Is Scrum is better than Waterfall in the context of Sales Management System (SaleManSys)? What about in context of all software projects?

Student Details:

Name: S M Ragib Rezwan
Student id: 103172423
Email: 103172423@student.swin.edu.au

Content:

Abstract

This research report will be focused on comparison between Scrum and Waterfall SDLC, in terms of the project "Sales Management System" (SaleManSys), in order to show why scrum is better suited for it and try to speculate whether it is superior to waterfall for other software projects too.

The keys findings in this examination have been the following facts:
For the SaleManSys, Scrum SDLC had been better suited than waterfall SDLC in terms of dealing with Scope change (either due to initial project scope inaccuracy or due to change in customer demand), in resolving issues internal team conflicts more smoothly (due to higher member interaction over time), in resolving team and client conflict (due to higher member to client interaction and by considering client as also a team member) and in completing tasks of similar quality in a shorter deployment time.

Introduction

Currently in modern times, we can hardly imagine life without software. It can be in the form of navigating our way around via GPS, chatting with our peers on social media like "Facebook", keeping track of transactions via "Sales Management Systems" or even ordering food via "Menulog". No matter where we look, we see different types of software, both specialized and general purpose, assisting us in completing our tasks and making our lives more far more enjoyable.

But how are all of these wonderful softwares being developed? This answer varies depending on the type of software being developed and the preferences of the team developing it. For some, it is the traditional software development life cycle like Waterfall, while for others it is the Agile software development method like Scrum.

But between those two very different Software development life cycles (or SDLC), which is better? Waterfall or scrum?

In order to answer this question, the had developed the project "SaleManSys" using both Waterfall SDLC and Scrum SDLC and then compared on how the major challenges faced in the project had been approached (and wherever possible, resolved) by the two development processes, in order to come to a conclusion on which is better.

So, in the following the paragraphs in this report, background of the Project, Waterfall process and Scrum process has been noted in brief, before moving into details about the challenges faced, how waterfall and scrum had approached and (wherever possible) resolved these challenges, and an overall comparison among them (for each challenges) in the comparison outcome, before deciding on which process is better in the conclusion.

Background:

SaleManSys

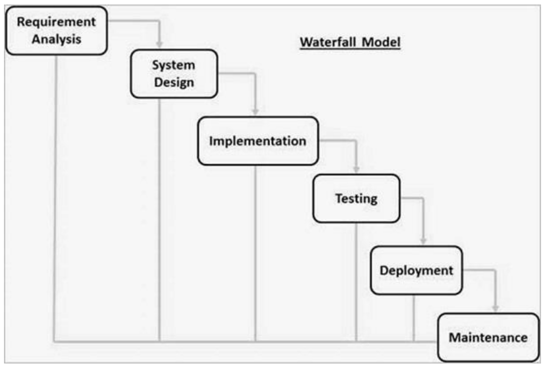
"Sales Management System" (or SaleManSys for short) is just one of the regular type of management software that are in quite high demand in today's market, especially for small business (like Ragib Televisions) who are trying to "go digital" **[1]** (ie move from paper based data storage and processing to one done electronically/ virtually on computers)**.** Although it's main purpose is to just "store, manage and analyse sales records of products" (ie. The televisions sold by the company), both with high accuracy and in short time, its data can also be used by the company to help their business in various other ways too. This includes, finding out about current customer demands and trends, keeping accurate and timely track of product stocks, etc. Thus, although the company is only seeking the software to help streamline a single aspect of its business (ie in management of its "sales records"), its reach and importance is far greater in the company.

