Comparison of agile software engineering processes

November 5, 2021

Contents

1 What is Agile?

Following the failed projects from the families of waterfall and its subsequent alternatives, the emergence of agile methodologies presented a new direction for software development; one which focused on process rather than strictly code, like its predecessors. The aim of an agile process is to create order, establishing the steps involved as interacting networks in which control comes from linking activities, as opposed to rigid, sequential procedures. Such processes allow us to organise, regulate and control the workload of the Software Development Life Cycle (SDLC).

1.1 Introduction to processes

There is a large number of agile methodologies, some of the more common ones include eXtreme Programming (XP), Scrum, Crystal, Feature Driven Development (FDD), Rational Unified Process (RUP), Dynamic Systems Development Method (DSDM) and Pragmatic Programming (Anwer et al., 2017). This report will be focusing on XP, scrum, crystal and DSDM methodologies.

1.1.1 XP

With XP, as with any agile process, there is an emphasis on visibility and customer involvement. By creating scenarios and use cases, as well as testing and implementation, both parties are able to refine the agreed needs of the project before embarking on new additions. Refactoring and importantly, repetition allows the development of the software, at which stage the larger team divides into pairs to tackle both the small tasks of the project as well as compartmentalisation of responsibilities (Yasvi, 2019). As suggested by Brooks (1975), "division of labour is what causes problems in large programming projects".

1.1.2 Scrum

The Scrum method includes 3 distinct, but rotatable roles: the product owner, the development team and the scrum master. The development team is organised based on abilities and as a result covers a range of duties. It is the product owner's role to optimise the value of their work, as well as managing product backlog, which may comprise tasks such as identifying and ordering items, and ensuring visibility and understanding of the backlog. Finally, the scrum master ensures that all members adhere to scrum practices (Rubin, 2013).
Within Scrum, the features of transparency, inspection and adaptation form the 3 pillars of process control. Transparency refers to the fact that the whole team must be aware, and in agreement of, any changes which may take place, allowing for the progress to be visible to all. Inspection entails reviewing documents produced from sprint, as well as overseeing the overall progress towards reaching goals. This is a crucial element in managing the process, however,it must be noted that care should be taken so that inspection does not itself become an obstacle in completing the work. Adaptation identifies the risks in the project and makes sure that the process stays on track. Here, there is the chance to act on the issues raised in the review process (Rubin, 2013).
As touched on before, sprint is a feature of Scrum and is described as a short timebox period by which the development team will have created a 'done' and usable artefact. Its activities include planning ($<$8 hrs), daily scrum (15 mins), sprint review ($<$4 hrs) and sprint retrospective (after product delivery).

1.1.3 Crystal

Across the crystal family, the process comprises a number of cycles, subsequently reducing in terms of time-frame (Singh, 2021). The first, which spans the entire timeline, is the project cycle. The next is the delivery cycle in which shows progress and delivery of the final product, taking from a week to 3 months. Following this, the iteration cycle involves estimation, development and testing covering the same time period. The work week divides the project, and all the work done, into 7 day periods. The integration cycle is similar to iteration, but lasting only up to 3 days as opposed to the three months for the former.

The work day shows the work done each day and finally the episode is the base unit within the project, in which designing, developing and testing of a section of code is done. Hence, all methodologies within this process use incremental development given the way in which the project is broken down.
Throughout the process, there is an abundance of guidance for users, down to policy standards during development. These are not dissimilar to features and techniques mentioned previously in the context of other processes, including but not limited to, incremental development and delivery of the tested product, customer involvement and a system of methodology adaptation following review and feedback (Anwer et al., 2017).

1.1.4 DSDM

DSDM provides the framework for Agile project management for small projects, all the way up to corporate projects. The basis of DSDM is to provide just enough content in each stage to be able to move onto the next (*Processes*, 2021). Inadequate detail in the stages is managed by subsequent iterations.
This process consists of phases which cover definition of objectives (pre-project phase), determining feasibility (feasibility phase) and development of requirements and a high level description of the project outline between team and customer (business study). After deployment, the post-project phase produces a review of the project, generating overall or individual feedback (Anwer et al., 2017).
There are many roles within DSDM which vary within organisations. However, the main responsibilities include developing, testing and analysing the programme (developer), liaising progress and questions between user and developer team (ambassador/advisor user), identifying and prioritising system requirements as well as ensuring these are met (visionary) (Anwer et al., 2017).

