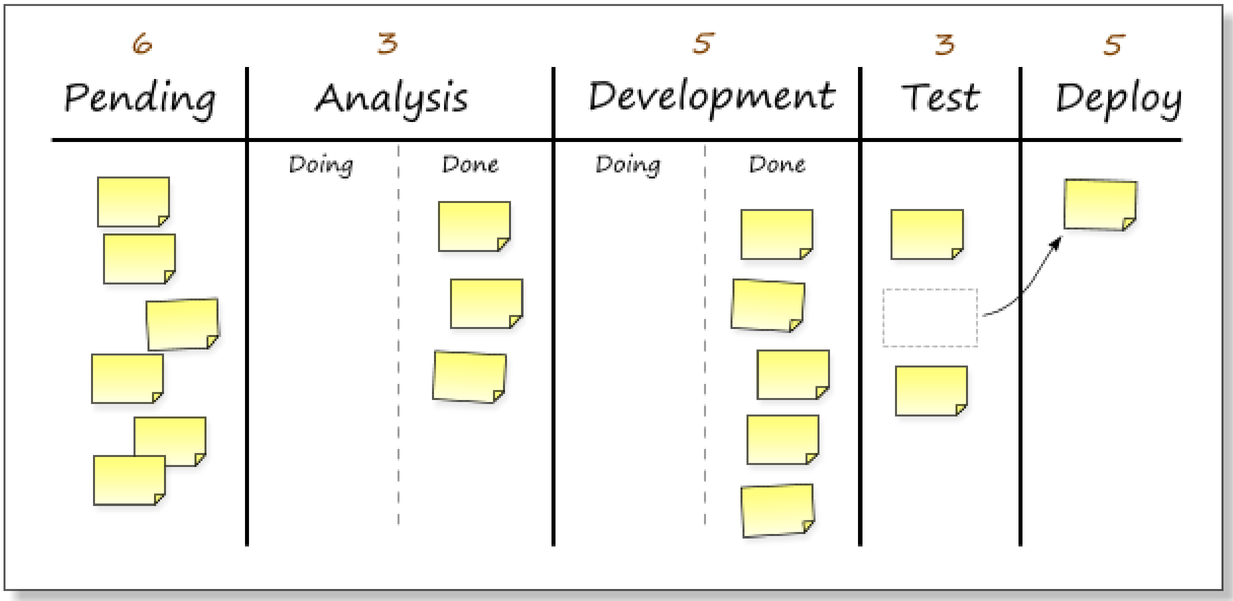
3.3 Lean

The core principle of Lean is to reduce or eliminate wastes or non-value added activities. It empowers the team by amplifying learning and delivers the product as fast as possible by deciding as late as possible. Hence it increases the customer value and satisfaction. The Agile process itself follows lean principles for software development. Lean and Agile are merging and adopting each other by following the processes to be agile and responsive to the changing needs of customers. Lean also measures progress of the project with incremental releases of working software. [24] [26] In one of the Forbes articles, “Rally says that its customers get to market 50% faster and are 25% more productive when they employ a hybrid of Lean and Agile development methods”. [25] The below picture shows how agile and lean are connected. The core principles of lean are mentioned on the outer circle and the principles of Agile are shown in the inner circle.


Agile with Lean
AGILE
1. Eliminate Waste
2. Build Quality In
3. Create Knowledge
4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the Whole
Welcome changing requirements
Sustainable development pace
Frequent delivery of software
Self-organizing teams
Face-to-face conversation is best
Regular reflection & adaptation
Build projects around motivated people
Simplicity
Continuous attention to technical excellence
Business people & developers cooperating daily
Highest priority is customer satisfaction
Progress measured by working software

3.4 Kanban

Kanban is an incredibly powerful and simple agile framework that emphasizes on “Just-In-Time” delivery. By creating a visual model of workflow and limiting work in progress, the bottlenecks can be revealed easily and addressed. The overloaded teams can be relieved from by analyzing the workflow and sharing the work load among the teams, leading to continuous improvement. The diagram shows how sticky notes are used to visualize the workflow in Kanban [20]



3.5 Crystal

Crystal is a family of development methodologies because Mr. Cockburn believed that one size cannot fit all the projects. Depending on the criticality of project (Comfort, Discretionary Money, Essential Money, Life) and size of team, the methodology will be chosen and assigned a different color. Crystal emphasizes heavily on face-to-face communication, prefers collocated team rather than distributed team. Crystal clear is one of the popular crystal methodology where the following are the main steps in the delivery: [22] [23]

- **Chartering**: Requirements gathering, analysis and initial plan would be developed.
- **Cyclic Delivery**: Development consists of 2 or more delivery cycles where each delivery cycle ranges anywhere from one week to three months.
- **Wrap-up**: Software would be released and based on the customer feedback, the changes will be implemented.

