Metropolia

Nicolas Viera

Agile Methodologies

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor’s Thesis

16 February 2023

Abstract

Author:	Nicolas Viera
Title:	Agile Methodologies
Number of Pages:	38 pages
Date:	16 February 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Smart IoT Systems
Supervisors:	Antti Piironen (Principal Lecturer)
	Anne Pajala (Senior Lecturer)

Once the waterfall method became obsolete, programmers wanted to develop a new way to improve the work of software development teams. The waterfall method was taking teams too long to deliver products. Sometimes making the product was outdated and impractical by the time of its release because of the constant growth of the software development market.

This contributes to the goal of the thesis, which is to study this new way of software development called Agile. This study will explore the ways of Agile, the history, the Manifesto, values, principles, Pareto Principle and the different variants of Agile that were created for specific needs. These methodologies follow certain criteria and workflow. Furthermore, this thesis will include the benefits and drawbacks of these Agile methodologies and how they can be used in companies.

One of the main focuses of this thesis is the Agile methodology called Scrum. Scrum is the most popular of all frameworks and is being endlessly improved as more teams use it. Towards the end of this thesis, Scrum implementation will be defined. The roles of the Product Owner, Scrum Master, and development team are thoroughly explained. Also, Scrum Events and Scrum Artifacts and how they are imperative for Scrum is discussed.

Keywords: Waterfall, Software Development, Agile, Manifesto, Pareto Principle, Methodology, Scrum, Product Owner, Scrum Master, Development Team, Scrum Events, Scrum Artifacts

Contents

List of Abbreviations

XP:	Extreme Programming
DSDM:	Dynamic System Development Method
UAT:	User Acceptance Test
IT:	Information Technology
DevOps:	Development Operations
WIP:	Work in progress
LSD:	Lean Software Development
TPS:	Toyota Production System
IBM:	International Business Machine Corporation
PO:	Product Owner
SM:	Scrum Master
DoD:	Definition of Done

1 Introduction

Competition on the software development market is currently booming and businesses must make sure that their products are of the highest quality, delivered in a timely manner, and updated for the markets ongoing demand. Businesses frequently deliver products that are no longer useful at the time of their release and the project would either be scrapped or in need of refinement, and, therefore, wasting time and money. This can be prevented if the customer was more involved in the project and businesses adopted the Agile method [1].

Adapting to Agile methods allow businesses to be more customer-focused, flexible and able to produce better results, including in development, management, logistics, human resources, and delivery. For instance, during meetings, teams may swiftly adjust to requirement changes without having a detrimental influence on release schedules. Agile lowers costs, raises customer gratification, produces products of superior quality, increases alignment with the customer and overall business value [1].

The goal of this thesis is to explore and analyse Agile. Additionally, the most used methodologies which include Kanban, Extreme programming, Lean, Crystal and Scrum. Also, the benefits and drawbacks of these Agile methods as well as a brief history of software development concerning Agile predecessors, the waterfall and prototype models are discussed. Moreover, the Agile Manifesto along with its four values and twelve principles, as well as, how the Pareto Principle fits into Agile are explained and analysed. Finally, the objective is to thoroughly explain Scrum and how it is implemented as Scrum is the most widely used Agile methodology [1].

2 Software Development History

The world’s first piece of software was created by computer scientist Tom Kilburn and executed at the University of Manchester in England in June of

1948. Kilburn and his colleague Freddie Williams had built the Manchester Small-Scale Experimental Machine which is also known as the “Baby”. The Baby was the world’s first computer with storage that was able to read and write data reliably [2]. This marked the beginning of software development, but it was not until the 1970s that the waterfall and prototype models were introduced [3][6].

2.1 Waterfall Model

The waterfall model is a step-by-step approach to management methodology that mimics a waterfall in its trajectory, as shown in figure 1. This method was founded by Winston W. Royce in 1970 and it is considered the oldest software development practice [3].

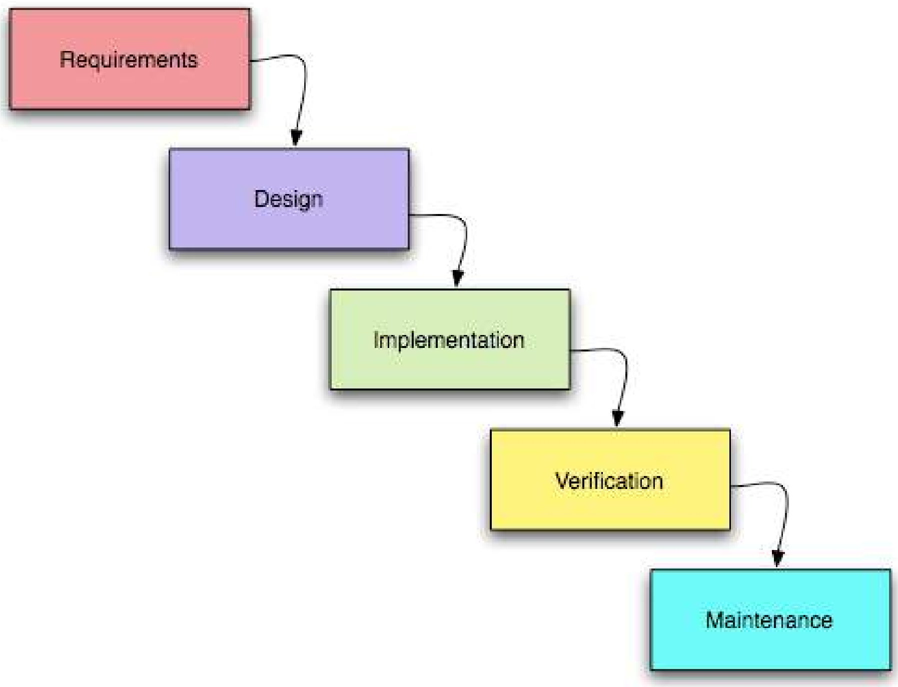


Figure 1. Step-by-step model of the waterfall methodology [3].

The process starts with the requirements phase, where the project manager gathers a detailed understanding of the customers’ requirements for the product. This information is then captured in a product requirement document which consists of title, author, purpose, scope, stakeholder identification, product overview, technical requirements, high level workflow plans and performance metrics [4]. The next phase involves the design. This is broken up into the logical design and the physical design. The data gathered during the requirements phase are used to design the system independently of any hardware or software system. Once the higher-level logical design is finished, the system specialists can start to convert it into a physical design in accordance with the requirements of the customer [4]. Afterwards, it is the responsibility of the programmers to code the applications during the implementation phase, using the project requirements and specifications as a guide [5]. During the verification phase, the quality assurance team evaluates the product to make sure it complies with the required specifications and if it needs debugging. If major faults or bugs are found, then the project moves back one step to the implementation phase. The team can utilize UAT (user acceptance test) to verify customer satisfaction and move on to the next step [5]. Finally, the maintenance phase, where the product is in use by the customer and if problems are found then changes are made. This phase is ongoing until the product is no more or for as long as the contract dictates [5].

The waterfall model was used for many years and is still used to this day. It follows a chronological model, has strong documentation and does not have customer involvement which provides budget clarity, estimated project completion and easier on new employees. In the end, costs may be higher due to customer dissatisfaction and project plans lack flexibility due to the nature of waterfall [5].

2.2 Prototype Model

Since the early 1970s, prototyping methods have been used to combat the waterfall method disadvantages of delivering the project at the end of its development. The prototype model is the development of software or product that is a functional replica of the desired product. As a result, the customer can provide feedback early on and see whether deadlines for the project can be successfully met. The prototype may be missing certain capabilities, not reliable and even be extremely inefficient but this is intended as it is used to evaluate designs based on customer needs. This model was and is mainly used to give the customer a very basic idea of what to expect. Prototype designs are done until the customer is satisfied with the result [6]. The process of this model is shown below in figure 2.

