Title of the article
Scrum is better than Waterfall – a case study of software project GotoGro-MRM

Dylan Jarvis
Student id 102093138
102093138@studnet.swin.edu.au

Abstract
The scrum development method is better than the waterfall method.

1. Introduction
This article explores the similarities and differences between the Scrum and Waterfall models of software development, using the Goto-Gro project as a case study. The Goto-Gro project was undertaken using Scrum, however the waterfall method is a mainstay of industry development and comes with its own set of advantages. This report outlines the case requirements and a brief background on the two models before comparing the pros and cons. A conclusory statement will be made on which model is most appropriate given the information analysed.

2. Background
Goto-Gro is a membership-based grocery store which sells a large range of products. With recent problems in inventory management, they commissioned a development team to implement a digital solution which would track their members, products, and sales data to the ends of providing a weekly stock report and improve their inventory management. The reports were to be delivered in csv format, with a UI system for accessing/changing all records. These requirements were straight forward and could have been handled in several ways. Two of the most ubiquitous models are presented here:

Introduced in 1970 by Winston Royce, the waterfall model would structure the task linearly, with each phase of development only starting after the preceding task much like the flow of water down a fall as shown in **Figure 1** [1].

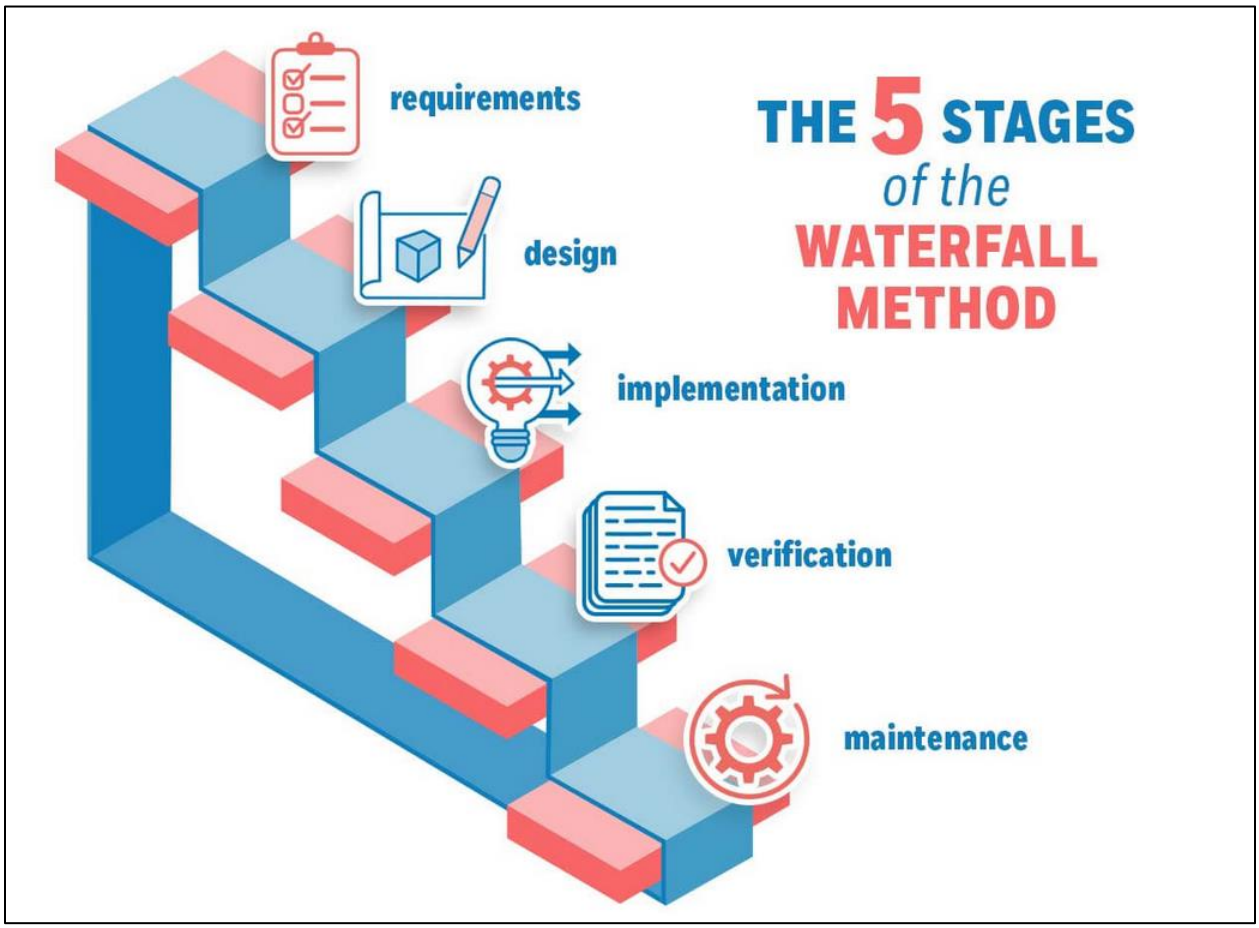

Figure 1: *Waterfall model.*

The success of the Waterfall method hinges on the quality of work done in the initial stages, that being the planning and documentation phase. Following the adage of “measure twice, cut once”, the predictions made at the start are hypothetically highly accurate [2]. A large focus on planning allows for many problems to be identified and addressed before development starts.

Contrastingly, the Scrum model is much newer, proposed in 1995 almost specifically to address limitations in the Waterfall model. It is considered more adaptive and cooperative, with many components able to be run in parallel using the strengths of different team members to ensure constant progress [3]. The Scrum method is considered more iterative, with a looping structure shown in **Figure 2**. Check-ins are taken every day to update on progress and redistribute development resources as necessary, though at its most effective, the goal of the Scrum is also clearly defined like the Waterfall.

Figure 2: *Scrum model.*

3. Waterfall vs Scrum for Addressing Common Project Challenges

This section compares the scrum and waterfall models using the project challenges as case points. The major challenges facing this project are largely the same as the challenges facing any software project, but a few stand out.

a. Loose Problem Definition

First and foremost, problem definition and scope were the greatest challenges. The client, though clear on some requirements, was vague on others. This becomes a problem specifically when there is a clear outcome (i.e. csv reports), but no clear content for the outcome (i.e. what should be included). This goes hand in hand with another major problem being scope creep, or the tendency for a underdefined problem to gain complexity throughout the planning phase [4].