4 Change Management

4.1 Change in Requirements

Change in Requirements means modification or transformation of current behavior to a different one. Change is inevitable during the course of the software development life cycle, and there could be various reasons for the change. Changes can include addition of new or modification or removal of existing requirements. The basic reasons for the change are mentioned below:

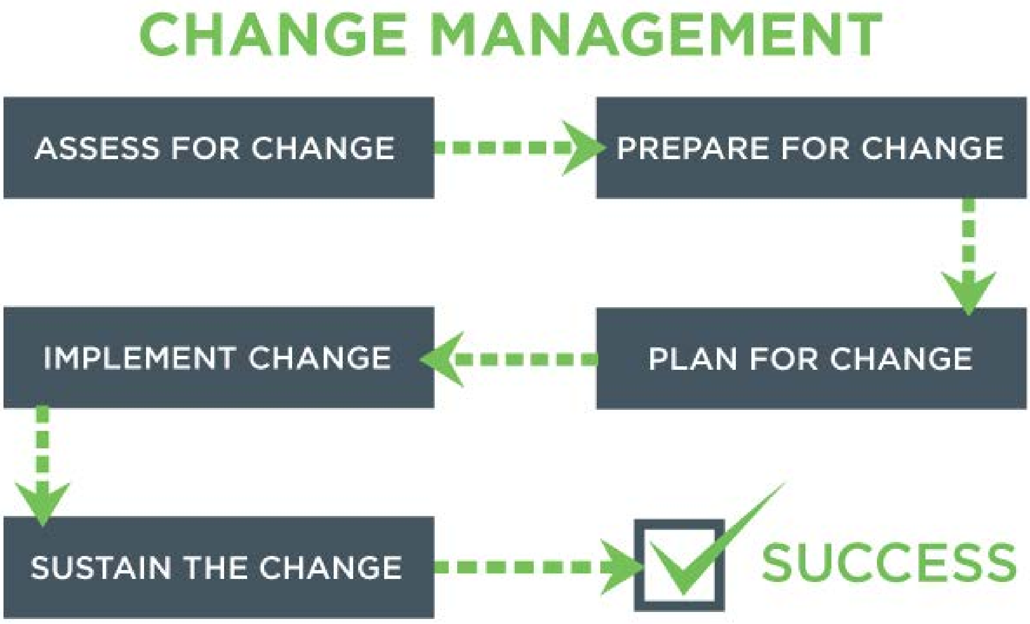
1. Incomplete Requirements
2. New Customer Requirements
3. Incorrect / Redundant requirements
4. Change in Market Place
5. Technologies Evaluation
6. Organization Changes / Politics
7. System Empowerment
8. Defect Identification / System Implementation Errors

4.2 Need of Change Management

"Change management is the task of managing and enabling changes rather than preventing them" [7]. It is a disciplined, streamlined and flexible way of implementing changes in the requirements. One of the challenges in traditional change management is that every new requirement or change request is considered a significant risk to the project schedule and quality of delivery, especially for the projects with predefined deadlines and budget squeeze. Almost all the software and system development projects are subjected to ever changing circumstances like reprioritizing customer's needs or modifications in system due to its inherent nature.