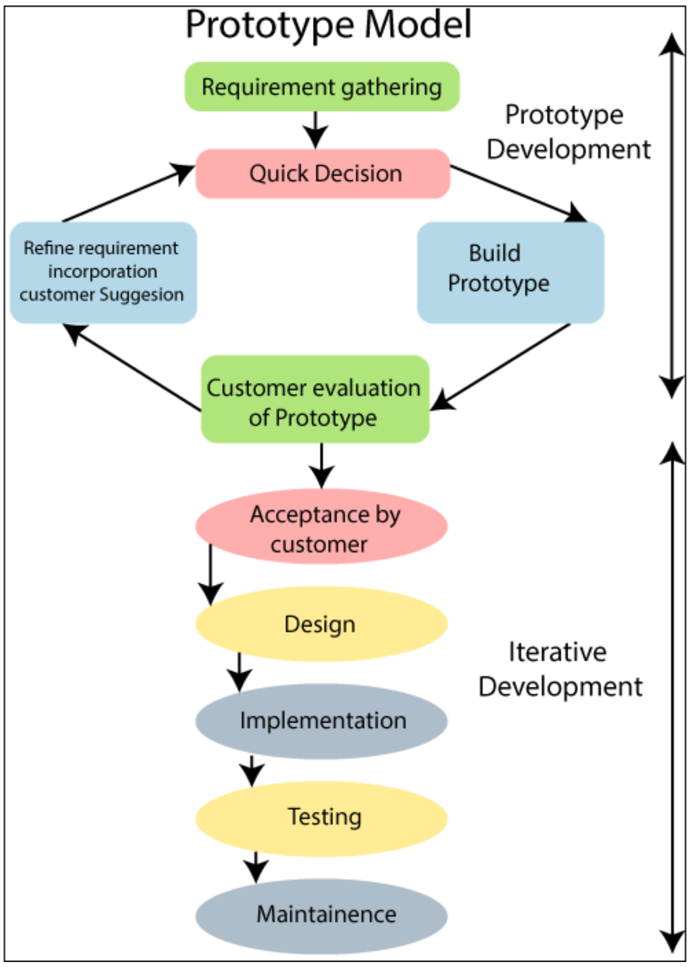


Figure 2. Prototype Model. [6]

The first stage requires comprehending the fundamental specifications of the product, particularly regarding the user requirements. This can be done by interviewing those that are going to be using the product [6]. The second phase involves designing a preliminary and basic proposal. It is primarily intended to provide a brief summary of the product; however, it is not the complete design [7]. The third phase involves the actual prototype building where some features may not work as intended but even so, it is to show a small working model with the basic requirements and customer needs [6]. The next phase is where the prototype is reviewed by the client and other business stakeholders of the project. This is the phase where information and productive remarks are gathered in order to further improve the prototype. This helps find the strengths and weaknesses of the prototype [6]. The final step is to implement the product and maintain it. The product is thoroughly tested and then deployed to production while going through continuous maintenance to minimize disruptions and prevent large-scale failure [7].

3 Agile

The Agile methodology is a strategy to manage projects by segmenting them into several phases. Continuous improvement is needed at every level, as well as ongoing cooperation with stakeholders. The customer is also involved in the development of the project in order to upkeep customer satisfaction. Due to its adaptability to change, flexibility and high level of customer involvement, using the agile method has been one of the most popular approaches when it comes to project management and software development [9].

According to statistics, at least 71% of United States companies use Agile and 64% of Agile projects are successful compared to the 49% waterfall projects have [12]. Doing the mathematics, Agile projects are 150% more successful than waterfall projects. After transforming to Agile, an average of 60% profit and revenue have been reported by these companies. The top five reasons they gave to adopt Agile were faster delivery, enhance the ability to manage priorities, increase productivity, improve business and IT alignment and enhance software quality [12].

3.1 Manifesto

A group of seventeen experienced software developers from Extreme Programming, Scrum, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming and other types of software development got together in 2001 to discuss the need for an alternative to tedious, documentation-driven software development practices and being locked to the waterfall development process. As a result, what they came up with was the Agile Manifesto [8].

The Manifesto is about the four values and twelve principles of Agile. It serves as a declaration that seeks to enhance software development methodologies and directly addresses the inefficiencies of conventional development procedures. The representatives wanted a way to not rely on heavy documentation and minimize oversight on critical information [8].

3.1.1 Values of Agile

Individuals and interactions over process and tools, is the first value of the Agile Manifesto. This suggests that the people behind the processes and tools are more important than the latter. Software teams used to care more about having the best possible tools and processes rather than building the right group of individuals who can collaborate and solve any problem that arises [10].

Working software over comprehensive documentations, is the second value of the Agile Manifesto. The Agile methodology values having working software for their customers and before the Agile Manifesto, software developers would spend a lot of their focus on working on documentation. This was never a bad thing; however, they would spend an excessive amount of time working on documents before providing any work on actual software [10].

Customer collaboration over contract negotiation, is the third value of the Agile Manifesto. One of the most important and key features of the Agile methodology is that the customer is constantly involved in the project. Before Agile, contracts would be done and signed with the requirements of the customer but then the final product would turn out to not be what was expected. With Agile, the customer is always involved in an endless loop of giving feedback to ensure that the product is what they need [11].

The final value of the Agile Manifesto is responding to change over following a plan. With Agile, you can adapt to change, new customer demands, and the ever-changing market because Agile does not follow a static roadmap where you follow everything to detail and which has no room for change [11].

3.1.2 Principals of Agile

Below are the twelve principals of Agile taken from the Agile Manifesto official website Agilemanfiesto.org [8].

> Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
>
> Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
>
> Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
>
> Business people and developers must work together daily throughout the project.
>
> Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
>
> The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
>
> Working software is the primary measure of progress.

> Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
>
> Continuous attention to technical excellence and good design enhances agility.
>
> Simplicity--the art of maximizing the amount of work not done--is essential.
>
> The best architectures, requirements, and designs emerge from self-organizing teams.
>
> At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly. [8]

These are twelves principals of Agile that define a culture that change is always welcome and continuous customer involvement and feedback is part of the work. Being able to consistently deliver new releases while gathering feedback from the customer is much better than delivering a release all at once and having to deal with a massive number of bugs and customer requests [8].

Having an Agile team with people that can hold their own and giving them the support or resources, they need removes the need for micromanagement and you can trust that the work will be of highest quality. Agile also ensures a good work-life balance to maintain a high morale in the workplace. A maximum of 7.5 hours a day should be allotted to working hours [11].

3.2 Pareto Principle

The Pareto Principle, also known as the 80-20 rule or Pareto Rule, was devised by an economist named Vilfredo Pareto. Pareto’s research specified that 80% of the consequences originate from 20% of the causes, which concluded that the correlation from inputs and outputs are not identical. His research was based on Italy, where he concluded that 20% of the population owned about 80% of the land and after checking into other countries, he devised that the same applied to them. Pareto derived that this rule can be applied to work environments as well [25]. For instance, to software development because statistics show that only 20% of a software’s features are used and the rest are left to more professional users that need them for specific tasks [26].