1.2 Comparing XP, Scrum, Crystal and DSDM

XP: XP is a cost and time effective method where coding is the main goal, meaning that certain activities used in other methods are eliminated. Due to the developers' main focus being code, it allows for faster software delivery and permits more time for testing and alterations. Regular testing ensures that only what the customer wants is being developed (Yasvi, 2019). This process is good on qualitative metrics but it is unclear how these steps are organised. Since there is no design technique, this may cause issues between the programmers on what the final picture should be, and may even lead to the development of the software going in the wrong direction (Ahmad and Dalalah, 2014).

Scrum: Scrum has an advantage over methods, such as Crystal, with large projects. Sizeable projects are broken down into comfortably manageable sprints. Methods such as Crystal could not take on such projects as, for Crystal to work, each team member has to be available at the same location to work. Scrum meetings, which occur frequently, present each team member's progress within the project (Chandana, 2021). However, these frequent meetings may irritate some members. Furthermore, this method suffers from encapsulation, meaning that if a team member leaves the project or is absent, this can have a negative effect as the other members may not be able to perform that task (Chandana, 2021). XP overcomes this issue as this method requires pair-work, so when one member is absent, the other one can carry on making progress within the project.
XP and Scrum are often used together as they complement each other's needs, with one making up for the pitfalls of the other.

Crystal: Crystal is a very flexible method which adapts to projects and team requirements regarding size. Other methods mentioned, such as Scrum, are more strict when it comes to changes. For example, when the sprint is reached in Scrum, no more changes are permitted. In addition to this, the project budget is firmly set prior to the start which allows for better planning and therefore decreases the chances of any budget-related issues. Comparing this to Scrum, the clients within this methodology can demand more features, increasing the price of the project and potentially leading to the project becoming unfeasible (Singh, 2021). Another consideration is a clear definition of the processes. Crystal as a methodology

is defined poorly, with only Crystal clear and Crystal orange being clarified out of the 4. Also, team organisation is not clear-cut (Anwer et al., 2017).

DSDM: This method can be seen as a more rapid and robust way of software management compared to others. Also, three key properties of this method are that there is a fixed cost, time and quality (*Processes*, 2021). Proper guidelines are in place within DSDM such as development techniques or management, which is advantageous over methods such as XP or Crystal.
However, there are many roles in DSDM, which has the potential to lead to administrative misunderstandings and confusions (Anwer et al., 2017) This method also requires heavy documentation to take place, which causes the delivery of the project to be much slower than some of the other methods mentioned (Ltd, 2021).

Method	Positive	Negative
XP	Great visibility Customer involvement No issue of encapsulation	No design technique Not ideal for large teams
Scrum	Breaks large projects into manageable sprints Transparent for all members	Encapsulation Very frequent meetings
Crystal	Flexible Budget set early on	Methodology defined poorly No clear team organisation
DSDM	Rapid and robust software management Proper guidelines	Many roles cause administrative issues Heavy documentation

Table 1: Comparison of processes

2 Tool support for pair/team working

Jacobson (1998) states "tools are integral to process". Nowadays, when it comes to developing software projects for research, business, government, and other fields, it is inevitable that many kinds of tools would be used during the development processes. Therefore, since within the agile process tools are not negligible, when using agile methodologies to create software, there, too, will be some tools supporting development work.
Here, tools that support pair/team working during the development processes by applying agile methodologies will be investigated. The analysis will focus on aspects including functionally oriented tools (visual modeling, implementation, and testing) and project management tools which support SDLC (Jacobson, 1998).
When considering functionally oriented tools, these can be divided into different categories based on their different functions, and more importantly, they are applied in different development stages. As for SDLC project management tools, these matter more on global functions rather than specific aspects of the whole development. Following their introduction, we will analyse the positives and negatives of these tools and choose and conclude which tool is most suitable for cooperation among a team with almost ten people, as this is the team size for coursework 2.