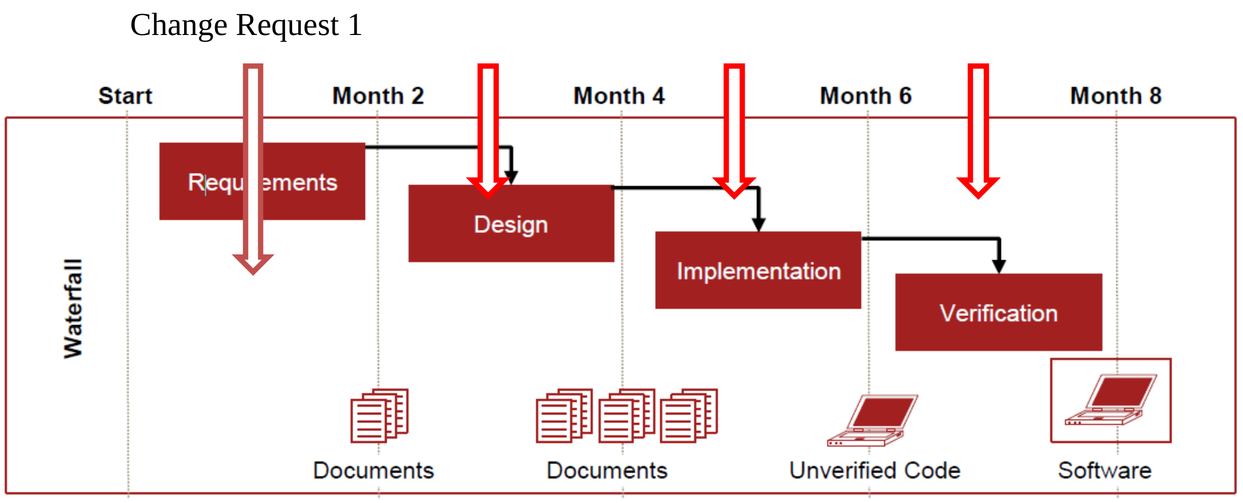
Implementing the changes straightforward in the system in accordance with the desired new behavior could possibly break the existing system developed so far partially or totally. Therefore, changes must be planned, tracked, and released without compromising the quality of concrete system or subsystem. Change management process should involve everyone affected by the change request including customers, users, managers, developers and testers. Therefore, Change Management is a vital part of the overall development and engineering process. [7]

The below diagram shows the key steps in change management: [14]



4.3 Traditional vs Agile Change Management

In traditional development models, the scope, quality, time and cost of the project are locked down and changes in any of these attributes needs Change Management. Non-Agile methodologies do not anticipate changes. Usually, when the scope changes in traditional projects, the project takes longer, incurs more cost and the quality of the project suffers. Changes requested may be approved by the project manager depending on the priority of the change ensuring that implementing the change does not compromise the quality of the product. The below diagram depicts the inadaptability of waterfall model to change requests. [9]

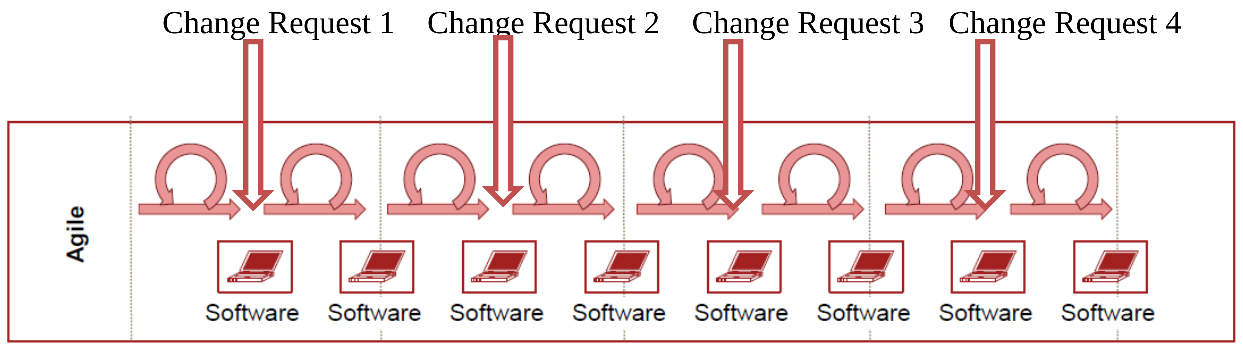


Agile encourages and enables the product owners to change their requirements and even overall goals during the development of the software. Agile supports scope flexibility but the cost, time and quality of the project are fixed. The reason behind the success of Agile methodologies is that the high priority requirements providing the most business value will be delivered first.

Agile needs a robust and reliable change management system to embrace, plan and track the progress of change requests. The rationale for using Agile methods is Rapid Change Management. Thus Change Management is critical and essential in Agile model and it is also acknowledged as one of the principles behind the Agile Manifesto. [7]

“Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.” [7]

The below diagram shows the flexibility of handling change requests anytime through the iterative and incremental model before the delivery of the end product· [9]



4.4 Characteristics of Change Management in Agile

The following are the significant characteristics of change management in Agile:

1. The objective of Agile is to embrace Change Management.
2. Change Management is inevitable in Agile.
3. There should be no change management in the middle of an iteration.
4. Change management in turn includes prioritization.
5. Change management does not change highest value story scheduling.
6. Change management provides a disciplined, streamlined way to manage and resolve the challenges encountered by mitigating the risks involved.