The Waterfall model addresses this problem well. The Waterfall uses a concept called “change control”, which basically means the agreed product at the end of the client consulting phase is the final product and cannot be modified in any way. Though this directly address the problem from a developer’s perspective, this falls over when the product owners outright cannot or will not provide clarity due to changing market conditions or pending changes to other parts of the business. Further, the strictness of the Waterfall forces product owners to think in advance and basically plot out everything they need before they have seen it in action, often an unrealistic

expectation. Specifically for this project, a strict final product would be unreasonable, especially given that the software commissioned was the client's first digital solution. With nothing to draw on for comparison, the client can not be expected to have a firm grasp of what is even possible let alone what they need to inform business decisions.

The Scrum model on the other hand seems to suffer more from scope creep, however it is not a prevalent as it appears at face value. Scrum is fundamentally different to Waterfall in accounting for changes that *will need to be made* in future, which means that at the time budgeting and planning phase, cushions are put into place to account for "unknown unknowns". Should the client then ask for a change or new feature in the middle of development, it is not considered scope creep if it can be fit into the budgeted time. Despite this, it can still occur, but the Scrum way of handling it allows for the project to be expanded, *only if* the timeframe is expanded as well [5]. If this fits with the client, this can be the ideal win-win situation. For this project the Scrum models malleable nature worked well. As components of the software were developed, new features were thought of that could not have come up in a classical planning phase. Where time permitted, this allowed for the product to better suit the client's needs.

b. Difficulty Estimating Time and Resources

Another prevalent issue comes from estimation. For experienced group of developers, estimation is easier whichever model you look at, however for a new development team, or a development team facing a brand-new problem, estimation is a critical challenge that must be managed or monitored to achieve success.

The Waterfall model addresses estimation by drawing from previous cases. For the same number of developers and the same set of tasks on the same software, the Waterfall model can be considered highly reliable in determining a timeframe. In many ways the Waterfall model *needs* to be highly accurate, as a project running under this model *must* be delivered on time or risk conflicting with other projects or worse: Breaking the terms of the contract. With this in mind, it can be surmised that the Waterfall model does not handle discrepancies in time and resources well, if at all. For this project, noting the team composition, the Waterfall model would have put unnecessary pressure on the planning stage, and if the task list started to expand beyond control, would put extreme stress on the team.