2.1 Project management tools for agile teams

Project management tools are most representative of cooperation among agile teams since they are applied almost throughout the whole development process, and even within new system development versions. The main functions of project management tools are listed below:

1. Requirement management - (a) Task sharing, assigning and recording (b) Planning for deadlines (c) Reporting progress

2. Communication (a) Sharing thoughts with teams (b) Exchanging information

3. Quality assurance (a) Reporting issues and bugs (b) Automatically generating report files (c) Evaluating team performance

Applications	Platform	Positives	Negatives	Cost
Jira	Web/Mobile	Customisable workflows Enough features for agile	Complicated high learning cost Limited file uploading(10 MB)	Multiple price choice/Free trial/Free for 10 users
Trello	Web/Mobile/Desktop	Simple UI Kanban System Flexible with many templates	No review iterations Not enough storage Limited features for free	Multiple price choice/ Free trial/Free for 10 users
Clickup	Web/Mobile/Desktop	1000+ integrations from other apps Internal communication Customisable Simple UI	New users can feel overwhelmed	Multiple price choice/ Free trial

Table 2: Summary of project management tool comparison

2.2 Visual modeling tools

Visual modeling is an approach to requirement analysing and system model design. The objective is to define and create models that could meet the system requirements and solution domain and prepare implementation and test processes. Here, Unified Modeling Language (UML) will be discussed as well as the tools used alongside it. The UML is a family of graphical notations, backed by a single meta-model, that helps in describing and designing software systems (Fowler, 1999).

Tool names	Functions	Positives	Negatives
Lucidchart	Programme design Uniform naming rules Translate requirements into diagrams	Drag and drop input Team sharing Many templates to start Easy export to pdf Price from free	UI is not beautiful No desktop app
Visual paradigm		IDE integration Powerful features	UI is not beautiful Price from £4.50/month
Microsoft vision		Easy to learn Team sharing Many templates to start Powerful features	Licence fee is expensive

Table 3: Summary of tool comparison

2.3 Implementation tools

The objective of implementation is to deliver the system components including source code, scripts, executables, and so on. During the process, developers have to use a range of tools like editors, compilers, debuggers, error detectors, and performance analysers (Jacobson, 1998). Fortunately for developers, Integrated Development Environments (IDEs) such as Eclipse, IntelliJ IDEA, PyCharm and others integrate all these tools. However, the developers' selection of IDEs is based on programming language preference, familiarity with IDEs, and so on. Therefore, the IDEs will not be illustrated since they all have their own subjective strengths. Instead, the source code and version control tools will be introduced, which are Git and SVN.

Type	Tool Names	Functions	Difference	Positives	Negatives
Git	Github Gitlab SourceTree (GUI Tool)	Code tracking Code review Version control Co-develop	Git is a distributed version control system	Enhance collaboration between team members Allow different coders to edit the same code with track	High learning cost
SVN	Cornerstone TortoiseSVN		SVN is a centralised version control system	Easy to rollback code version Gitflow	

Table 4: Summary of implementation tool comparison

2.4 Testing tools

The objective of this technique is to test applications and system components, so as to find and fix bugs, to keep the system operating efficiently. Although the decision of which programming language will be used in the implementation of the CW2 has not yet been made, Objective Oriented Programming (OOP) is recommended since some primary programming languages like JAVA, C++ and Python are all object-oriented. Unit testing is the main technique of OOP. Apart from unit testing, manual tests, which can be used in complement with unit tests, are also important for system development. Olan (2003) suggest that "unit tests invoke one or more methods from a class to produce observable results that are verified automatically". As for manual tests, the project management tools mentioned above could be used for bug reporting and communication between team members. Table 5 below shows JUnit and PyUnit which are testing tools based on different languages.

Type	Language	Functions	Positives	Negatives
JUnit	Java	Testing during incremental developing	Test automatically using cases	Increase workload
PyUnit	Python	Code review	Increase code quality	Learning cost

Table 5: Summary of testing tool comparison

3 Practices to support pair/team working

In this section of the report, the specific techniques used in supporting team work will be presented for each of the processes.

3.1 XP

Collective ownership In XP, pair working is applied in all areas of the system development. Pairs of developers take charge of the whole system and are responsible for all of the code, and anyone could change any single line of code.

Pair programming In this technique, developers work in pairs to develop the same requirement at the same time, coding together face to face. This allows for them to check and review each other's work and offer suggestions and support to finish the job as perfect as possible.