4.5 Prioritizing Requirements

Some of the most popular ways to prioritize the requirements in the backlog are described: [27]

- **MoSCoW:** Each requirement in the backlog will be tagged to MSCW, where "M" being the highest and "W" being the lowest priority.
  - **M – MUST** have this
  - **S – SHOULD** have this if possible
  - **C – COULD** have this if there is no impact on anything else
  - **W – WON'T** need this time but may be in future

- **Business Value Based:** Each requirement offers some business value to the company, which will be decided by the product owner on behalf of all the stakeholders. The requirement that delivers the highest business value will be given highest priority and implemented during the initial releases.

- **Technology Risk Based:** Requirements are prioritized based on the risk involved in implementing them. The requirement that has the highest technical risk associated will be given highest priority and implemented during the earlier iterations.

- **Kano Model:** The requirements are assigned priorities based on the customer preferences of qualities like Must-be/Basic, One dimensional, Attractive, Indifferent and Reverse. [28]

- **Walking Skeleton:** In this method, the requirements are picked to build a minimum viable product with selected end to end features to be developed in a short span of time.

- **Validated Learning:** Features are selected based on the highest market risk. The new product would be released in the market to get the feedback and the learning will be applied to develop new features.

4.6 Change Management in Agile

"Agile Change Management is concerned with increasing the ability of the project to be responsive to requests for change and to quickly implement accepted change requests." [29]

The principal factors for the success of agile change management are the use of:

- Iterative and incremental development with short feedback cycles – Key to embrace Change
- Close Customer collaboration – Key to Responsiveness

Responding effectively and quickly to change is rapid when the following are minimized:

1. The cost of knowledge-transfer between the team members.
2. The amount of knowledge captured or updated in the artifacts.
3. The waiting time in making decisions about implementing changes.

The two most important aspects of Agile Change Management are: [29]

- **Be more responsive to change requests:**
  - *Short and Frequent iterations* – Shorter iteration cycles and releases help realize the functionality sooner with the help of working software and the priorities can be altered faster.
  - *Collaboration and Communication* - between customers and developers on a regular basis helps take change control decisions more effectively by participatory decision making. Product owner is empowered to make any decisions about the changes or issues that arise during the development. Customers and stakeholders should be involved by providing updates about the progress of the project, changes and defects by deploying appropriate tools.
  - *Risk Analysis and Intimation* – Product owner should be given control of the scope and priorities. Changes should be allowed and accommodated through a highly collaborative agreement process by informing the customer about the risk and the impacts on scope, schedule, quality, cost, the proposed changes may introduce.
  - *Team Empowerment* – Authorizing the team members to correct problems with the code or structure without having to wait for the formal approvals and reviews, as trust is one of the important factors in Agile.

- **Implement those changes more quickly and easily:**
    - *Short and Complete cycles* - Keep the development cycles short, test-driven and complete, this gives an opportunity to get better feedback from the customer.
    - *Product Design* - Simple and emergent designs should be mandated to make the code simple, readable and easy to change.
    - *Minimizing non-code artifacts* - Number and size of non-code artifacts like documentation, approvals to be produced or updated should be minimized, helps in faster implementation of changes.
    - *Focus* – Team should focus on the features scheduled for the current iteration.

5 Interviews

As part of the research, I have interviewed project managers from two different companies, who adopted Agile methodology to develop and deliver software applications. They faced many challenges during the process and adapted various tools and techniques for successful delivery.

5.1 Case Study 1

Organization: Verizon, USA
Website : http://www.verizon.com/home/verizonglobalhome/ghp_landing.aspx
Project Name: Verizon Enterprise Center (VEC)
Team Size: 38 (4 teams)
Software Development Methodologies: Waterfall, Agile - Scrum
Project Duration: Feb 2010 - Oct 2015

5.1.1 Description

Verizon VEC is a legacy ordering system for enterprise customers and the system interacts with more than 50 other systems to deliver service to a single customer. This required a lot of manual effort to perform repeated as well as redundant tasks. The time to provision the services increased leading to customer dissatisfaction and losing to competition. In the year 2010, Verizon decided to build a new ordering system using Waterfall Model with new technologies with the concept of "One function – One system" by decommissioning all the legacy applications.

