TRADITIONAL VS. AGILE SOFTWARE DEVELOPMENT METHOD: WHAT TO CHOOSE?

Introduction

Whenever someone is taking on a new development project, they first have to figure out which development method they want to use and implement their product with. Figuring out which development method is best suited for their project is a necessary as it optimizes each step of the process which results in a product of higher quality, lower cost and lesser development time.

However, even though there are plenty of development methods to choose between, it is important to note that; *"The stages that all development project go through are very similar, regardless of the methodology, techniques or tools that are used. What changes greatly from project to project is the way what we approach each of the stages, the precise tasks that we choose to carry out and how we sequence them." (Weaver, 2004, p.52)*:

One of the most important procedures of any development process is the "Software development life cycle" (SDLC). The SDLC is a popular practise used by organizations for designing and developing high quality software applications. It defines how the development work is organized, by acting like a framework which holds some specific tasks which has to be achieved during each phase of the software development progress. A typical SDLC consists of the following phases:

- Planning
- Defining
- Designing
- Building
- Testing
- Deployment

At the moment, there are two main SDLC methodologies which are utilized by most system developers; the *traditional* development method and the *agile* development method.

Traditional development method

Traditional development methodologies like the waterfall method, spiral and incremental model are classified as heavyweight methodologies. These methodologies are based on a sequential series of phases like requirements definition, solution building, testing and deployment. As a consequence of these phases being linear, each phase must be fully completed in order to move onto the next phase.

Heavyweight methodologies require a stable set of requirements which has to be well documented and well defined before the project start. If any (important) requirements have been forgotten after development start, it can quickly turn costly and time consuming as it often leads to the project having to start over from scratch due to the linear progression of these heavyweight methodologies.

It is therefore important for people using traditional software development methods to follow the set of predetermined documentation and procedures which takes place during development, as well as constantly updating the documentation to get a better overview of what has been done and what lies ahead in further development. The success of a project is determined by knowing and understanding all of the sets of requirements before development begins and to be alert of it as implementing changes during the development lifecycles often can be very problematic. The better job is done during planning, makes it easier to determine the project cost, as well as scheduling time and resources accordingly.

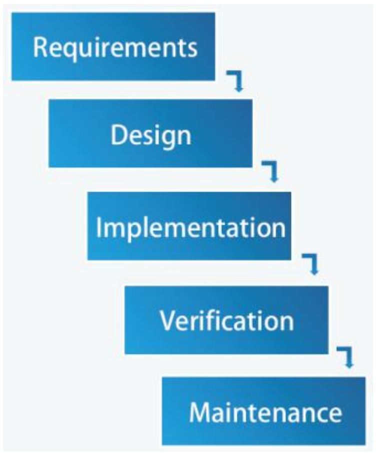
Waterfall model

“The waterfall approach emphasizes on a structured progression between defined phases. Each of these phases consists on a definite set of activities and deliverables that must be accomplished before the following phase can begin. The phases are always named differently but the basic idea is that the first phase tries to capture what the system will do, its system and software requirements, the second phase determines how it will be designed. The third stage is where the developers start writing the code, the fourth phase is the Testing of the system and the final phase is focused on Implementation tasks such as training and heavy documentation.” (Awad, 2005, p. 10*).*

Features

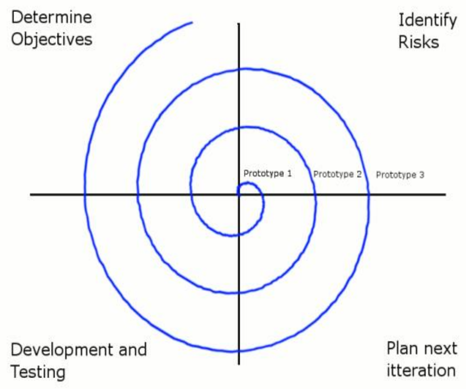
- Carried out sequentially
- Each stage quality assured
- Each stage starts at the agreed baseline
- No stage repeated
- Risk for mismanagement and bureaucracy

- Suited for large, multi-agent projects with need for security
- Suited for projects with complex and a lot of requirements
- Interesting for big industries



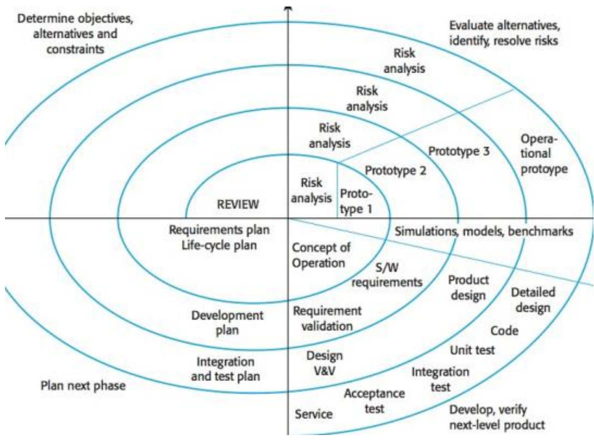
Spiral model

In short summary, the spiral model is the most flexible of the traditional SDLC models and is quite similar to other iterative models in the way it emphasises on repetition. The spiral model goes through steps of planning, design, build and test phases over and over, with gradual improvements at each pass. The spiral starts of in the centre, and for each iteration it keeps on adding, this improves the flexibility of the development process and allows it to add new features later on, and is much more forgiving than the waterfall method.