[Note: Here only brief information has been given about the product. The details regarding the software's background is given in the Project Proposal done in 81HD task]

Since the software's importance to the company can be quite clearly understood, it is also important for the team producing the software to follow proper standards to overcome all challenges in its development, whilst ensuring its quality. This led to the conundrum on which Software Development Life Cycle (or SDLC) is better for the team to use in developing this software:

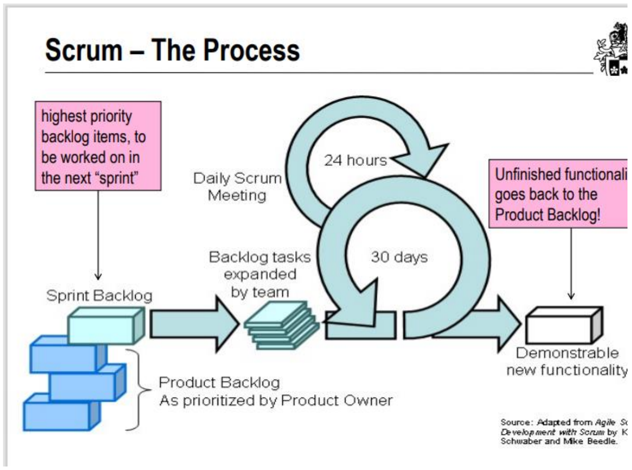
A) Waterfall
B) Scrum

WaterFall
Waterfall is a traditional development method where the project plan is detailed and fixed at the very start, allowing little to no alteration from the plan later on (even if problems appear or client's demands change). That's because, like a waterfall, it follows a sequence of steps from conception to deployment (and maintenance), with each step heavily reliant on the outcome of the previous step, and so, once the process starts, there is no way to change it in any way. Furthermore, it only demonstrates the product in its deployment stage where all of the functionalities detailed in the plan have been completed, and thus even if there have been any miscommunication between the client and the team, the feedback can only be received at the very end, which by then is too late.



(Fig. 1 Waterfall Model SDLC. Adapted from **[2]**)

Scrum
Scrum is an agile development method that is mainly used in developing complex, unknown software whilst ensuring both high quality and also the demand of the software being produced. It is done by breaking the big project down into small chunks where each would be completed in short fixed timeframes called sprints with short daily meetings to ensure things are on track. Furthermore, at end of each sprint, the team also demonstrates the functionality produced or the work done with the clients (who are also considered part of the process) to assure them of the software's quality and also gain their feedback on whether to proceed or alter with the pre-decided project plan.


Scrum – The Process
highest priority backlog items, to be worked on in the next "sprint"
Daily Scrum Meeting
24 hours
30 days
Unfinished functionali goes back to the Product Backlog!
Backlog tasks expanded by team
Sprint Backlog
Demonstrable new functionality
Product Backlog As prioritized by Product Owner
Source: Adapted from Agile S Development with Scrum by K Schwaber and Mike Beedle.

(Fig. 2 Scrum Process SDLC. Adapted from **[3]**)

Major Challenges faced in SaleManSys:
Although we can already come to a certain conclusion from the general information regarding both processes *(noted in the two paragraphs above)*, it is better to focus on the software being developed (ie the SaleManSys) and go through all the challenges that had been faced by the team in developing the software, one by one, to ensure we have not missed anything. In this "Sales Management system" for company Ragib Televisions (i.e. a company that sells television), the following aspects affected the success of the software:

A) Initial Project Scope Inaccuracy/ Team and client misunderstanding
B) Changing customer demand and client requirements (Scope creep)
C) Time And Resource
D) Internal Team conflicts/ member interaction
E) Testing for quality and deployment time

A) Initial Project Scope Inaccuracy /Team and client misunderstanding:

When a project plan is first developed and accepted, it is almost always filled with inaccuracies. This had been especially seen in the management projects of SaleManSys where client's intentions had been quite vague in some of the functionalities they sought. For instance, they stated that they wanted the system to give them monthly sales details but they never stated on how the information would be displayed. It may be in tabular format, chart format, etc. But, since they only mentioned "monthly sales records" and had not detailed on how it should be presented, the team just assumed they wanted it in table or record form instead. Thus, an issue had been formed where an inaccuracy existed in the scope as the client's requirement for the functionality and the team's understanding of the requirement was not the same, which in turn raised a new challenge.