The team worked on the planning, requirements gathering and prepared design documents for 6 months and started building new system using new technologies in the year 2010, requirements were categorized and distributed to multiple teams. The whole system has been implemented, merged and deployed for the integration testing in the year 2012. During the testing phase, the project encountered many unpredicted design and requirement gaps and could not enable all the needed services to the customer, leading to a big failure. Then the team adopted Agile Scrum methodology along with Change Management techniques to address the change requests.

5.1.2 How did Agile help to become successful?

Following scrum processes were followed by the team to be agile and successful.

Sprint Duration – 2 weeks

- All the requirements were divided into multiple **user stories** and each user story was further split to small working components of maximum 8 hours each.
- A **prototype of the application** was made before the actual development starts to make sure requirements are understood in the right way. Changes will be made to prototype as per the customers needs and wants and it will be given as a reference to developers.
- Every second week of the sprint, a **sprint planning meeting** was scheduled for the upcoming sprint with the customer.
- At the end of each sprint, a **retrospective meeting** was scheduled to go over what went well and what not went well and other development and testing concerns of the team. Demos were given to the customer at the end of each sprint by showing working software to get **customer feedback**. Changes requested were easily incorporated in the next sprint
- **Customer was constantly involved** with the project at every phase of the development.
- Code was **continuously integrated, deployed and tested everyday**, there are separate testing teams to test end to end functionality of the system to avoid issues proactively.
- During the sprint, Solution Architects and Business Analysts focused on the technical and functions design to provide clear requirement details to the development team and be able to minimize the issues and meet the expectations easily.
- After each sprint, the team picks the **prioritized features** from the backlog and repeats the same process.
- At end of each 8 sprints, the team has completed implementing Install Ordering system and new orders are placed in the system successfully.

5.1.3 Management Feedback

Manishi Sharma, Project Delivery Head said "VEC is our own system and we know end to end functionalities and we initiated the project to build new system and deploy at one shot and we waited to complete the development, but when it came for testing we had lot of design and requirement issues which was unexpected and had lots of challenges, later we decided to adopt agile change management methodologies and tracked down the entire work to small features and prioritized them and closely monitored each sprint and implemented changes whenever required and successfully developed the new system"

5.2 Case Study 2

Organization: Ericsson, USA
Website: http://www.ericsson.com/
Project Name: CenturyLink Lion (CTL LION)
Team Size: 44 (5 teams)
Software Development Methodologies: Waterfall, Agile - Scrum, Rally
Project Duration: Feb 2014 - Nov 2016 (Ongoing)
Scrum Master Interviewed: Suhas Tare

5.2.1 Description

CTL LION is an order management and billing system, it enables CenturyLink to sell L2VPN Ethernet Services to their customers. In the year 2014 Ericsson signed an year contract to deliver the whole application with 2 Ethernet services to CTL using traditional development model. After 8 months CTL has requested new L3VPN services to accommodate with the existing functionalities with additional features. The existing design did not support the new service and with the Waterfall model, the new features could not be implemented, so Ericsson failed to accept and deliver the customer requirements on time.

As part of the new contract, Ericsson and CTL agreed to deliver the software using Agile Scrum Methodology using Rally tool.

5.2.2 How did Agile help to become successful?

Sprint Duration – 2 weeks

- All the requirements are divided into features, which are further divided into multiple user stories, each user story represents a task of maximum 5 – 8 points, each point represents maximum 16 hours.
- Before the start of each sprint, grooming sessions are conducted.
- Customer prioritizes the features and user stories, every developer reviews the task and provide estimations.
- Tasks are assigned to users and logged into rally tool. Each developer has access to rally tool to download design documents and update the hours burnt.
- Working code will be deployed and tested every day.
- On completion of each user story, scrum master schedules a meeting with customer and a demo will be given to the customer on the working component to get feedback. Based on the feedback, any additional changes needed would be done in the upcoming sprints.
- At the end of each sprint, customers are given demo about all the user stories to get overall acceptance. If there is any gap found in the implementation, the user story will be carry forwarded to the next sprint.