In Agile, the team will give 80% of their time on the 20% of the most vital parts of the project. This gives the customer more value out of the product because they can demo the most significant features sooner than expected. The team can identify which features will bring the most value by locating which one is causing the most questions or doubts from customers. The customer can also provide which features would be the most critical to add to the next iteration. Having the 20% delivered faster also gives the developers more leeway with the 80% because those features will not be as vital to the customer and users. It is also worth mentioning that this rule is not followed by every Agile team [26].

3.3 Agile Methods

Agile methodologies which this thesis include are Kanban, Extreme Programming, Lean, Crystal and Scrum [13]. They all follow the Agile philosophy but are all different in the way they aim to work. Choosing the right methodology will depend on how the Agile team wishes to produce and deliver their products.

3.3.1 Kanban

Kanban is an Agile methodology that is broadly known as one of the simplest frameworks because it allows project managers to efficiently manage and keep track of their projects. Kanban was initially developed in the late 1940s by a Toyota industrial engineer called Taiichi Ohno [19]. His aim was to create a simple planning system to control and manage work and inventory during every project stage. Toyota created a flexible and efficient production control system with Kanban, which enhanced productivity while eliminating expensive materials and partially completed products that never got to be delivered [19].

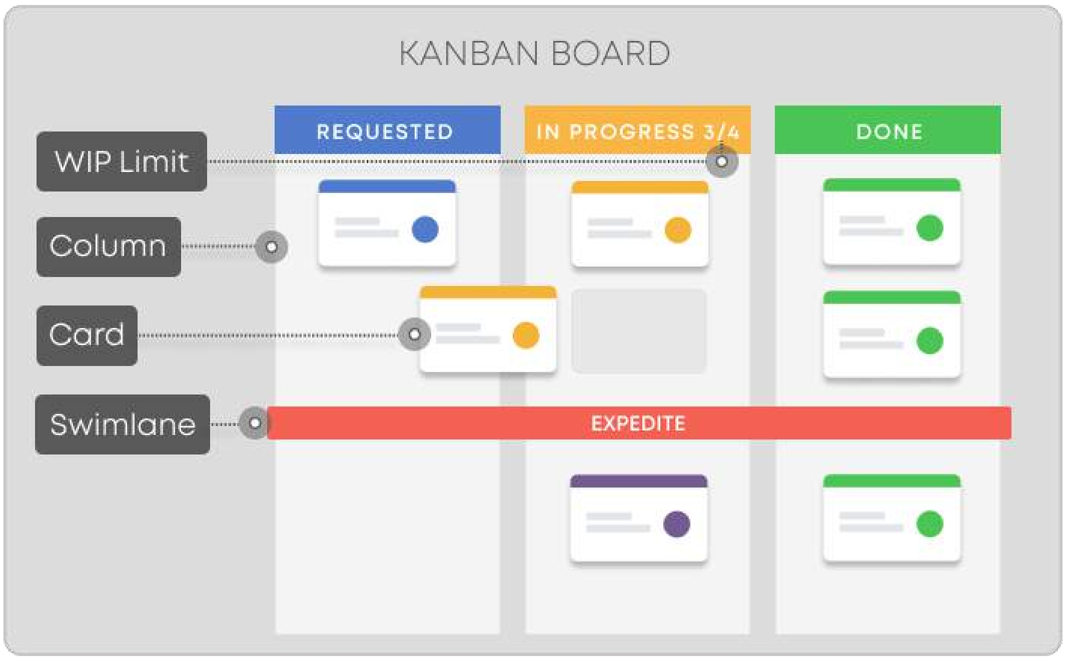


Figure 3. The Kanban Board [19].

Kanban boards, as shown in figure 3, are used to represent workflow allowing the team to optimize work delivery across several teams and handle intricate projects. Kanban's focus is to be able to visualize workflow, limit work-in-progress, and maximize efficiency. This is all planned in the Kanban Board, it helps Kanban teams picture their daily workload.

The Kanban Board consists of cards which are a visual representation of the projects. Typically, one project or work item per card. Columns represent the different stages on workflow, this includes "to-do", "in progress" or "complete" and this goes on until the completion of the column. Work-in-Progress limits are the limits on the number of cards that can be in a column. For example, if the WIP limit is 5 then there cannot be more than 5 cards in that column. If such a thing happens that there is more than the limit, then the team needs to work on that column to move them forward. Swim lanes are used to visually separate workflow items of different types on the same board. The commitment point is where the team decides to start working on a certain project, whereas the delivery point is where the work is considered done. The Kanban board allows the use of a pull system where work items are pulled into the queue only when they are requested and in prioritized order [19].

Much like other Agile methodologies, Kanban is flexible, offers continuous delivery, increases productivity, and effectiveness. But what separates it from other methodologies is the Kanban board and that it is a simple and easy to understand system which can be applied by any company [20]. However, for Kanban to work, the board always needs to be updated or the team will start to work on items that no longer needs work. Additionally, the board can sometimes become too difficult for some team members, and it can be flooded with overcomplicated cards. There are also no direct timeframes for the workflow which can disorientate team members and cause stress [19].

To tackle these problems with Kanban along with Scrum, Scrumban was developed. Scrumban is a hybrid Agile approach which combines the structure and routines of Scrum and visual work of Kanban. Scrumban takes the best from Scrum which is planning, reviews, retrospectives, prioritisation, and identifying how much work can be put into a Sprint. From Kanban, it takes pulling items when the team can, no specific roles in the team, flow charts, and the Kanban board [23].

3.3.2 Extreme Programming

Extreme programming (XP) was developed by Kent Beck back in 1996 to find a better way of doing software development. XP follows the same principles as Agile, where teams are producing high quality software and can adapt to changing requirements. But what makes XP different is that it focuses more on how engineers work and delivering higher quality code. XP also follows a set of five values which are communication, simplicity, feedback, courage and respect. It also follows the twelve principles similar to the Agile Manifesto principles [22].

Communication is always key when working within a team. People can learn from each other and help others when there is a problem they can solve. This saves time and money. Simplicity, as the name suggest, is keeping things simple. Only doing what is necessary and not going above and beyond without needing to. Feedback is always important to product better results. Without feedback, the team will be continuing working without refined requirements to customer needs. Gathering opinions from team members and customers are a must. Courage is needed to stop working on something that is not going work and to act on feedback even when it is difficult to accept. Kent Beck defines courage as an “effective action in the face of fear” [22]. Respect is one of the most vital values of XP, people in team need to respect each other to be able to work well together and collaborate in a healthy setting. There is no gain from disrespecting one another. This will only cause distraught within the team and hinder project continuation [21].

The twelves principles are as follows:

- The Planning Game.
- Small Releases.
- Metaphor.
- Simple Design.
- Testing.
- Refactoring.
- Pair Programming.
- Collective Ownership.
- Continuous Integration.
- 40-hour week.
- On-site Customer.
- Coding Standard.

With these values and principles set, XP bring many benefits to development teams. Organized programming which brings fewer mistakes and creates less bugs. This is done through pair programming which is two programmers sharing a workstation. One person is the driver, who is actively on the computer and the navigator who is showing the directing the driver. This makes producing code very effective as well as more fun and engaging. Moreover, it is also expected for the pair to swap roles every now and then. There is more programmer and client satisfaction because the customer has control. The customer makes the business decisions and provides what features are to be added as well as their acceptance criteria. Additionally, the customer will always be directly involved in the project as part of the team [22].