B) Changing customer demand and client requirements (Scope creep)

Even if scope had been properly communicated and set at the very start, it may not remain static. That's because demands of customers, the facilities of the systems used by their competitors and other environmental factors can lead to change in the scope itself. Thus it wasn't too surprising when the following scenario took place in developing this software:

"In midst of the software's development, the company's competitor started to use a system where their sales records would be analysed by an AI in order to not only portray the current product demands but also predict future demands of their customers. Thus the company itself had suddenly become hard pressed to get a similar system too, so as to not fall behind."

Thus, the scope had now been altered with a new functionality being demanded by the client, raising a new challenge.

C) Time and Resource

Generally, the client would allocate a certain amount of time and resource to the team in order to develop the system. In this project, the client had given the time period of **20** days (or **about a month's time**) for the team to complete the project with the resource being the **amount of money** they would pay the team as compensation. Being able to develop the software under these conditions was also a challenge that needed to be resolved.

D) Internal Team conflicts/ member interaction

Generally, whenever a group of individuals team up in order to accomplish a certain task, conflicts arise for several reasons. This can be due to lack of motivation, availability of multiple "solutions" or ways to accomplish a task, external environmental factors like "truck factor", etc. In this project's case, it was due to technological issues (like lack of certain skills or tools, lack of past knowledge regarding management software development, etc.) and communication issue between team members (like member's misunderstanding each other's instructions) which also posed a challenge that needed to be resolved.

E) Testing for quality and deployment time

All projects require some form of testing in order to ensure that quality has been maintained and then deploy it in time to assure the client that the system is ready to be used. Thus, these had also part of the challenges faced by the team in developing this software too!

In Waterfall, the challenges have been approached (and wherever possible resolved) in the following ways:

- For Challenge A (Initial Project Scope Inaccuracy /Team and client misunderstanding):

In waterfall, the scope had to be fixed or defined at very start (ie in requirement gathering stage) but validated near the end (in the testing stage). So, the team had thus gone through the all the stages basing upon "their interpretation of the client's instruction" (noted in the very first stage) while the client had mistakenly believed that they *[the team]* had completely understood their *[client's]* desires. Hence, during the testing and deployment stage when they finally brought in the client for their feedback **[4]**; the client had been disappointed as the product's functionality hadn't been how they had wanted it. But by then it was already too late as the product had already been built and there hadn't been enough time remaining in order to fix this feature in the project. Thus in the end, although the feature had not been exactly what the client had desired, they had to accept it then and ask for another waterfall cycle to correct the inaccuracy and misunderstanding

- For Challenge B (Changing customer demand and client requirements (Scope creep)):

This had been troublesome for waterfall for the same reason as in the point before as due to its rigidity, once the scope has been set in very first step (ie in requirement gathering stage), it cannot be changed until testing stage by which time the project is mostly completed. That's because each step here is highly dependent in the output of the previous step **[2]** and thus developing the new functionality is not possible in the same cycle. So, the client had to wait until the current waterfall cycle had been finished before requesting for the new functionality and then wait for an even longer period of time (as the team had to go through the same cycle once more). Thus, by the time the system had been developed with the new functionality, the client company had already fallen behind, making it too late for them to use the software with the new functionality to their full benefit **[4].**

- For Challenge C (Time and Resource):

In waterfall, the time and resources for the system had been vigorously discussed about and documented in its very first stage (i.e. requirement gathering stage) which in itself took up a lot of time (as the time and resource must be kept static throughout the project and so must be error-free). Thus, as long as no change occurred in the surroundings or the project scope, the client would be able to predict the exact date when the system would be operational and the exact amount of resources needed to develop it from the very start **[5]**. But when any changes occurred, a lot of resource and time would be needed to address that (as the team would need to go through the same cycle once more to accommodate for the change).

- For Challenge D (Internal Team conflicts/ member interaction):

Waterfall resolved this issue by providing a clear and fixed scope, requirement during its first stage (requirement gathering stage) and again in its design stage when deciding the language, framework and other technical details that would be used in the project **[7].** Since it had spent a long time in clarifying these details in these steps with all the members in the team, there hadn't been any major technical or communication issues (ie team members not understanding what was being asked of them) occurring in the later stages when the software was actually being built.