Refactoring This is essentially a practice which revisits and allows for the rebuilding of code. All system developers are supposed to refactor code sustainably whenever some code with room for improvement is found. Crucially, they should also work together to keep code clear, easy to read and maintainable.

Testing Practices for testing differ, where unit tests are created by programmers, and acceptance tests are created by customers.

3.2 Scrum

Daily meeting Scrum team members get together and have a meeting to share information, describe development progress and identify task priorities, sometimes even bringing new problems arising from the real development process. For co-located teams, scrums should ideally be face-to-face. However, it could also be that an online meeting might be more adequate in situations of location restrictions such as the COVID-19 lockdown.

Scrum board The scrum board could be a real whiteboard in an office or a virtual/online board, like with Kanban, on which information can be attached, or removed, by anyone in the team. Such information may include, sprint backlog, task deadlines and a record of which developers are free or busy at any given time. This increases the visibility of the development, as members are able to see the states and progress of other team members and know how much work is underway or done. Moreover, it can be a good way for the scrum master to get a global vision of the system development, and arrange resources to improve cooperation and efficiency. For stakeholders, the scrum board may serve as a monitoring tool and proves a good way to achieve customer involvement and transparency.

Sprint review meeting This kind of meeting takes place at the end of each sprint, involving the whole team. Team members gain experience from the last sprint, discuss how things could be done better with other members, and then prepare for the next sprint.

3.3 Crystal

Anwer et al. (2017) state that "Crystal methods focus on people and communication among people rather than process to frequently deliver a working software". This is attractive to teams due to scalability. **Communication practices in crystal clear:** designed for teams with 6 members, the team members have to be located at the same place because they must have discussions everyday about requirements, design and other things pertaining to the project. **Communication practices in crystal orange:** the project has 20 to 40 members. These people are not in the same team and more than one team works on the same project for different iteration tasks. This introduces communication between teams, moreover, teams have to communicate efficiently so that they can handle the changing requirements.

3.4 DSDM

There are two principles of DSDM which guarantee cooperation in teamwork: (a) Collaborating and cooperating with each other, (b) Communicating continuously to get feedback

Business study Before this phase, requirements from customers are almost settled on. The priorities of requirements are agreed upon through communication between customers and the development team during this stage. Additionally, prototype plan would be made together.

User involvement The user supports the team throughout the whole of development, ergo communication is essential between the development team and users. For instance, an ambassador user could be regarded as a bridge between the users and development teams. They convey users' needs to the development team and pass the development progress to users as feedback (Anwer et al., 2017).

Cooperation between stakeholders Developers and stakeholders should share the responsibilities. Cooperation between the two is essential as stakeholders are important decision makers in deciding what to develop.

4 Justified selections of process, techniques and tools for the second coursework

Supposing that we will have an 8 member team in coursework 2, which could be considered as a medium-sized team compared to the pairs in coursework 1. According to the processes, tools and techniques discussed, we plan to adopt Scrum as our main frame, along with some suitable practices from XP, like pair programming. The reasons are listed below:

- The team needs management and Scrum is a management-focused agile method. Since the team is formed by students who are not equipped with much product development experience, Scrum has a mature workflow which could be followed and its activities would help ensure team member cooperation.
- In XP, changes can be accepted at any stage and requirement changes are not appropriately documented. This would complicate product maintenance for the developers. Also, XP requires face-to-face communication which is hard to achieve, but it could be a good activity if made more flexible.
- Crystal, although having a huge range of methods covering variable team sizes, does not provide well-defined team structure. This would easily introduce confusion in role assignment and running the team.
- DSDM does not provide specific guidance about issues related to team size and iteration length (Anwer et al., 2017) and there are often fifteen roles in DSDM which could not be fulfilled by an 8-member sized team.

4.1 Detailed plan, including selected tools and different role duties

Here, as an example, the whole iteration process is illustrated, including specific tools, techniques and roles to be used (Figure 1).