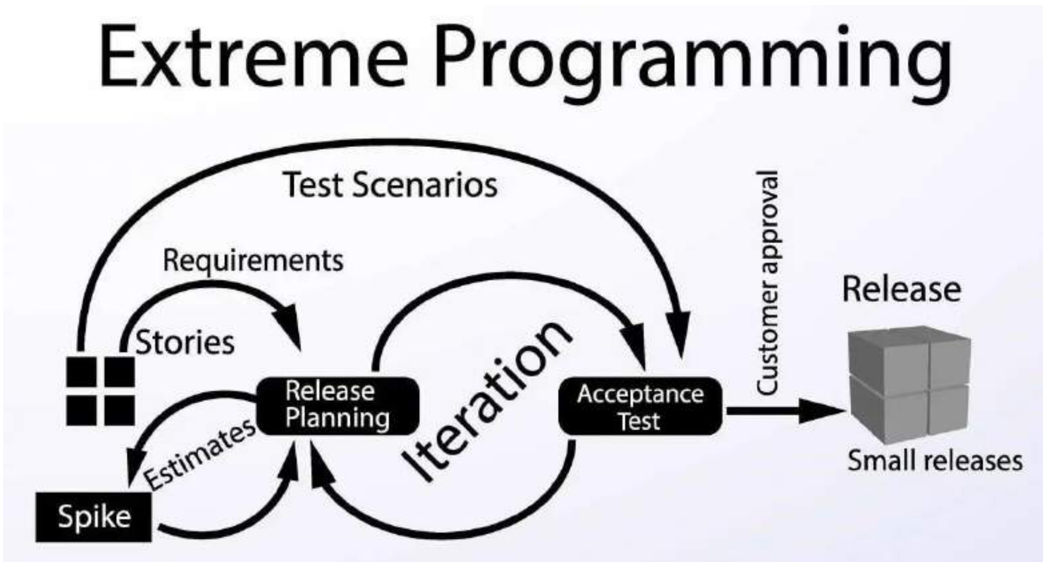


Figure 4. Extreme Programming Cycle. [24]

Similarly, to other methodologies, XP has user stories which are short descriptions of the customers' wants and needs for the project as shown in figure 4. This is also where the requirements for the stories are gathered. Furthermore, there are iterations in XP which are weekly cycles. These are held at the start of the week to check on the team's progress and for the customer to select which stories are to be delivered on that week. Ultimately, the goal is to have these stories produce running tested features by the end of the week [24].

3.3.3 Lean

Mary Poppendieck and Tom Poppendieck are the creators of Lean Software Development (LSD). They wrote a book in 2003 called "Lean Software

Development: An Agile Toolkit" and provided the readers with Lean manufacturing principles as well as comparison to traditional Agile methods. Which led to them combining and transforming these ideologies into their own method. LSD originates from the Toyota Production System (TPS). TPS was designed to remove irregularities in production and manufacturing to remove waste as well as to create a process that can deliver required results smoothly [30].

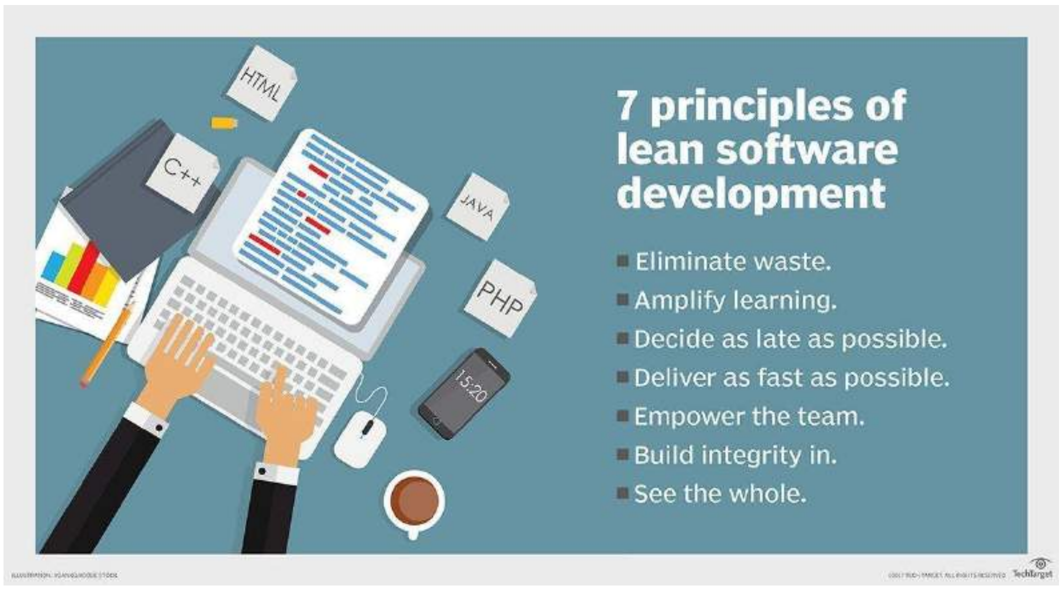


Figure 5. 7 principles of lean software development [30].

Figure 5 shows the 7 LSD principles which are mentioned on Mary's and Tom's book. These principles are derived from TPS's principles which are waste of overproduction, waste of time on hand, waste of transportation, waste of processing itself, waste of excess inventory, waste of movement, waste of making defective products and waste of underutilized workers. The difference is that TPS principles were meant for manufacturing companies and LSD principles are for software and engineering [29].

To be able to deliver more value to the customer and improve satisfaction, eliminating waste is key. By eliminating waste, you increase efficiency, save time and money. Removing unnecessary code or features that do not provide any worth to the customer or users can help improve feedback loops and minimize the development processes. Disregarding unclear requirements result in having less frustration and quality issues [29].

Amplified learning is about creating knowledge within the team and allowing every developer to learn from each other. This can be done by pair programming as explained previously in the Extreme Programming chapter. Furthermore, team members can hold code reviews, create documentation, and set up meetings to share knowledge. To achieve the best results possible, decisions are best to be made as late as possible. This helps clear uncertainties and allows team members to make decisions based on facts and not assumptions. As a result, this prevents having to start over or make changes to the project [30].

Just like every other Agile approach, LSD heavily emphasizes on delivering as fast as possible. With the current fast and ever-changing technology market, this approach is seen as indispensable to software development teams. Empowering the team by respecting and communication with each other is another LSD principle. Teams need to focus on understanding and respecting one another because otherwise conflicts will arise which will result in an adverse workplace. Managers should be professionals who are able to listen, act and provide valuable suggestions to other members of the team [29].

The customer should be providing a constant flow of information to the developers and the same should be done from developers to the customer because this builds integrity. The customer then has an idea of how the product will work and can already experience the system or interface. Optimizing the whole, allows the team to examine the project from the beginning to the end and lets them take apart tasks that may be challenging into smaller or easier tasks. This helps teams to understand how much they can deliver, therefore, preventing unstable software. Overall, Lean Software Development is customer oriented, highly adaptable, very flexible, and minimizes waste. However, having professionals in an LSD team is essential because they would need to follow all the values and principles of Lean [30].

3.3.4 Crystal

Alistair Cockburn, an American computer scientist working at the International Business Machine Corporation also known as IBM, research many different types of software development teams that did not follow official methodologies. However, these teams ended up with fully successful projects. This is how he came up with the Crystal methods which focus on team members, interaction, skills and communication [32].

The Crystal framework adopted seven principles where the first three are mandatory but the last four are optional and should be implemented if required. The first three are frequent delivery, reflective improvement and osmotic communication. Osmotic communication refers to having the whole team in the same location because this allows team members to easily pass around information. This can become a problem if there were to be team members who work abroad or remote and wish to communicate with the team. The last four principles are personal safety, focus on work, access to subject matter experts and users, and finally technical tooling. Technical tooling in a sense means automation. This means that faults and mistakes would be found automatically [31].