- For Challenge E (Testing for quality and deployment time):

In waterfall, there was only a single step to test all the functionalities of the software with the specifications to ensure that everything had been built following the client's requirements. This had been done this by maintaining comprehensive checklist for each and every functionality (like coding standards, architecture reviews, etc.) **[4]** and ticking them off one by one. Although it took a long time, it was extremely thorough. Thus, by the time it had gone through the testing stage and had been deployed **[10]**, it was not only completely bug and error free, but had upheld the quality requirements which had been decided upon in the requirement gathering stage.

In Scrum, the challenges have been approached (and wherever possible resolved) in the following ways:

- For Challenge A (Initial Project Scope Inaccuracy /Team and client misunderstanding):

In scrum, the scope is quite flexible and thus can be easily altered after each sprint is over. Furthermore, since the client is also considered as part of the team (as the Product Owner), they are also present in the daily scrum meetings. Thus, the client had been able to clarify the misunderstandings between them and the team quite easily and early on in the software's development. Thus, there had been more than enough time to make the modification, fix the inaccuracy, fulfil the client's requirements completely and maintain their trust.

- For Challenge B (Changing customer demand and client requirements (Scope creep)):

In scrum, no issue exists in changing the scope (which has the list of functionalities to be developed) as each functionality is allocated to certain sprints where feedback from client is obtained at the end of each sprint. So, this change (which had been requested by the client) had been addressed in the end of one sprint (during the client's feedback) and thus had been easily added to the backlog, in order to be developed in one of the later sprints. So, it had been easily resolved by allocating a small amount of time and resource towards developing the functionality and thus was able to fulfil the client's desires completely, without losing their competitive advantage.

- For Challenge C (Time and Resource):

In Scrum, the time and resources needed for the system were quite flexible. That is because each functionality (or small groups of functionality) had been developed in their respective sprints which were "time boxed" **[6]** and thus depending on number and type of functionality required, the time and resourced needed could be easily modified. So, there had been no need to fix the resource and time needed at the very start and instead it could be refined as the software is developed. Although this led to higher levels of unpredictability, it could still accommodate for any change that occurs, with only a smaller increase in time required and resources used.

- For Challenge D (Internal Team conflicts/ member interaction):

In Scrum, this had been resolved in a slightly different way. Here, the team had at first declared the scope and the KoST in their project proposal where they had detailed the technical expertise (like skills, tools, pre-existing knowledge and solutions) required. Also, during each sprint, the team had performed daily meeting where each member had discussed their problems and thus had collaborated with one another to make up for their lack of skill or past knowledge. Furthermore, it had also promoted self-organisation in the team where the team took ownership in not only doing the tasks but also on solving the problem and producing results, which in turn led to production of work of high quality **[8].**

- For Challenge E (Testing for quality and deployment time):

In Scrums case, each functionality (or functionality group) had been tested during their own sprint time (when they had been developed) to ensure their quality before considering them as “done”. This not only had taken lesser time (as the team had tested each functionality the moment they had finished it and thus had been fully aware of where which part of code had been written) but had also avoided the almost exponential spill over effect where each “undone” or untested parts build up over time, which would have led to an extremely long stabilization period (and in worst cases even delaying the deployment) **[11]**

Comparison Outcome:
For challenge A, Scrum had been better than Waterfall method as its scope had been more flexible and it had more team and client interaction (as client is also part of team) and thus could resolve the issue more efficiently and quickly.

For challenge B, Scrum has been better than the Waterfall method. That’s because compared to waterfall, scrum took smaller amount of resource and time to address and resolve the challenge in developing the new functionality and thus allowed the client to retain their competitive advantage with the developed software.

For challenge C, although waterfall had a more predictable time and resource usage, Scrum had been better overall. That’s because Scrum had accounted for practical situations where anything can change at any time **[4].** Thus, although it was more unpredictable, it was better suited for the project as it had been able to handle the change in scope (noted in the previous two points) with just a smaller increase in time and resources used.

