# Chapter 0: Introduction to Agile Development

What is Agile development?

Agile software development is made up of various approaches to software development under requirements and solutions grow through the combined effort of cross functional and self-organized and multi-functional teams working with their customers or the end user of the software. The Agile process that was used for this project is called SCRUM. This process allows us to focus on delivering the highest business value in the shortest time. SCRUM allows us to quickly and constantly inspect our actual working software. This is done in rapid periods of time known as “sprints”. Sprints usually last anywhere from one or two weeks up till one or two months. The time for the sprint would depend on the size and difficulty of the task that the team has been assigned. In SCRUM, the team does not set the priorities, the business sets the priorities for the team. Then the teams will self-organize to determine the best way to deliver the highest priority features. As a result of short sprints anyone can see real working software and decide to release it as the software is or continue to work on the and improve upon the software for another sprint.

Why did Agile come about?

Agile development came about as a result of the need for a lightweight development method for developing software. Agile development allows the development team to change and develop software easier than any other methods. This is due to the sprints referred to in the previous paragraph. The sprints of an agile development method, such as SCRUM, allow the user to constantly adjust, test, improve and implement new and existing features in the code, resulting in a faster and much more streamlined development approach. Let’s say for instance if I’ve been working on let’s say a simple CRUD app which is part of the larger project as a whole, because of these short sprints I may be given only a few weeks to implement this element However, unlike other software processes I can see straight away if my code works with the project as a whole as with testing is done as part of the sprints unlike other methods which will only allow testing to occur at the end of the development process. Therefore, this makes implementing the code and testing the code becomes easier and quicker. If bugs are discovered in the software, they can be found out quickly and then can be fixed or re implemented before we move on to new code or other parts of the project. That means that when the project is finally done, it is done, and the need to go back and fix code that was developed five or six months ago, which would be harder to develop as it will no longer be fresh in your mind, it may even take longer to fix than it would have five or six months ago.

Principles of Agile software development

There are 12 principles of Agile software development. They are:

1) Customer satisfaction by early and continuous delivery of valuable software.

2) Welcoming changing requirements, even in late development.

3) Deliver working software frequently.

4) Close, daily cooperation between businesses and developers.

5) Projects are built around motivated individuals, who should be trusted.

6) Face to Face conversation is the best practice of communication.

7) Working software is the primary measure of progress.

8) Sustainable development, able to maintain a consistent pace.

9) Continuous attention to technical brilliance and good design.

10) Simplicity the maximizing the amount of work not done is necessary.

11) Best architectures, requirements, and designs emerge from self-organization.

12) Frequently the team looks at ways to become more effective and will adjust where necessary.

The focus of Agile development

The main focus of Agile development is on quality or quantity. The use of specific tools and methods are, for example, pair programming, test-driven development, design patterns, code refactoring, unit testing. These methods are used to improve quality and enhance the product development agility. Compared to traditional software engineering, Agile software development mainly targets complex systems and product development with dynamic, non-deterministic and non-linear characteristics. Accurate estimates, stable plans, and predictions are often hard to get in early stages, and confidence in them is likely to be low.

The characteristics of Agile development

Self-organizing teams are able to manage and look after themselves which leaves very little managing. In a quick progress, product progresses in a series of month-long sprints which allows the project to move on at a rapid pace. Requirements are captured as items in a list of product backlog. The use of generative rules to create an agile environment for delivering projects. The main characteristic of Agile development is the sprint, your typical sprint will last 2-4 weeks. The constant duration helps the team to find a better rhythm. Before the next sprint begins the product is designed, coded, and tested during the current sprint which means this sprints problem does not become the next sprints problem. The sprints should be planned around how long you can commit to keeping change out of the sprint.

The SCRUM framework

There are three parts to the SCRUM framework, they are Roles, Ceremonies and Artifacts:

* The Role consist of the product owner this is the person who is essentially in charge of the product, will lay out features, decide release dates, prioritize features, and accept and reject the result of the work. Next there is the SCRUM master they are the head of each SCRUM meeting and are the link between management and the team, they shield the team from external interferences and ensure the team is fully functional and productive. Then there is the team, which is typically made up of 5 to 9 people, and they range from software developers to UI designers to testers, etc. Most of the team will be full time with a few exceptions.
* The Ceremonies are the sprint planning where the sprint is prioritized. The daily SCRUM takes typically 15 minutes and anyone who is involved is invited. However, only team members, SCRUM master and product owner are allowed to talk. The purpose of the scrum meeting is to avoid other unnecessary meetings during the day. At the meeting everyone will answer three questions, “What they did yesterday”, “What they will do today” and “Is anything in their way”. In the sprint review the team presents what it has been accomplished during the sprint. This will typically be a demo of new features. In the sprint retrospective you periodically take a look at what is and is not working.
* The Artifact is the product backlog, which is a list of all desired work on the project. It should ideally be shown such that each item in the backlog has values to the users or customers of the project. In the sprint backlog, individuals on the team sign up for the work that that want to do work is never assigned in this method. The sprint goal is a short statement of what work will be focused on during this particular sprint. Any team member can add, delete or change the sprint backlog. A sprint burndown chart is created to make the work of the team visible as a graphical display.

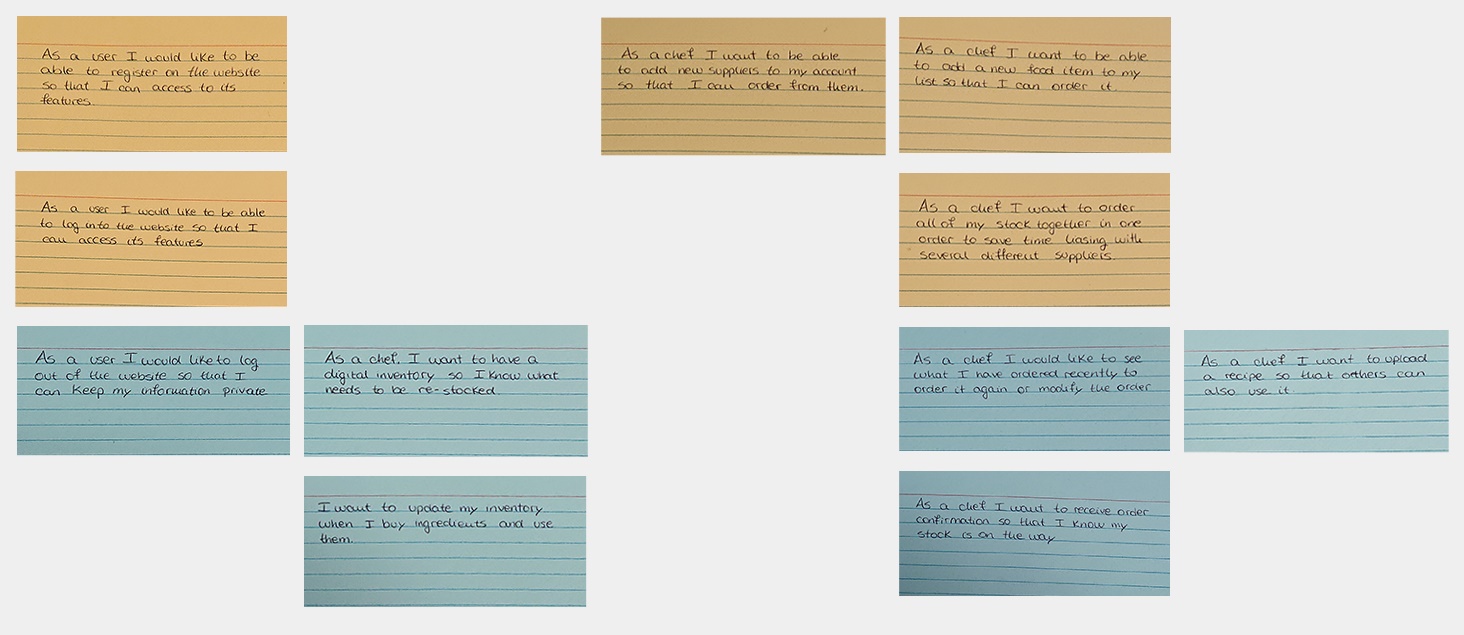
The last feature of Agile development would be planning poker. Planning poker is a method of time estimation. What happens is the team will assemble with a deck of cards each with values that represent the number of hours. The moderator will read out a user story and each member will place a card face down in front of them, and when all cards are down, they are flipped over. If the numbers, which are estimates for how long the user story would take to develop, vary widely it is discussed and the members repeat this process until they can get their time estimates closer together.