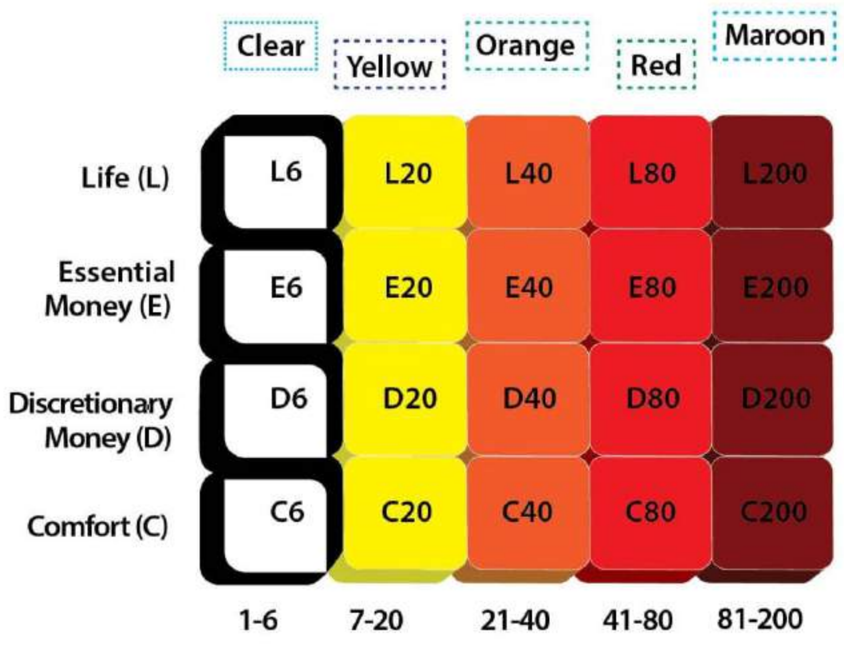


Figure 6. Methods of Crystal Family [33].

Crystal is a family of many variant approaches such as Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, Crystal Maroon, Crystal Diamond, and Crystal Sapphire as shown on figure 6. Crystal Diamond and Crystal Sapphire are used for dangerous and large-scale projects that may put human life in danger. Crystal focuses heavily on the size of project and how important the project is to the company or team. Therefore, Alistair separate the importance of a project to different marks as shown in figure 6. L for life, E for essential money, D for discretionary money and C for comfort. Depending on these two factors, they will determine which Crystal family the project will fall in to. Crystal also has leading roles in software teams which include Executive Sponsor, Executive, Lead Designer, Programmer, The Ambassador User, and Tester [33].

Every Crystal methodology follows an incremental development process flow which can last anywhere from one week to three months. This process involves many practices which are continuously happening throughout the duration of the project. These practices include staging, monitoring, revisions and review, parallelism and flux, holistic delivery strategy, methodology tuning technique, user viewings and reflection workshops [33].

Staging is a planning session of when the next increment will start. A schedule will be mapped out from start to finish of the project and developers will decide on which activities they will work on based on their professional capability. Monitoring is where the team monitors the progress of the project and guarantees that it is going as planned. Monitoring stages are done during project milestones which includes the start, first review, second review, during tests, while delivering, and in stability stages. Revision and review occur during every increment and release where developers involved in activities such as construction, demonstration and testing, as well as reviewing the increments. Developers start to work on the design and coding part of the project. After this is done, testers can begin testing for bugs and mishaps. If nothing is found, then developers can finally start reviewing the product to verify that it fits with the requirements. Parallelism allows teams to work in correlation which involves two or more projects depending on how many teams are involved. Flux is the flow of work, where teams monitor that work is being done and it is stable enough to be presented. Holistic diversity strategy follows the idea that larger teams are to be split up into smaller chunks. This allows smaller teams to be more functional and can be split up by their proficiency [33].

Crystal was created by fine-tuning different software development methodologies into one family and this is how the methodology-tuning technique was founded [32]. Similarly, to how Alistair created Crystal, this technique uses the data from interviews, workshops, and feedback from development teams to fine-tune the Crystal family. During every increment, teams are to gains more knowledge and devise which techniques and tools where most appropriate for Crystal. User viewings are done at least twice for every release of a final product. Depending on if the project is of a larger scale, then user viewings can be increased to a minimum of three. This gives value to the customer and end-users because it increases the chances that the requirements for the project were fulfilled [33].

One of Crystal’s focuses is to adjust projects and methods depending on size and budget of the project to provide a solid idea on which method to use. Reflection workshops are used to brainstorm previous and upcoming projects to determine what was learned and what can be learned. Workshops are also beneficial to come up with strategies that are to be used. Ultimately, this saves time, money, and increases effectiveness.

3.3.5 Scrum

The origins of the Scrum framework date back to 1986, where Takeuchi and Nonaka described how first-class and innovate products are developed in cross-functionals teams in the article “The New New Product Development Game” [16]. The article’s main points about the Scrum were built-in stability, self-organizing project teams, overlapping development phases, “multilearning”, subtle control, and organizational transfer of learning [16].

Scrum is the most popular Agile framework because its flexibility, and it can adapt to any size of team or project. One of the prime benefits to Scrum is that the work is done by the development team concurrently rather than chronologically. Scrum breaks down a large workload into small pieces and turn them into one-to-four-week Sprints. This allows the project to be changed at any given point of the development and allows developers to start developing without having all the questions answered by the customer [17]. With Scrum, instead of having the completed work released at the end of the project, Scrum takes the most important tasks to be done by order of importance. This contributes to the fact that user and customer satisfaction is significantly improved because they receive a usable portion of the project. As a result, the customer can then report faults and give feedback to development about the product. This is a critical part to development because it ensures success [17].

Just like all things in life, nothing is perfect, and Scrum is no exception. Scrum may have the benefit of being able to work in a simultaneous manner, but this also leads to a drawback that there can be a scope creep [18]. This means that there might not be a definite project due date and the team might feel discouraged by that fact. This might also lead to inconsistencies and team members becoming uncooperative or uncommitted. Consequently, the project might end up with delays, not meeting expectations and ending in failure. The Scrum team is also limited to ten people and working on bigger projects could potentially hinder the workforce of organizations [15]. Scrum also requires very experienced personnel who can work intensively and provide professional feedback and reports to the team. Learning and implementing Scrum could prove difficult for organizations because of the need for experienced professionals. Consequently, if a team member was to leave during a project, this could lead to a lot of pushbacks or failure [18].

4 Scrum Implementation

Implementing Scrum is not an easy task because it requires a set of defined roles, principles, and workflow as shown in figure 7.

