Table of Contents

How to apply Agile development to school management software

By: Wissam Salhab, Nathan Miller, Moijul Islam

Description of Agile

Agile is a development ideology which, at its core, is based around working closer with clients to create a project closer to their desires. It was designed by a group of 17 individuals who met in Snowbird, Utah and put out a manifesto outlining their findings on how to make software development better[1]. They said they valued: Individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan[2]. The hope is that you can build a framework which is nimble enough to adjust to changing demands, while having a deliberate tolerance to learn from mistakes as they are being made, ensuring that there is a tighter feedback loop from clients to developers, and ultimately resulting in a faster product to market that's more durable and flexible.

Agile consists of 12 principles, which are often applied in similar ways. These twelve principles are:

- The development team's highest priority is to satisfy through early and continuous delivery of valuable software.

- the team should welcome changing requirements, even late into development, because that way it caters more to the clients and their satisfaction.

- the aim of development is to deliver working software frequently, with the shorter the development cycle the better, that way the clients can ask for modifications if they spot something that would be better if changed.

- developers must work daily with businesspeople, because that allows you to create a sort of synchronization between the people who are responsible for developing the software and those who plan on selling it.

- build projects around motivated people and give them the resources and trust for them to get the job done. By motivating team members, the teams efficiency increases and the product of that is faster development of better quality software.

- the most efficient way to convey information is through face-to-face conversations. Considering that the principles of agile development were made in 2001, this method is outdated now since in the modern work environment contacting someone across the globe is severely less limited that face to face. That being said, face to face conversations can show human reactions to certain things which if interpreted correctly can show you a lot more.

- working software is best way to measure progress, this means that regardless of how many hours was put into the project the result at certain milestones must be reached. The customer expects working software and that is what should be delivered.

- development should sustainable, and thus the pace should be able to be maintained indefinitely so you can frequently deliver to the market and respond to any modifications required.

- continuous attention to good design and technical skills enhances agility. Continuous excellency ensures that your product is of high quality at every stage which means responding to changes made would be easier since the program is less likely to have bugs when modified.

- simplicity, that is to say the minimisation of work down, is essential because the client wants a simple solution to their issue, making a complex solution just makes things harder.

- the best work comes from teams that organize themselves. When provided with freedom, these teams would be able to work more motivated and thus providing a better product.

- you must regularly reflect on how to be more effective and adjust accordingly [3]. While these can be applied in many ways, the most common way is the Scrum framework.

To achieve their goal of creating a more efficient development method, Scrum works in a fundamentally different way to more traditional (waterfall) development- instead of planning out the entire system beforehand, and then coding the entire system at once, Scrum works by planning and developing each part of the system individually. Each of these is called a sprint- a chunk of time, usually 4-6 weeks but in some places as low as 1-2 weeks, where a specific ticket is worked on. A ticket is derived from a specific requirement from the shareholders. At the end of each ticket, a retrospective is held, allowing each member of the team to talk about how they felt the sprint went and give feedback to the scrum manager. The scrum manager is in between a project manager and a stakeholder ambassador; they help keep the development team running at capacity by caring for their various needs, but also help ensure the requirements are being stuck to so that the client gets what they want. Unlike typical project managers, scrum managers often have no background in technology and more of a background in management. Scrum managers also run stand ups, which are meetings that are held, in most cases, daily. They get their name from the fact that each member of the team is encouraged to stand rather than sit, as this makes it harder for them to be distracted. At a stand up, each member of the development team says what they did yesterday, what they plan to do today, and any problems they are currently facing. The scrum manager will then solve the problems being faced by the team, and relay progress to other teams or to the client[4]. Having these meetings regularly means that the team always knows how much progress has been made and what needs to be focused on. The efficiency of a sprint, and the productiveness of a standup is highly reliant on a scrum manager; it is of the utmost importance that a scrum manager keeps on top of the problems facing the team.

How to apply Agile

Were we to follow the Agile principles, through Scrum or through another method, we would want to tackle each part of the brief separately, writing a ticket for our current project. We could elect a Scrum manager, who could rotate for each sprint, and then work together on the current ticket. There are other methods than Scrum however; we are unlikely follow Scrum due to its leadership structure. As long we split the project into small tickets and complete them one at a time, producing useable code, we will be following the Agile principles.