5.2.3 Management Feedback

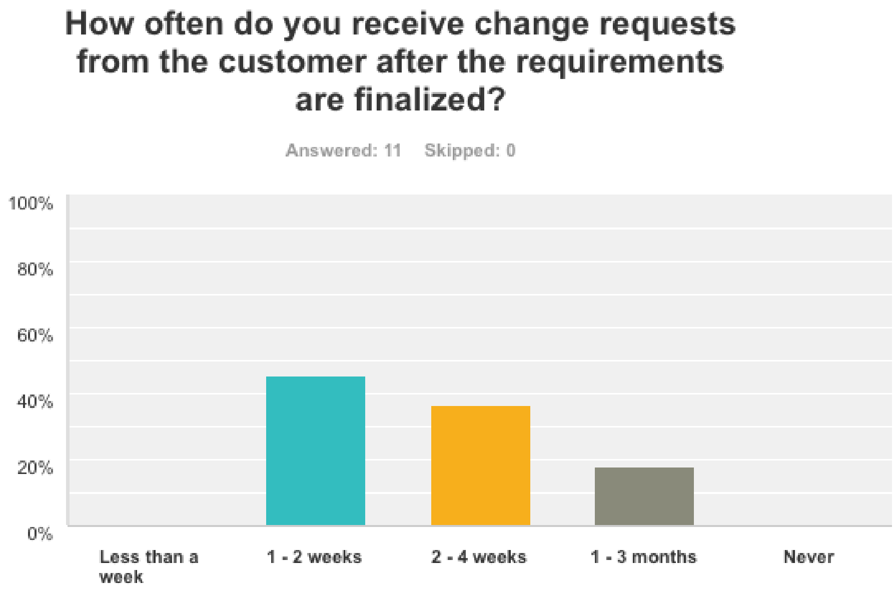
Suhas Tare, scrum master of the project stated that "with the agile scrum methodologies development is transparent and flexible, it helps team to visualize features requirements and do the better design in place. Also requirements can be reprioritized at any point of time and change requests from customer are easily accommodated. At the end of each sprint, customer can see the working component and provide feedback and add additional features if any. Customer Satisfaction was immense with agile as they get what they want and constantly engaging. It was wonderful and smooth journey with agile and change management got easier. We implemented many add-on features for the customers and added more value to the organization."