Incremental model

The incremental build model is a method of software development where the product is designed, implemented and tested incrementally until the product is finished. Each increment provides functionality to the customers, and after the first increment, a core product is delivered and used to receive feedback from the customer. With this feedback, you then add more increments which builds on the customers desired features one bit at a time. The process is continues all the time until the final product is delivered to the customer, and they are satisfied. The increment model combines the elements of the waterfall model with the iterative philosophy of prototyping.



Agile development method

The agile development method focuses on the ability to create and respond to change. It is a way of dealing with and succeeding in uncertainty and a constantly changing environment, and is therefore often used in software development.

What sets the agile development method apart from other development methodologies is the fact that it is more of a mind-set than a method and it is based on the Agile Manifesto and its twelve principles. Agile focuses on the people doing the work and how they work together to find eventual solutions through collaboration between self-organized cross functional teams by utilizing the appropriate practises and frameworks.

Most agile development methods breaks its product development into small increments to minimize the amount of up front planning, design and workloads. These iterations are often known as sprints in agile, where each sprint consists of a set of tasks and requirements which has to be done within a time frame which typically consists of one to four weeks. Working on these sprints are functional teams which are working in all the functions such as: analysis, planning, design, coding, testing and acceptance testing.

Being adjustable increases user satisfaction, by tailoring to the constant changes or additions which fits the customer's needs. Scrum and Kanban are two of the most widely used agile methodologies, but we also have other agile methodologies such as; Extreme programming, crystal, dynamic systems development method and more.

According to Agile Manifesto, the four values of agile are:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.
-

Scrum

Scrum is a lightweight, iterative and incremental framework for managing complex work. It is designed to help teams develop software faster and learn more over time. Scrum consists of small teams breaking tasks down into smaller components known as sprints, to better help rapidly integrate user feedback into a product as well as distributing the workload amongst each other. Each sprints represents a set of requirements and tasks that has to be completed within a deadline. A sprint is typically worked on for one to four weeks, before it is handed over to the clients for them to review.

At the end of each sprint a working product (early on can be demo) is presented to the stakeholder, who then gives feedback on eventual features they want added later on. By presenting the product each iteration, it minimizes risk and allows the product to be adapted swiftly to eventual changes. The goal is to have an available release each iteration with as few bugs as possible which can be presented to the customers. Larger features take longer time and might need multiple iterations to be completed. One of the parts of scrum is realizing that the client will often change their minds about what they want or need, and it is therefore important to be ready to face unpredictable changes, for which a traditional approach is not suited.

When Daily Scrum meetings take place, it helps foster a collaborative approach across the team and makes sure there is a continuous feedback cycle throughout the team. To better divide work and making sure everyone is doing the part, scrum has its own roles known as the scrum master, product owner and scrum team.

The **scrum master** is responsible for setting up the team and the meetings whilst also removing any obstacles to the development progress. **The product owner** focuses on creating a product backlog, and prioritizes the bat log whilst being responsible for the delivery of the functionality at each iteration. **The scrum team** manages its own work and organizes the work to complete the sprint cycle.

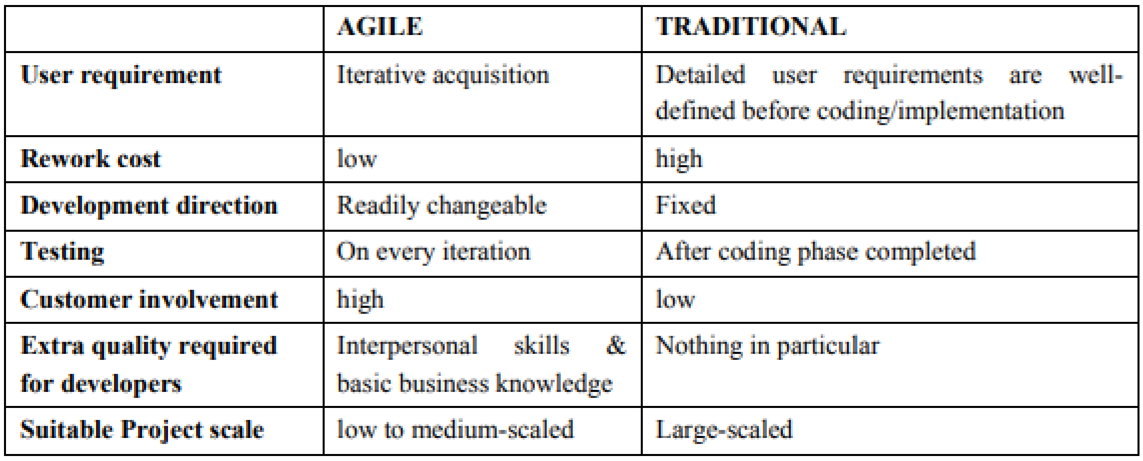
Kanban

Kanban is a Japanese term that means "card", and the development method is about putting cards that contains details on how to complete a request within a clearly defined process presented on a board. The general Kanban board consists of the following three columns: To Do, In Progress and Done. The cards flow through this board to visualize the status of each card and which process they are currently in. Each column has explicit requirements to ensure that the cards has completed each step in the process, and once a card has reached the last column, the content card will be 100% ready for delivery or flows over to another board to continue this process until its finally completed.