Set up and planning The product owner invites all members to a project on Jira. They collect user stories including product features, requirements and system improvements which are documented into the ”Backlog” section. Prototype diagrams describing how the game works are drawn and shared using Lucidchart. This is done before the planning meeting to help others get a brief understanding of the new features and solutions.
The whole team has a planning meeting where user stories are split into designing, coding and testing tasks. Feasibility is discussed, as well as time allocation, task deadlines and task priorities. After the meeting, the product owner documents all agreed upon tasks into the ”Project page” section as ”Project requirements” on Jira. These are then assigned to a relevant team member. If some user stories could not be agreed upon by the team, the product owner should communicate with the customer again, negotiate, and then have a planning meeting again to settle them or put them into the next sprint.

The sprint When starting the sprint, the time period depends on the number of tasks and the result of the planning meeting. Since coursework 2 is from week 6 to week 11, there may be only one or two sprints.
During the sprint, the team holds daily meetings to review progress. Anyone can share information and report issues, also task priorities may be altered depending on the real development situations. Every member should update the status of their own task timely on the ”Board” section (whiteboard) so that others in the team can access the latest information. There are default statuses including ”TO DO”, ”IN PROGRESS” and ”DONE” in the ”Board” section, and these are customisable. The team can create new, edit, or remove existing statuses in line with the real needs. The ”Board” section could be replaced by Trello which has similar functions and a mobile app which could facilitate information sharing.

Role assignment/duties The development team is significant, and should have the majority of team members, including designers, coders and testers. After the planning meeting, the development team should devote themselves to developing and dealing with their tasks.
Designers in charge of the user interface, graphical images or even 3D models, could use tools like Photoshop, Sketch and so on, to offer image materials to thhe coders. These should not wait until the materials are passed to them to start working. Instead, according to the prototype diagrams and tasks, they should use Lucidchart to design the database and define the models, and what properties and functions should belong to every model (OOP programming). Following this, they should keep developing the tasks from high priority to low.
In this phase, Github will be adopted as the code control tool. Based on the master trunk of the code, every coder is in charge of their own tasks and can develop on their own branch, then merge them to the master trunk. This method helps coders work together and will not influence others' work if the tasks are divided into appropriate function modules. If there are some difficult features, pair programming could be applied to attempt to enhance the problem fixing power.
As for the testers, they should write unit tests, using JUnit or PyUnit, immediately after the planning meeting according to every feature. Of course, once the game is made, everyone in the team should take part in the testing work. Hence, just one person could become the tester in charge of the unit test programming.

Iteration At the end of each sprint, the team holds a review meeting and the whole team should be involved. The purpose is to discuss the last sprint and which activity in the whole process could have been done better; not only regarding the product, but also individual work and/or cooperation among the team.