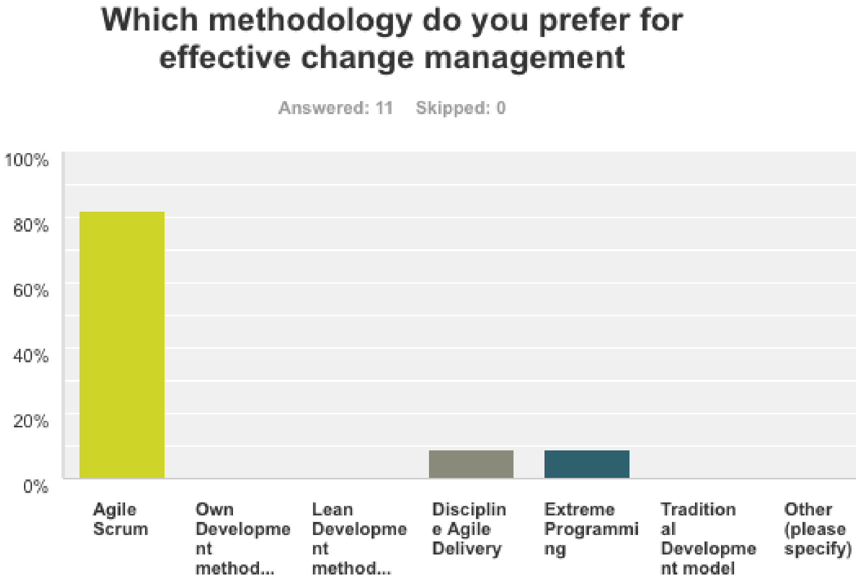
6 Agile Survey and Analysis

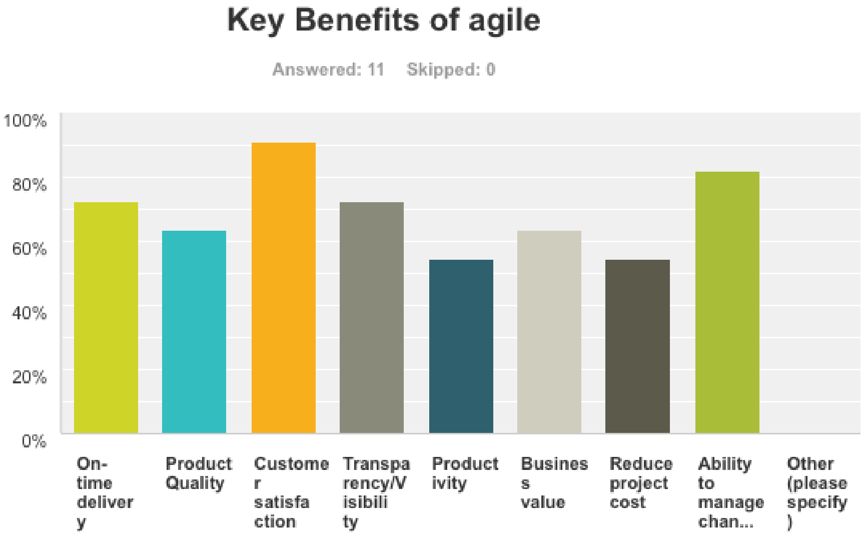
I have conducted a survey to know how employees in software development (developers, managers, architects) at various levels think about Agile. I have also analyzed the reports based on the responses I received. Questionnaire and the survey results are mentioned below:

Survey Link: https://www.surveymonkey.com/r/CB96YSX

Survey Responses: https://www.surveymonkey.com/results/SM-SNWJQTWS/







Interesting Results:

- More than 80% of the people suggest Scrum is the most preferred agile methodology for change management.
- Customer Satisfaction is the most significant benefit of Agile, followed by the Ability to manage changing priorities.

Hence, Agile Scrum is proved to be the most popular, preferred methodology for Change Management.

7 Conclusion

Change management has always been a challenge in software development, whether we use agile methods or not. Rapid change management is the rationale for using agile methods and can add significant value to resulting software. With Agile Change Management, the project would be more responsive to accept and implement changes quickly. As changes can be implemented earlier in the development cycle in agile, risks will be minimized. Todays enterprise solutions are complicated and change rapidly according to business needs. Agile methodologies, being adaptable to change, flexible, transparent and responsive are well suited to the software development projects. Scrum agile methodology has been found to work well with most of the projects in most of the organizations. However, company needs to consider which agile methodologies are beneficial for the company or specific projects and appropriate change control and management techniques should be followed to effectively manage changes and deliver high quality and high value software.

8 References

[1] Makarainen, Minna. "Software Change Management Processes in the Development of Embedded Software." (n.d.): n. pag. Web. 13 Apr. 2016. <http://www.vtt.fi/inf/pdf/publications/2000/P416.pdf>.

[2] "The Standish Group Report CHAOS." (n.d.): n. pag.Https://cdn.projectsmart.co.uk. Web. 13 Apr. 2016. <https://cdn.projectsmart.co.uk/white-papers/chaos-report.pdf>.

[3] "Agile Project Management." (n.d.): n. pag. Http://www.ccpace.com/. Web. 13 Apr. 2016. <http://www.ccpace.com/asset_files/Agile_Project_Management.pdf>.

[4] Ramachandran, Muthu, and Rogério Atem De. Carvalho. Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization. Hershey, PA: Engineering Science Reference, 2010. Print.

[5] Wilkinson, Lloyd. "Agile Development: A Brief History." The Proven Method. N.p., 31 July 2014. Web. 13 Apr. 2016. <http://www.provenmethod.com/agile-development-brief-history/>.

[6] Http://training.leadingagile.com/. N.p., n.d. Web. 13 Apr. 2016. <http://training.leadingagile.com/pmi-agile-certified-practitioner/>.

[7] Berger, Christian, and Bernhard Rumpe. "Supporting Agile Change Management by Scenario-Based Regression Simulation." (n.d.): n. pag. Web. 13 Apr. 2016. <http://www.se-rwth.de/publications/Supporting-Agile-Change-Management-by-Scenario-Based-Regression-Simulation.pdf>.

[8] Ambler, Scott W. "Agile Requirements Change Management." Agile Requirements Change Management. N.p., n.d. Web. 13 Apr. 2016. <http://agilemodeling.com/essays/changeManagement.htm>.

[9] "Adopting an Agile Methodology Requirementsgathering and Delivery." (n.d.): n. pag. Https://www.pwc.com/us/insurance. Web. 13 Apr. 2016. <https://www.pwc.com/us/en/insurance/publications/assets/pwc-adopting-agile-methodology.pdf>.

