Vincent Dante-Maniglia February 5, 2017
Software Development I CMPT 220L-201

Agile Development

The Software Development Life Cycle (SDLC) is a framework defining tasks performed at each step in the software development process. This process is used by the software industry to design, develop and test high quality software. The goal of the SDLC is to produce high quality software that meets or exceeds customer expectations and reaches completion within times and cost estimates.[1]The software development industry utilizes general/traditional models as well as modern models of the SDLC. In traditional models, each phase produces deliverables required by the next phase in the life cycle which is considered a predictive approach. This approach begins with requirements specification and then continues with system analysis, system design, implementation, testing, deployment and maintenance. The use of predictive methods depends entirely on the requirement analysis and planning done in the beginning of the cycle. The Agile model, the most popular model used in the industry, is a modern model which uses an adaptive approach. Agile introduce the concept of fast delivery to customers using prototype approach. Agile divides the project into small incremental builds with specific deliverable features that are provided in iterations. Each iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.[2] These teams are fed with objectives not with tasks. Solutions evolve through collaboration between self-organizing, cross functional teams utilizing the appropriate practices for their context.[3]There is a strong emphasis on continuous customer collaboration within the Agile process.

[1] https://www.tutorialspoint.com/sdlc/sdlc quick guide.htm
[2] https://www.tutorialspoint.com/sdlc/sdlc quick guide.htm
[3] https://www.agilealliance.org/agile101/

There are advantages and disadvantages involved in both traditional and modern models of SDLC. Traditional Waterfall and V shaped models are considered simple and easy to use as well as easy to explain to users. The stages and activities are well defined and each phase has specific deliverables, and there is verification and validation of the product in early stages of product development. [4] Waterfall is a poor model for complex and object-oriented projects, long and ongoing projects, and projects where requirements are at a moderate to high risk of changing. A disadvantage of the V-shaped model is there is a lack of flexibility and adjusting scope is difficult and expensive.[5] Prototyping and Iterative and Incremental models: offer improved and increased user involvement, can detect problems earlier, can accommodate some change requests between increments, and these models are more focused on customer value than linear approaches.[6] The incremental model has the following disadvantages: each phase of iteration is rigid and does not overlap other phases; problems may also arise pertaining to system architecture since all requirements are not gathered up front for the entire software cycle.[7] According to tutorialspoint.com, the following list highlights the advantages and disadvantages of the Agile Model:

Pros

- Is a very realistic approach to software development
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.

Suitable for fixed or changing requirements

- Delivers early partial working solutions.
- Good model for environments that change steadily.

[4] https://melsatar.wordpress.com/2012/03/15/software-development-life-cycle-models-and-methodologies/
[5] http://codebetter.com/raymondlewallen/2005/07/13/software-development-life-cycle-models/
[6] https://melsatar.wordpress.com/2012/03/15/software-development-life-cycle-models-and-methodologies/
[7] http://codebetter.com/raymondlewallen/2005/07/13/software-development-life-cycle-models/

- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required
- Easy to manage
- Gives flexibility to developers

Cons

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

Given the above, I would at this time select a traditional model of SDLC for the following reasons:

a) Simplicity of use

b) Since detailed specifications are necessary at the beginning of the process, the developers begin with a good understanding of customer's requirements and the overall scope of the project

c) Ability to be involved in all aspects of the development process rather than as a part of a cross-functional team. As a young developer, it might be a better way to learn about each of the stages in the process, rather than just be involved with a part of the project.

d) Each phase of the project builds upon the previous one, so successful testing must take place before proceeding, which probably will result in less system error down the line.

e) The project manager would not have to be an Agile or Scrum master.

For a new software developer, the next logical progression after learning, understanding, and successfully executing projects through the use of traditional models, would be to embrace and utilize the Agile Model in order to develop projects that would best benefit from a collaborative approach as a way to overcome the limitations of the traditional model process.

Works Cited

Allen, Raymond Lew. "Software Development Life Cycle Models." CodeBetterCom. Wordpress, 13 July 2005. Web. 05 Feb. 2017.

Sami, Mohamed. "Software Development Life Cycle Models and Methodologies." Mohamed Sami. Wordpress, 06 Dec. 2016. Web. 05 Feb. 2017.

Tutorialspoint.com. "SDLC - Quick Guide." Www.tutorialspoint.com. N.p., n.d. Web. 04 Feb. 2017.

"What Is Agile Software Development?" Agile Alliance. Agile Alliance, 07 Nov. 2016. Web. 05 Feb. 2017.