The Scrum model again fills in the shortfall in the Waterfall model. It does not call for stringent estimations of time and resources, as there are inbuilt redundancies. For this project, though the deadline was defined, the estimation effort of various components needed to be more flexible. This flexibility also goes hand in hand with another distinctly Scrum-based advantage, which is the opportunity to leverage any extra team member capacity and easily redistribute them to another task, something that is more explicitly explored in the next section.

c. Completing on Time

Obviously, it goes without saying that completing the project on time is the most inherent challenge to a software project. While previous sections have mentioned that the Scrum model

can overcome scope creep by responsibly extending the development timeframe, more often than not, this is not possible in an industrial space. Even for this project, the deadline was fixed with no option to push it back even if constant work was delivered throughout the sprint. How then, do the models handle this most fundamental of issues?

The Waterfall model handles it by planning, planning and more planning. With expertise and research, this model seeks to make the problem a statistical minority by clarifying every step of the process as tightly as possible. By doing this, the team should be able to say with confidence exactly when they will be finished. To this effect, each team member will also be aware of exactly what they are doing at each stage of the development. The catch is that some team members may be on hold waiting for a prior step to be completed. For this project, this kind of linear progress would not have been ideal as the idea was to engage as many team members as possible. In tandem with the loose scope, the team could not afford to spend time waiting for other components to be complete before starting theirs as the delays could compound and blow out of proportion.

Here, the Scrum model presents possibly the greatest advantage over the Waterfall model. Where the Waterfall would call for a UI developer to wait for the back-end developers to finish, the Scrum instead pushes the UI developers to start on their own components immediately. While this may lead to a bunch of components complete in their own right but not linked together, this is fine as the time saved in building the components in parallel can now be used gluing them together, often for a net gain. This is not exclusive to the development phase either. The benefit of Scrum is that team members can start working even at the planning phase and in turn inform the planning in itself. It does not and should not take the whole team to liaise with the client, so instead of waiting for instruction, the agile approach of Scrum allows for better time management over the Waterfall. This is ideal for this project as there were multiple definable components which could be built in parallel. Loosely speaking there were SQL statements, triggering code and UI components, all able to be worked on independently, and the Scrum model allowed the team to harness parallel development to its fullest.

4. Conclusion

To the ends of completing a project on time in the face of a loose scope with difficulties in estimating time and resources, the Scrum model prevails as being the most suited to the job. Before concluding outright that it is superior however, it must be acknowledged that the Scrum model itself was only made possible by a changing landscape of technology. There is a reason that it is the younger management model, and that reason is because is has only appeared following the advent of visual website builders and top-level UI building software. Before that, high level components literally *could not* be built before the underlying code was made. In this sense, it is logical to structure the Waterfall model linearly.

Today, the Waterfall model still has a place, but only in development where the end goal is either strictly defined, or the client has a lot of experience with digital technologies and is able to articulate exactly what they want at an early phase of the project.

With the ease of access to technologies allowing for parallel development, and when addressing a small company's first digital venture, as it was for the Goto-Gro project, the Scrum model is the ideal management model.

References

[1] C. Kaur and V. Kumar, "Comparative Analysis of Iterative Waterfall Model and Scrum," *IJCSR*, vol. 3, no. 1, pp. 11-14, Jan. 2012. [Online]. Available: https://ijcsr.forexjournal.co.in/paperspdf/9.pdf. [Accessed: 23-Oct-2022].

[2] Adobe, "Waterfall methodology - a complete guide | Adobe Workfront," Adobe Experience Cloud Blog. [Online]. Available: https://business.adobe.com/blog/basics/waterfall. [Accessed: 23-Oct-2022].

[3] "What is scrum methodology? & scrum project management," Digite, 30-Sep-2019. [Online]. Available: https://www.digite.com/agile/scrum-methodology/. [Accessed: 23-Oct-2022].

[4] Orient, "10 most common software development challenges," 10 Most Common Software Development Challenges, 30-Dec-2021. [Online]. Available: https://www.orientsoftware.com/blog/software-development-challenges/. [Accessed: 23-Oct-2022].

[5] A. Smith, "4 ways to deal with Scope Creep," Work Life by Atlassian, 07-May-2020. [Online]. Available: https://www.atlassian.com/blog/inside-atlassian/4-how-tos-dealing-with-scope-creep. [Accessed: 23-Oct-2022].