A major part of this process would be far more flexible than a waterfall development cycle, due to continuously iterating between planning and implementation. This is good as conventional waterfall development leads to a far too rigid and strict process, decreasing the quality of the final product. By

contrast, we would be working to complete the project by setting reasonable goals regularly and allowing ourselves time to fail on each step of the implementation multiple times. We would then meet up and tell each other what was holding us back.

The essence of our project is effective communication among members either through face-to-face or online conversation. This would be reflected in our regular meetings wherein we establish the work that needs to be completed and the work that has already been completed. By knowing what other people have done and are doing, we would increase our sense of teamwork and thus our motivation, making the production faster and simpler.

Our main objective under Agile would be to practice simplicity which is important for the system to avoid the structure being too complicated. By this we mean maximizing the amount of work that isn't done, such as superfluous features and redundancy in code, specifically through classes. This does many things; it reduces the overall file size of the system, making it faster to load and easier to download to new computers; it reduces the amount of work that needs to be done by the team, making the project faster to complete; it makes the code more readable for other developers in the future.

Conclusion

While not better than Waterfall development in all cases, Agile development is far more useful for our purposes. Following the 12 principles will allow us to create useful code quickly and efficiently, without overloading ourselves with planning before we've even started.

References

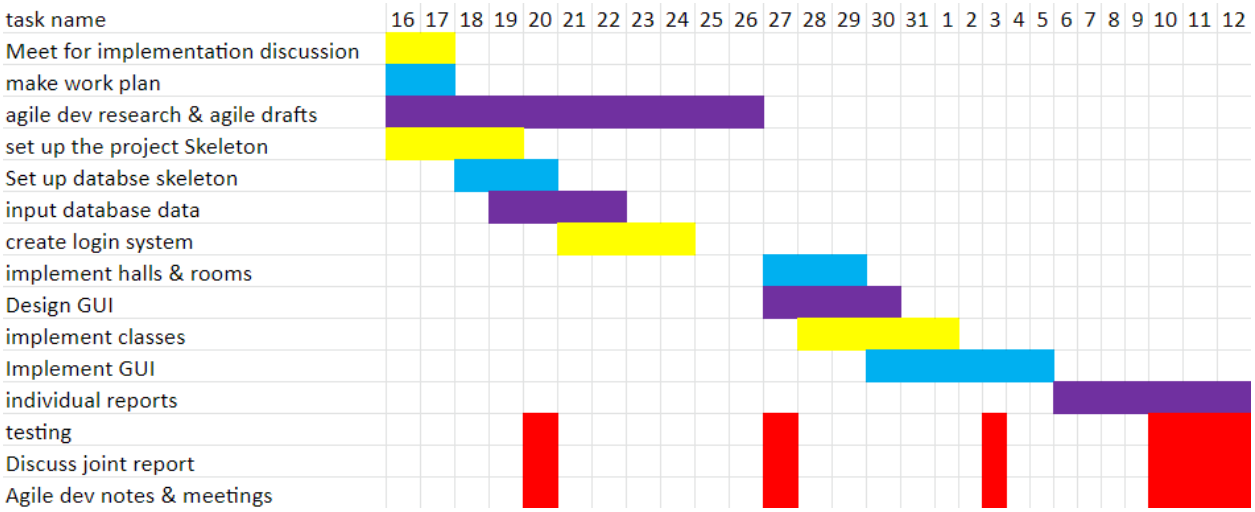
1. Beedle, B. (2001) *History: The Agile Manifesto*. Available from: https://agilemanifesto.org/history [Accessed 13 December 2021].

2. Beedle, B. (2001) *Manifesto for Agile Software Development*. Available from: https://agilemanifesto.org/ [Accessed 13 December 2021].

3. Beedle, B. (2001) *Principles behind the Agile Manifesto*. Available from: https://agilemanifesto.org/principles [Accessed 13 December 2021].

4. Schwaber K., Sutherland J., Casanave C., Miller J., Patel P., Hollowell G. (1997) *SCRUM Development* Process [online]. (eds) Business Object Design and Implementation. Springer, London. [Accessed 16 December 2021]

Appendices


task name
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12
Meet for implementation discussion
make work plan
agile dev research & agile drafts
set up the project Skeleton
Set up databse skeleton
input database data
create login system
implement halls & rooms
Design GUI
implement classes
Implement GUI
individual reports
testing
Discuss joint report
Agile dev notes & meetings