For challenge D, although both SDLCs had their own methods to resolve the issue, we can see that Scrum’s method had been slightly better as here as there was far more interaction and motivation amongst the team members whilst doing the project and we already know that effective communication and motivation are major factors in a project’s success **[9].**

For challenge E, although both had their own unique ways to perform testing and had resulted in software of similar quality, the Waterfall SDLC took slightly more time in testing and in reaching deployment stage compared to Scrum. Thus, in this factor, Scrum had been better than Waterfall, albeit by a small margin of time.

Conclusion
We all know that both Waterfall and Scrum are effective ways that can be used develop software. But in the context of the SaleManSys, we can see that Scrum is more suitable for it in terms of practicality (and also risk management when considering scope change), flexibility, shorter timeframe, more team interaction, more client trust and good quality of the software produced.

But does this mean it will be the same for all software projects?

Not necessarily. That’s because although Waterfall is traditional and inflexible, it does have it merits too. Thus if a project needs to be done where there is well-defined and fixed scope, need little client interaction and need to be done in a fixed and predictable timeline with a fixed resource usage (or budget), it will be better for it to be done with waterfall instead, as it is more suitable for it.

So, in conclusion, although Scrum SDLC is better than waterfall in terms of developing this software (SaleManSys), this outcome cannot be used to show that scrum is superior to waterfall for all types of project. Instead it would be better to repeat this study in context of other projects before determining the overall conclusion on which SDLC is the best for all software.

References

1) N. Glentworth. "How small businesses can stay on the front foot of digital transformation". NEIL GLENTWORTH. https://www.neilglentworth.com/commentary/2021/9/6/how-small-businesses-can-stay-on-the-front-foot-of-digital-transformation (Accessed May 16, 2022)

2) Tutorialspoint. "SDLC - Waterfall Model". SDLC. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm (Accessed May 16, 2022).

3) Dr. M.L. Edmonds. (2022). Swe20001 lecture 1b Scrum. [Pdf] Available: https://swinburne.instructure.com/courses/40777/pages/week-1-sdlc?module_item_id=2705766

4) S. Shekhar. "Software Development Life Cycle (SDLC)". Linkedin Learning. https://www.linkedin.com/learning/software-development-life-cycle-sdlc/waterfall-model-brief-overview?autoplay=true&resume=false&u=2091708 (Accessed May 16,2022)

5) L. Hoory, C. Bottorff. "Agile vs. Waterfall: Which Project Management Methodology Is Best For You?".Forbes ADVISOR. https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology (Accessed May 16,2022)

6) M. Rehkopf. "Sprints". ATLASSIAN Agile Coach. https://www.atlassian.com/agile/scrum/sprints (Accessed May 16, 2022)

7) M. Martin. "What is Waterfall Model in SDLC? Advantages and Disadvantages". Guru99. https://www.guru99.com/what-is-sdlc-or-waterfall-model.html (Accessed May 16, 2022)

8) Fowler, Frederik. (2019). Navigating Hybrid Scrum Environments: Understanding the Essentials, Avoiding the Pitfalls. https://doi.org/10.1007/978-1-4842-4164-6

9) H. Estler et al, "Agile vs. structured distributed software development: A case study," Empirical Software Engineering, vol. 19, (5), pp. 1197-

1224, 2014. Available: https://www.proquest.com/scholarly-journals/agile-vs-structured-distributed-software/docview/1564862174/se-2?accountid=14205. DOI: https://doi.org/10.1007/s10664-013-9271-y.

10) R. Sherman, Chapter 18 - Project Management, Editor(s): Rick Sherman, Business Intelligence Guidebook, Morgan Kaufmann, 2015, Pages 449-492, ISBN 9780124114616, https://doi.org/10.1016/B978-0-12-411461-6.00018-6. (https://www.sciencedirect.com/science/article/pii/B9780124114616000186)

11) K. Schwaber. "Succeeding With Scrum - a conference by Ken Schwaber (part 1/3)". Youtube. https://www.youtube.com/watch?v=VCzIFn8vt_c (Accessed May 16, 2022)