With all the benefits that Agile software development has, there are a few downsides: Agile practices can be inefficient in large organisations and certain types of developments. Lots of organisations have the believes that agile software development is too extreme and therefore they tend to adopt some hybrid approaches that is a mix of elements of Agile software development. In relation to time testers, customers and developers must constantly interact with each other, this involves numerous “face to face” conversations as they are the best form of communication. All involved in the product must have close cooperation. People who are daily users need to be available for quick testing and sign off on each phase so developers can check it off as complete before moving on to the next part of the project. This ensures the product meets user expectations but is very cumbersome and time consuming. Therefore, it demands more time and energy from all those involved in the project. In Agile development projects can easily fall off track and go by the wayside. As Agile development requires very little planning to start and takes the assumption that the consumers’ needs are constantly changing. With very little to go off of this can limit the Agile development model. If a consumers’ feedback or communication are not clear a developer may go ahead and focus on the wrong areas of development. A very simple project can become a never-ending project very quickly.

# Chapter 3: The SCRUM process. (4,500 words)

First, every member of the group thought of an idea about what to do with the project. Then all the members got together and presented our ideas. We decided to mix two of the ideas and, after deciding what our web application was about, we started the Agile development process by holding the initial SCRUM meeting. In this meeting it was agreed to create a story map, where the features of the application would be captured, to set releases of each sprint and to decide the duration of them.

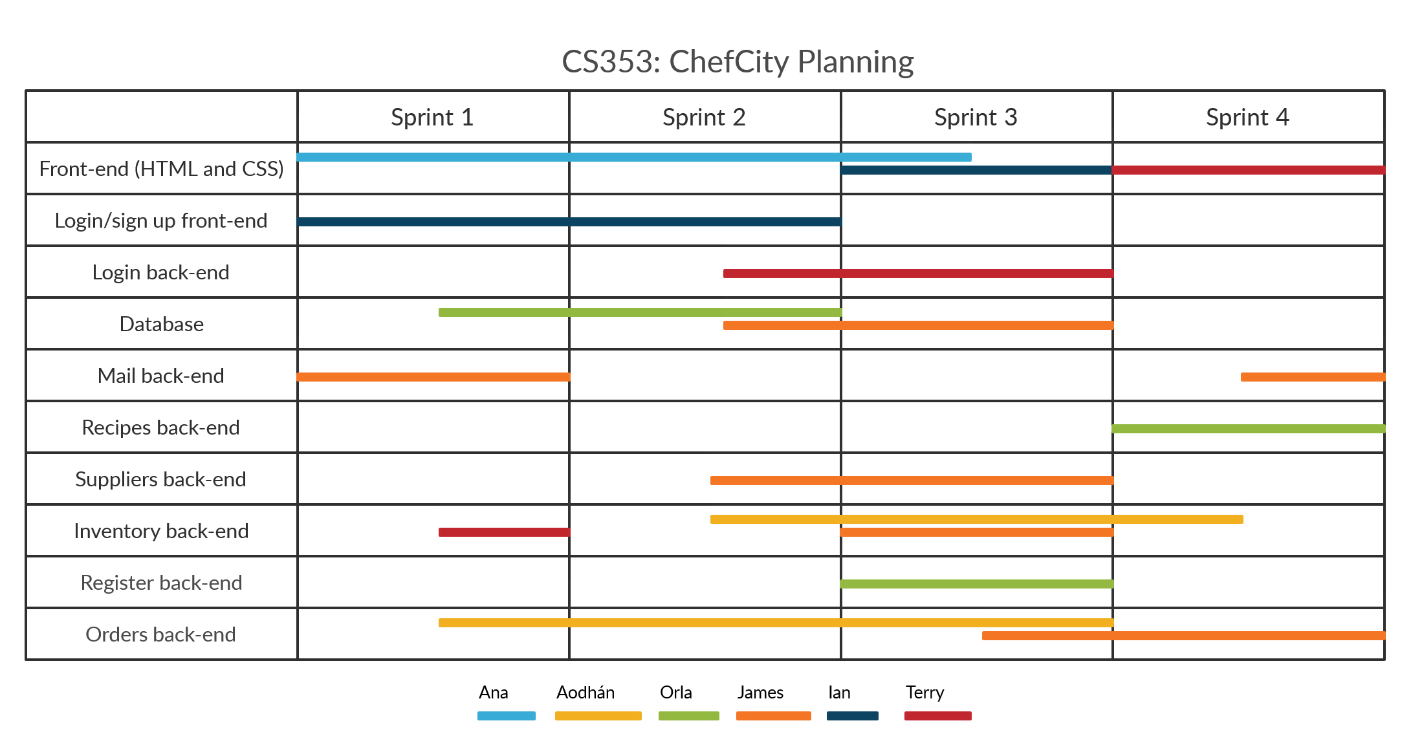
A user story map shows all the end-user requirements for the application and helps people to organize user stories into a useful model for understanding the functionality of the system. Some of the benefits of using a story map are, for example, manage backlog with an overview structure and prioritise user needs. The goals have to be determined first, then the tasks are set and finally these are implemented using the tools.