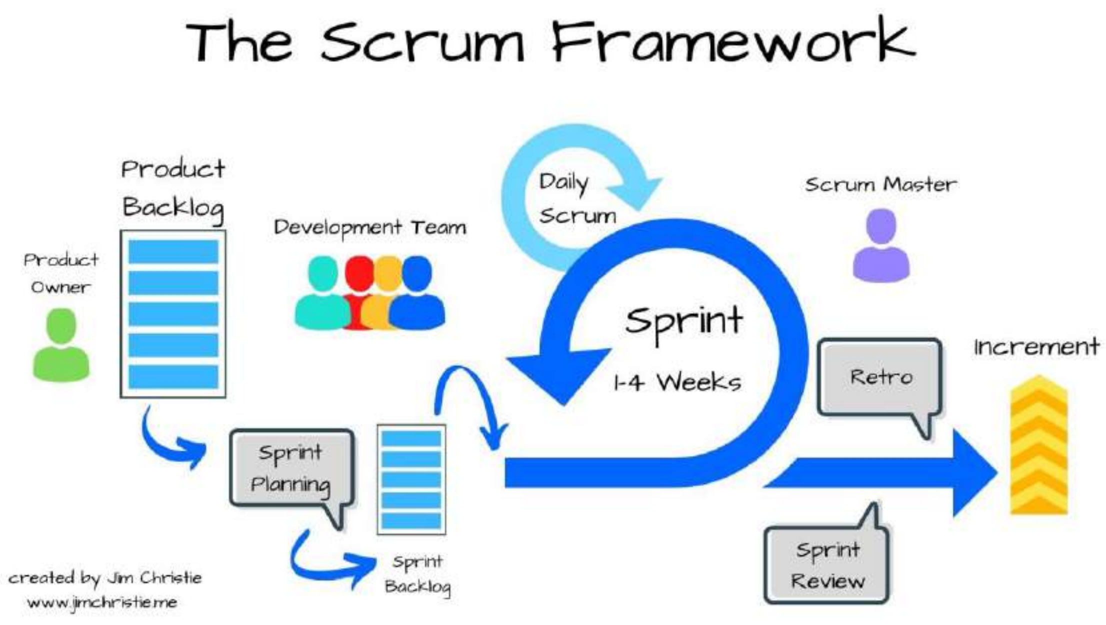


Figure 7. The Scrum Framework [34].

4.1 The Scrum Team

The Scrum team is devised of a Product Owner, Scrum Master and the development team which consists of five to ten individuals. This collective is supposed to work together as a whole to deliver product increments and encourage one another to communicate on a high-level [15].

4.1.1 Product Owner

The product owner (PO) is the person who takes care of business and is representing the needs of the stakeholders and customers to the team. They take those needs and translate them into user stories which are stored in the product backlogs to be arranged depending on customer and stakeholder requirements. The PO is responsible of the product backlog management and uses that to maximizing the value of product by providing the product’s vision and ensuring the best work performance from the Scrum team. They must also get constant feedback on the product and product backlog from the customer and stakeholders this ensures success for the team [35].
Besides managing the product backlog, the PO must have the respect from the Scrum team because this ensures other team members will agree with the decisions the PO makes. Only the PO can change a product backlog item and if another member of the team wishes to make changes, they must get it passed through the PO [35].

4.1.2 Scrum Master

The Scrum master (SM) is the person who makes sure that the team is following the foundation and principles of Scrum. Much like the PO, the SM is accountable for the success of the team which involves having group sessions or one on ones with development team members. The SM is the coach of the team. They help the members be self-organized and be cross-functional. If the development team has some sort of blocker or impediment, they will be the ones to try and assist in overcoming this obstacle. The SM will ensure that team collaboration is fruitful, constructive, and productive [17][35].

The PO can also benefit from the assistance of the SM because they can provide useful techniques for product goal definition and product backlog management. The SM can also help the PO facilitate product planning when there is a more compound environment and an ample amount of backlog items to breakdown. The SM will also be responsible for making sure that the product backlog items were understood by the development team and if some items are misunderstood, the SM will encourage the team members to approach the PO with questions for clarity [35].

To be an effective Scrum Master, there are certification exams that are available and offer certificates that validate a certain amount of knowledge. These include the Profession Scrum Master I which demonstrates fundamental level of Scrum mastery, Professional Scrum Master II which demonstrates an advanced level of Scrum mastery and Professional Scrum Master II which demonstrates a distinguished level of Scrum Mastery [35].

4.1.3 Development Team

The development team is a set of experts who are not defined by titles, instead they go by the name of “developer”. Everyone that is part of the development team can be proficient on one or multiple specific professional areas. They are the ones responsible for planning and creating the Sprint backlog. Developers are also responsible for establishing quality by following the Definition of Done (DoD). DoD is when the product must satisfy all condition made by the user or customer and it is ready to be accepted. The development team members must plan their days to fit the Sprint goal so that they can commit to the stories they chose at the start of the Sprint [35].

4.2 Scrum Events

Scrum events are time-boxed events, or Sprints, which are established in advanced with a max duration of a month. These events include the Sprint planning, daily scrums, Sprint review and the Sprint retrospective [35].

4.2.1 Sprint Planning

The Scrum Master is responsible for holding the Sprint planning meeting and making sure that all required team members are invited and present. The product owner makes sure that all team members are ready to discuss the most important product backlogs and product goal. They will usually send documents to team members to read beforehand or attach the files in a joint storage. Sprint planning is also limited to eight hours or less for a one-month Sprint and if the Sprint is shorter than the planning it will be adjusted accordingly [35].

Sprint planning follows an agenda, and it starts with the Sprint goal. The Sprint goal is a commitment done by the development team because only they can make sure that the goal and Sprint will be manageable. This goal also defines why the Sprint will be beneficial to stakeholders and customers. Once the goal is set then the PO and development team discuss what can be done in the next sprint. The developers select items from the product backlog based on the team's current performance, confidence and capacity. Developers also plan their work in increments that meets the Definition of Done and break down larger tasks into minor tasks which could be done in a day or less. The Sprint goal, backlogs items, and sprint plan altogether are referred to as the Sprint backlog [35].

4.2.2 Daily Scrum

Daily Scrum is setup by the Scrum Master, but it is conducted by the development team. This is a 15-minute meeting to look at the progress of the Sprint goal and Sprint backlog. If the development team is having issues maintaining the 15-minute window, then it is up to the SM to teach the team in how to better plan their sessions. The meeting is held at the same time of everyday of the workweek to promote consistency [15][35].

During the meeting, developers are to plan what they are going to do for the day and answer three questions. What did you do yesterday? What are you going to do today? Do you have any blockers? If the developer has a blocker, then it is the SM’s responsibility to help them remove this impediment. Developers can also set up a meeting sometime after the Daily Scrum for general Sprint discussions or to rearrange Sprint backlog items [35].

Daily Scrum is designed to help build communication in the Scrum Team. This allows for quick and impactful decisions that can facilitate the removal of impediments within the team. These meetings also improve the development team’s knowledge as they can share blockers and the team can discuss how they are to be solved. This allows for future similar blockers to be removed swiftly, without the need of discussion and therefore, increasing efficiency [35].

4.2.3 Sprint Review

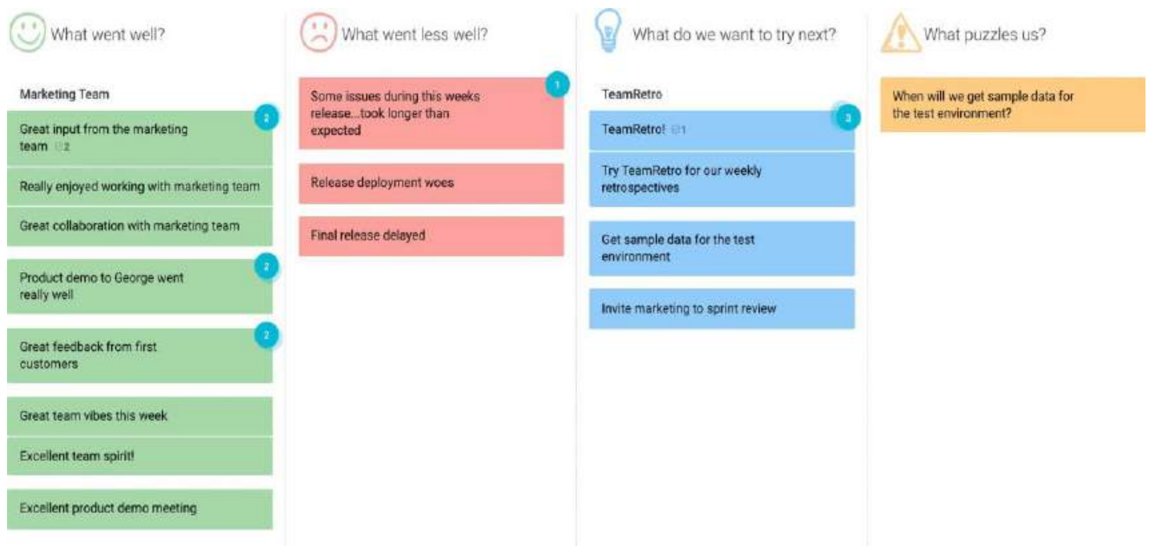
The Sprint Review is arranged by the Scrum Master, but it is demoed by the Product Owner. The PO is also responsible for inviting key stakeholders to the meeting and for showing product goal progress. They are to showcase what was and was not accomplished during the Sprint. The Sprint Review is also set to last a maximum of four hours for a one-month Sprint [35].