Pros and Cons

Before discussing which development method to choose between, it would be helpful to first go through some of the pros and cons of each methodology. (Source table 1: (Leau, 2012, p.4)**).**

Table. 1: Comparison of Agile and Traditional Approaches



	AGILE	**TRADITIONAL**
User requirement	Iterative acquisition	Detailed user requirements are well-defined before coding/implementation
Rework cost	low	high
Development direction	Readily changeable	Fixed
Testing	On every iteration	After coding phase completed
Customer involvement	high	low
Extra quality required for developers	Interpersonal skills & basic business knowledge	Nothing in particular
Suitable Project scale	low to medium-scaled	Large-scaled

Which development method to choose.

Both the traditional and agile software development methods have their advantages and disadvantages. The most important part of choosing a development method is figuring out which SDLC best fits the project and the team working on it. Finding and adopting the best suited SDLC for a project ensures that the organization maximize their chance to deliver their software successfully.

Selecting and adopting the right SDLC has long term implications, and it is therefore crucial that the team study the differences, advantages and disadvantages of each SDLC before making a decision. As any setbacks can be time-consuming and expensive, it is important to be thorough throughout the planning stages, because it is more forgiving making mistakes early on than later.

By figuring out the; size of the project, size and complexity of the software, business strategy, engineering capability we can use this information gained and apply it with the previous gained knowledge on the differences, advantages and disadvantages of each SDLC to make a calculated choice on what system development life cycle is best suited for the project and team.

Due to the iterative nature of the agile SDLC it often exceeds the traditional SDLC. I would argue that agile is better suited for small-medium scaled project development and projects that rely heavily on customer involvement, whilst it is better to adopt traditional SDLC for large-scale projects and projects which contain sensitive information and has to be working 100% bug free.

The agile software development methods were developed to provide better customer satisfaction, shorten the development life cycle, reduce bug rates and to accommodate changing business requirement during the development process. Because customer requirements are acquired iteratively, agile development is able to deliver an end-product that better meets customer needs. This is because each iteration is presented to the customer for review and feedback and this cycle only stops when the customer is satisfied. The customer satisfaction results in recurring customers. A short development life cycle and working in smaller teams ensures that costs are kept low and profit margins remain high.
This methodology also suits start-ups really well. It enables them to publish a complete functional software application to be released faster, and later improved upon. This can help visualize the projects ideas and values to investors and earn money. Low costs and the ability to change their product to best suit customer needs and the market increases the project's chance of success.

Traditional SDLC would fit large scale projects the best as these types of projects often contain heavy workload which to be completed requires a lot of people working on the project. In the traditional SDLC, development teams often hold meetings with the stakeholders and obtain every detailed requirement during the early stages of development process. Then the development teams would start the design phase, followed by the actual coding phase. The testing phase will only start when the entire coding process is completed. Then only will the end-product be presented to stakeholders after there is no issue arises in the testing phase. This linear approach ensures that every important requirement is added and thoroughly tested. This means that the end result will be more stable and it is very suitable for departments like hospitals, law enforcement and military who rely on stable, secure and well-built programs.

One of the dangers of agile is the constant updates and improvements, it can lead to weaknesses, faults and backdoors to occur during and after updates if not toughly checked and tested by the developers before release. By not having to deal with this in a traditional development process, it prevents eventual weaknesses from occurring and remains stable and secure.

Sources:

Awad, M. (2005) *A comparison between agile and traditional software development methodologies*: http://pds10.egloos.com/pds/200808/13/85/A_comparision_between_Agile_and_Traditional_SW_development_methodologies.pdf

Leau, Y., Loo, W., Tham, W., Tan, S. (2012*) Software development life cycle AGILE vs Traditional Approaches*: https://pdfs.semanticscholar.org/69b1/9ddc8a578f4c63d1dfe15252a465ee12fe5d.pdf

Mavuru, I. (15.12.2019) *Traditional vs Agile software development methodologies*: https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies

Sciodev. (15.12.2019) *Traditional vs. Agile Software Development Method: Which One is Right for Your Project?:* https://sciodev.com/blog/traditional-agile-software-development-method/

Tutorialsonpoin (15.12.2019) *SDLC – Overview*: https://www.tutorialspoint.com/sdlc/sdlc_quick_guide.htm

Weaver, P. (2004) *"Success In Your Project, A Guide To Student System Development Projects"* Essex: Pearson Educated Limited

W3schools. (15.12.2019) *SDLC Waterfall Model*: https://www.w3schools.in/sdlc-tutorial/waterfall-model/