 We decided that our application would only have one user, a chef, who will be able to access all the functionalities of the system. In the beginning our story map had more functionalities, but as time went by, we realized that there were some things that were too complicated and beat about the bush, so we decided to eliminate them. The final story map is the following one:

We also determined the duration of each sprint, which in our case will be of two weeks, so there were four sprints in total. At the end of this meeting we decided which were going to be the objectives of each sprint. For example, we decided to finish first a kind of template of the front-end which everyone could work with. Then, each member would develop the back-end of every functionality so that they could do some demos to check whether or not it was working. At the same time, other members would work on the front-end and back-end of sign up and sign in pages.

The next thing we had to do was the planning poker. As explained in chapter 0, planning poker is a method of estimating the time it will take to complete a task. In our application, we estimated the time it would take to develop each task. In the following table you could see the planning of the development.

|  |  |
| --- | --- |
| Tasks | Estimated Time |
| Database | Two weeks |
| Website Front-end | Five weeks |
| Sign in/sign up back-end | Three weeks |
| Mail back-end | Two weeks |
| Recipes back-end | Two weeks |
| Inventory back-end | Three weeks |
| Supplier back-end | Four weeks |
| Inventory back-end | Three weeks |
| Orders back-end | Four weeks |

However, as with any estimation, it does not agree with reality as a whole. Therefore, now that the app is almost ready, the time it took us to do each task is indicated in the following Gantt chart:

In each beginning of sprint, we had a meeting that lasted around 15 minutes, where the SCRUM master asked each member of the group the three questions of the daily SCRUM, described above. Plus, at every meeting, we also did a demo of what each had implemented up to that point, if possible. That way, we knew what the others were doing and whether or not they were successful. During the first sprints, we hadn’t got too much to share with each other, but as the time went by, we were able to merge the parts that each of us developed.

The SCRUM Institute gives a thorough definition of a SCRUM Burndown chart as “a visual measurement tool that shows the completed work per day against the projected rate of completion for the current project release”. Its goal is to allow the project to be in the process of delivering the expected solution within the desired timeframe. The burndown chart related to this project is the following one:

In our case, this graph shows that we had to make more effort in developing the project in the weeks of the beginning (sprints 1 and 2) and as we were finishing the project, the effort has decreased.

As explained before, during all the sprint every member of the group worked in his own part and at the end of each sprints, we had a meeting where we explained and showed what we had done during those weeks. As each member developed a different task, each one had to do different tests of it to know when the result obtained was the expected one. Besides, in those meetings everything was pooled and all the code was tested together, seeing if everything was going as expected or there were some things to fix. After each sprint, the tasks were reorganized and the sprint process started again.

If we had an extension of time to keep working on this project, we could try to improve all the functionalities to the maximum and maybe we could add other features. We also could improve the front-end design to make it cleaner and visually appealling.

When you work on a project together with other people, the best way for everyone to share their code with each other and see what they are doing together is to upload the project to an online repository. We decided to use GitHub because it provides hosting for software development and it offers version control and source management, what is what we needed. At the beginning it was a little chaos because we weren’t familiar with how to use GitHub, but as everyone knows, with practice you get control.

We had the place to share our software but we needed a way to communicate each other, so we agreed to use a professional platform to communicate, such as Telegram. This is a cloud-based instant messaging and voice over IP service, and it is available for Android, iOs, Windows Phone…, so there wouldn’t be any problem with any mobile phone. This platform is also used to send messages and exchange photos, videos and files of any type.

# References

GitHub. (2019). Build software better, together. [online] Available at: https://github.com/ [Accessed 1 Dec. 2019].

Productplan.com. (2019). Story Mapping | Definition and Overview of Story Mapping Prioritization. [online] Available at: https://www.productplan.com/glossary/story-mapping/ [Accessed 1 Dec. 2019].

Telegram. (2019). Telegram – a new era of messaging. [online] Available at: https://telegram.org/ [Accessed 1 Dec. 2019].

Visual-paradigm.com. (2019). What is User Story Mapping?. [online] Available at: https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story-mapping/ [Accessed 1 Dec. 2019].