During the discussions, developers are to mention any blockers that arose during the Sprint, much like in the Daily Scrum, and how these blockers were resolved. The group can review the current market and determine what would be the most advantageous next step to do for the product. Timeline, budget, team capacity and potential should also be reviewed and updated [35].

The Sprint Review is not meant to be a presentation but instead a discussion for learning. The Scrum Team and the stakeholders collaborate on what to do for the next Sprint which is beneficial for the next Sprint Planning [35].

4.2.4 Sprint Retrospective

The Sprint Retrospective is a meeting which occurs after the Sprint Review and before the next Sprint Planning. This meeting can be facilitated by the Scrum Master or the Product Owner. The purpose of the Sprint Retrospective is to review what went well, what went bad and what can the team improved as shown in figure 8 [35][36].



What went well?

Marketing Team

- Great input from the marketing team
- Really enjoyed working with marketing team
- Great collaboration with marketing team
- Product demo to George went really well
- Great feedback from first customers
- Great team vibes this week
- Excellent team spirit!
- Excellent product demo meeting

What went less well?

- Some issues during this weeks release...took longer than expected
- Release deployment woes
- Final release delayed

What do we want to try next?

TeamRetro

- TeamRetro!
- Try TeamRetro for our weekly retrospectives
- Get sample data for the test environment
- Invite marketing to sprint review

What puzzles us?

- When will we get sample data for the test environment?

Figure 8. Sprint Retrospective session [36].

Pointing out what went well during the Sprint can keep the team motivated, create positive change and improve overall team knowledge by sharing new skills or tools that were used during the Sprint. Stating what went wrong is also essential for the team because it allows the team to reflect and acknowledge the negatives of the Sprint. It is crucial to remember not to play the blame game within the team because as a Scrum Team, everyone is equally responsible for any downfalls that occurred.

Retrospectives are a great way to get the whole team in one platform and be able to communicate freely about the current Sprint. Team members can give constructive criticism, admit mistakes and grow together as a whole.

4.3 Scrum Artifacts

Scrum Artifacts are pieces of information which a Scrum Team and stakeholders use to explain the product being established, activities to create it, and the actions executed throughout the development to achieve transparency. These artifacts are Product Backlogs, Sprint Backlogs and Increments [35].

4.3.1 Product Backlog

The Product Backlog is a list of what is needed to improve the product. In this prioritized list the most beneficial improvement is on the top, depending on its business value. This list is ongoing up until the product is ready, no longer available or not part of a project [15].

The Product Owner is the one that manages the Product Backlog items and oversees defining them to be more precise for the development team. Product Backlog items are usually prioritized by customer urgency, desire of feedback, and by how difficult it would be to implement. The PO does not decide how fast these items are being pulled, instead this is done by the development team depending on team capabilities and current workload in a Sprint. Therefore, it is important for POs to continuously review the backlog and ensure that it is correctly prioritized before the next iteration. This is often referred to as backlog grooming or backlog refinement. Nevertheless, keeping changes to a minimum is also imperative for the development team because it decreases the chance of mistakes being made [35].

4.3.2 Sprint Backlog

The Sprint Backlog is a list of items pulled from the Product Backlog which the team plans to complete during a Sprint. Usually, these tasks are presented on a physical or online board as shown in figure 9 [15].

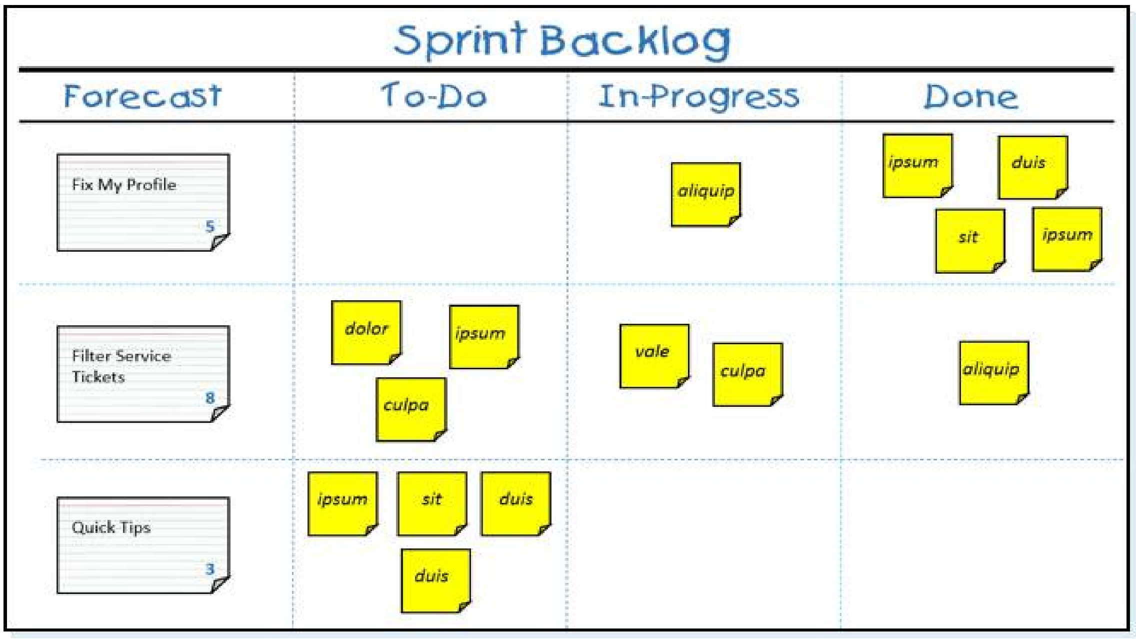


Figure 9. Sprint Backlog [35].

The Sprint Backlog is a plan that is composed of the Sprint Goal, Product Backlog items and delivering the Increment. This is created and used by the development team as well as regularly updated by them. The purpose of the Sprint Backlog is to keep all the information in a shared space for the development team and ensure that they can focus on a set of tasks avoiding scope creep [35].

4.3.3 Increments

Increments are a compilation of all previous increments and current increment where they have all been integrated, verified and are ready to be delivered. Therefore, the increment must be usable and provide value to stakeholders and customers. Developers use these increments to showcase working product and gather feedback from key stakeholders to guarantee customer satisfaction, continuously improve the product, swiftly make changes, and be more innovative [35].

5 Conclusion

Agile challenges software development teams to be self-sufficient, organized, and continuously improving in order to embrace agility for higher performance and quality. Research indicates that the traditional methods of waterfall and prototype models were not bringing the required results with the current ever-changing market. Adapting Agile has allowed companies to increase sales, product quality, business value, customer satisfaction, and adaptability to change. Regardless of company size, structure, ideals, or state, all types of organizations can adapt to the Agile framework.