[10] Awad, M. A. "A Comparison between Agile and Traditional Software Development Methodologies." (2005): n. pag. The University of Western Australia. Web. 13 Apr. 2016. <http://www.unf.edu/~broggio/cen6940/ComparisonAgileTraditional.pdf>.

[11] Rasmusson, Jonathan. "What Is Agile?" What Is Agile? N.p., 2015. Web. 13 Apr. 2016. <http://www.agilenutshell.com/>.

[12] McLaughlin, Mike. "Agile Methodologies for Software Development."VersionOne. N.p., n.d. Web. 13 Apr. 2016. <https://www.versionone.com/agile-101/agile-methodologies/>.

[13] Dhawan, Sanjeev, and Divendu Kumar Mishra. "A Pragmatic Study of Tools Used in Agile Software Development." (n.d.): n. pag. Http://iasir.net/. International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS). Web. 13 Apr. 2016. <http://iasir.net/IJETCASpapers/IJETCAS12-217.pdf>.

[14] "Change Management." Change Management. N.p., n.d. Web. 13 Apr. 2016. <http://navis.com/change-management>.

[15] "Scrum Methodology and Project Management." Mike Cohns Blog Succeeding With Agile RSS. N.p., n.d. Web. 13 Apr. 2016. <https://www.mountaingoatsoftware.com/agile/scrum>.

[16] Ambler, Scott W. "Agile Best Practice: Prioritized Requirements." Agile Best Practice: Prioritized Requirements. N.p., n.d. Web. 13 Apr. 2016. <http://agilemodeling.com/essays/prioritizedRequirements.htm>.

[17] "Process Management." Infimetrics. N.p., n.d. Web. 13 Apr. 2016. <http://infimetrics.co.in/process-management>.

[18] Wells, Don. "Extreme Programming: A Gentle Introduction." Extreme Programming: A Gentle Introduction. N.p., n.d. Web. 13 Apr. 2016. <http://www.extremeprogramming.org/>.

[19] McGee, M., and Fritsky. "What Is Adaptive Software Development?"WiseGeek. Conjecture, n.d. Web. 13 Apr. 2016. <http://www.wisegeek.com/what-is-adaptive-software-development.htm>.

[20] Peterson, David. "What Is Kanban?" Kanban Blog Atom. N.p., n.d. Web. 13 Apr. 2016. <http://kanbanblog.com/explained/>.

[21] "Agile: PMI versus APMG." TenStep Nederland RSS2. N.p., n.d. Web. 13 Apr. 2016. <http://www.tenstep.nl/agile-pmi-versus-apmg/>.
[22] Ramsin, Raman. "Software Development Methodologies." (n.d.): n. pag. Web. 13 Apr. 2016. <http://sharif.edu/~ramsin/index_files/sdmlecture12.pdf>.

[23] Williams, Laurie. "Agile Software Development Methodologies and Practices." Advances in Computers (2010): 1-44.Http://agile.csc.ncsu.edu/. Web. 13 Apr. 2016. <http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf>.

[24] "Lean Software Development." Wikipedia. Wikimedia Foundation, n.d. Web. 13 Apr. 2016. <https://en.wikipedia.org/wiki/Lean_software_development>.

[25] Woods, Dan. "Why Lean And Agile Go Together." Forbes. Forbes Magazine, n.d. Web. 13 Apr. 2016. <http://www.forbes.com/2010/01/11/software-lean-manufacturing-technology-cio-network-agile.html>.

[26] Yodiz. "Incorporating Lean Methodologies in Agile." Yodiz Project Management Blog RSS. N.p., n.d. Web. 13 Apr. 2016. <http://www.yodiz.com/blog/incorporating-lean-methodologies-in-agile/>.

[27] Krishnamurthy, Venkatesh. "Agile World." : Backlog Prioritization Techniques. N.p., n.d. Web. 13 Apr. 2016. <http://agileworld.blogspot.com/2010/06/backlog-prioritization-techniques.html>.

[28] "Kano Model." Wikipedia. Wikimedia Foundation, n.d. Web. 13 Apr. 2016. <https://en.wikipedia.org/wiki/Kano_model>.

[29] Appleton, Brad, Steve Berczuk, and Steve Konieczka. ""Agile" Change Management: From First Principles to Best Practices." CMCrossroads. N.p., n.d. Web. 13 Apr. 2016. <https://www.cmcrossroads.com/article/agile-change-management-first-principles-best-practices?page=0%2C2>.