Choosing the correct Agile methodology depends on business goals, type of project and customer needs. Kanban is for teams who appreciate visuals and can make use of the Kanban board. This method promotes collaboration, accountability and transparency but the team must be aware that the board must always be up to date. Extreme Programming allows teams to primarily focus on the software or coding part of the project. Through pair programming, there are less errors, continuous testing and improvement. Consequently, this means higher costs and more time investment. Eliminating waste and only using resources when there is demand is the key principle of Lean (LSD). This saves time and money by not instilling unnecessary code and requirements which bring no value to the customer. However, Lean requires more documentation and is less scalable than other methodologies. Crystal heavily focuses on individuals and communication, meaning that Crystal values the expertise of people rather than powerful tools and software. Nevertheless, the Crystal family tree can be difficult to understand for some organizations.

Scrum is the most popular of all Agile methodologies even though it follows strict guidelines. Scrum is well-documented and easy to implement, as long as teams follow Scrum procedures. Continuous feedback from the customer during each short Sprint allows for faster changes and product satisfaction. Feedback also contributes to faster changes for the product and saves the Scrum Team a lot of time and effort. Scrum Events enable Scrum Teams to evolve as a whole and endlessly improve on structural issues.

References

1 Qualitylogic 2023. 10 Reasons to Use Agile Software Development <https://www.qualitylogic.com/knowledge-center/10-reasons-to-use-agile-software-development/#:~:text=With%20Agile%20software%20development%2C%20teams,deliver%20a%20higher%20quality%20product.> Accessed 16.2.2023

2 Laneways 2022. History of Software Development: Brief Guide for Starting Developers. <https://www.laneways.agency/history-of-software-development/> Accessed 17.2.2023

3 Hughley, Douglas. 2009. Comparing Traditional Systems Analysis and Design with Agile Methodologies. <https://www.umsl.edu/~hugheyd/is6840/waterfall.html> Accessed 17.2.2023.

4 ProjectManager 2023. The Ultimate Guide Waterfall Model. <https://www.projectmanager.com/guides/waterfall-methodology> Accessed 17.2.2023.

5 Waseem, Ahad 2022. Waterfall Methodology: History, Principles, Stages & more. <https://management.org/waterfall-methodology> Accessed 17.2.2023.

6 TutorialandExample. 2019. Prototype Model in Software Engineering. <https://www.tutorialandexample.com/prototype-model-in-software-engineering> Accessed 17.2.2023.

7 Lewis, Sarah. 2023. Prototype Model. <https://www.techtarget.com/searchcio/definition/Prototyping-Model> Accessed 17.2.2023.

8 Highsmith, Jim. 2001. Manifesto for Agile Software Development. <http://agilemanifesto.org/> Accessed 17.2.2023

9 Wrike. 2023. What is the Agile Methodology in Project Management. <https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/> Accessed 17.2.2023

10 Product Board. 2023. Agile Values. <https://www.productboard.com/glossary/agile-values/> Accessed 17.2.2023

11 Eby, Kate. 2023. Comprehensive Guide to the Agile Manifesto. <https://www.smartsheet.com/comprehensive-guide-values-principles-agile-manifesto> Accessed 17.2.2023

12 Flynn, Jack. 2022. 16 AMAZING AGILE STATISTICS [2023]. WHAT COMPANIES USE AGILE METHODOLOGY. <https://www.zippia.com/advice/agile-statistics/ - :~:text=At%20least%2071%25%20of%20U.S.,more%20successful%20than%20waterfall%20projects.> Accessed 17.2.2023

13 Kiguolis, Linas. 2023. Software Development Frameworks in 2023. <https://codeornocode.com/software-development/best-agile-frameworks/> Accessed 17.2.2023

14 Knowledgehut. 2023. Scrum History. <https://www.knowledgehut.com/tutorials/scrum-tutorial/scrum-history> Accessed 17.2.2023

15 Digite. 2023. What is Scrum? < https://www.digite.com/agile/scrum-methodology/> Accessed 17.2.2023

16 Takeuchi, Hirotaka & Nonaka, Ikujiro. 1986. The New New Product Development Game. <https://hbr.org/1986/01/the-new-new-product-development-game> Accessed 20.3.2023

17 Agilest. 2023. Why Does Scrum Work? <https://www.agilest.org/scrum/why-does-scrum-work/> Accessed 20.2.2023

18 Indeed Editorial Team. 2022. List of Scrum Advantages and Disadvantages. <https://www.indeed.com/career-advice/career-development/disadvantages-of-scrum> Accessed 20.2.2023

19 Kabanize. 2023. What is Kanban? Explained for Beginners. <https://kanbanize.com/kanban-resources/getting-started/what-is-kanban> Accessed 20.2.2023

20 Javed, Rashid. 2022. Kanban. <https://www.accountingformanagement.org/kanban/> Accessed 20.2.2023

21 ProductPlan. 2023. What is eXtreme Programming? < https://www.productplan.com/glossary/extreme-programming/> Accessed 20.2.2023

22 Agilealliance. 2023.Extreme Programming (XP). <https://www.agilealliance.org/glossary/xp/ - q=~(infinite~false~filters~(postType~(~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1)> Accessed 20.2.2023

23 Teamhood. 2023. What is Scrumban? <https://teamhood.com/agile-resources/what-is-scrumban/> Accessed 20.2.2023

24 Digite. 2023. What Is Extreme Programming (XP)? & It's Values, Principles, And Practices. <https://www.digite.com/agile/extreme-programming-xp/> Accessed 20.2.2023

25 ProductPlan. 2023. The 80/20 Rule for Agile Product Managers. <https://www.productplan.com/learn/80-20-rule-agile/> Accessed 21.2.2023

26 Rice, David. 2015. Why 45% of all software features in production are NEVER Used. < https://www.linkedin.com/pulse/why-45-all-software-features-production-never-used-david-rice/> Accessed 21.2.2023

27 Lean Enterprise Institute. 2023. Explore Lean. <https://www.lean.org/explore-lean/> Accessed 21.2.2023

28 Kiguolis, Linas. 2023. Discover The Best Agile Software Development Frameworks in 2023. <https://codeornocode.com/software-development/best-agile-frameworks/> Accessed 21.2.2023

29 Simpilearn. 2021. Lean Software Development: Definition, Principles, and Benefits. <https://www.simplilearn.com/what-is-lean-software-development-article> Accessed 21.2.2023

30 Lutkevich, Ben. 2021. Lean software development. <https://www.techtarget.com/searchsoftwarequality/definition/lean-programming> Accessed 21.2.2023

31 Jena, Satyabrata. 2022. Crystal methods in Agile Development/Framework. <https://www.geeksforgeeks.org/crystal-methods-in-agile-development-framework/> Accessed 21.2.2023

32 Airfocus. 2023. What is the crystal agile framework. < https://airfocus.com/glossary/what-is-the-crystal-agile-framework/> Accessed 21.2.2023

33 Singh, Virender. 2021. Crystal Method in Agile. <https://www.toolsqa.com/agile/crystal-method/> Accessed 21.2.2023

34 Christie, Jim. 2020. Scrum Graphic. < https://jimchristie.me/blog/scrum-graphic/> Accessed 22.2.2023

35 Scrum. 2023. The home of Scrum. < https://www.scrum.org/> Accessed 22.2.2023

36 Teamretro. 2023. What is a Sprint Retrospective? < https://www.teamretro.com/retrospectives/sprint-retrospective> Accessed 22.2.2023