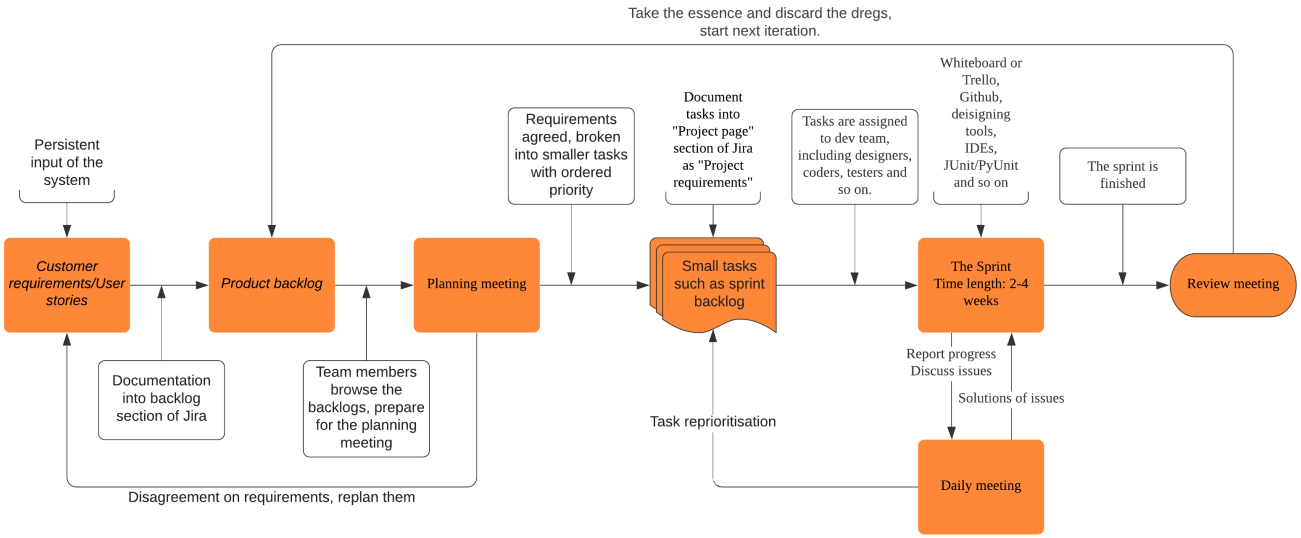

Figure 1: Flowchart summarising Scrum

5 Risk assessment for coursework 2

5.1 What may go wrong

Due to a tight deadline and a busy student schedule, the team may encounter issues such as running out of time. Further, the members all have different past experiences when it comes to software development or coding, varying from little experience, to working in industry. This may cause issues such as some less experienced members taking on tasks which initially may seem handable, but turn out not to be, or more

experienced members having to split their responsibilities to help out in other sections of the project.
Within this part of the coursework, the customer might change their needs and requirements for their product, known as scope variation (*Scope change request*, 2020). The team will be expected to engage effectively to adapt, meaning that rapid changes might have to be made. This could cause communication problems and some members could start working on the changes by themselves without properly informing the rest of the team. Such events could eventually lead to failures within the project and, due to the lack of communication, lack of ownership and accountability would ensue, further complicating the process of identifying and rectifying the initial source of the problem.
Finally, it must be mentioned that the code has to be clear and of high quality. Any bugs which arise must be fixed and comments should be made throughout the code to allow for better understanding.
Table 6 describes the potential risks and possible solutions to these.

Potential risks	Possible solutions
Inaccurate time estimations	Stick to strict, clear deadlines from the start
Scope variations	Agile practices such as iteration or prioritisation
End-user engagement, meeting expectations	User survey, confirming what is required in meetings
Poor quality code eg bugs, poor readability	Tests and code reviews
Lack of ownership	Document meetings, stick to transparency

Table 6: Potential risks and possible solutions

6 Pair working process

For the first few days of the assignment being set, both authors spent completing further research into the background of agile methodologies. A plan with a deadline was set, where author 1 would look into completing CR1 and CR2 while author 2 started on CR3 and CR4. Within the software engineering computer sessions, the two authors discussed what each other has written in those sections. Any agreed on additions or changes were made here.
Once these sections were completed up to an agreed standard, author 2 completed CR5 as the two authors settled that this would be favourable, due to this author having hands-on industry experience with agile methodologies, allowing them to write from experience. Concurrently, author 1 completed CR6 and CR7. CR8 was completed by both authors simultaneously in the software engineering group session on the Wednesday prior to the submission.
Both authors contributed equally to the addition of further information to each section as well as forming the document itself. In the last step the report was read together and finalised so that the text would flow smoothly from one section to the next, making sure that no information is repeated to maximise readability and space.

7 Critical analysis of the experience of pair working

During the project, both authors decided to split the content requirements in a bid to speed up the process. This approach was chosen as one pair member expressed they though that pair programming was a waste of productivity to some extent. Though no issues arose, in the context of a different team, contradictions may have become an issue if the pair were to disagree and fail to meet a compromise. At the end of each section, the authors read through each others work. This allowed to supplement the text from different perspectives. It became evident that there were different writing styles. In light of this one of the members was appointed as the editor. Modifications had to be approved by both parties throughout the project. This elevated the quality, however other teams may disagree on changes.
In future teamwork, the report writing could be improved by methods suggested by Matsudaira (2018) such as engaging with other people performing the same task and talking about it.

References

Ahmad and Dalalah, 2014. Extreme programming: strengths and weaknesses.

Anwer, F., Aftab, S., Waheed, U., and Muhammad, S.S., 2017. Agile software development models tdd, fdd, dsdm, and crystal methods: a survey. *International journal of multidisciplinary sciences and engineering*, 8(2), pp.1–10.

Brooks, F.P., 1975. *The mythical man-month – essays on software-engineering*. Addison-Wesley.

Chandana, 2021. *Scrum project management: advantages and disadvantages*. Simplilearn. Available from: https://www.simplilearn.com/scrum-project-management-article.

Fowler, M., 1999. *Uml distilled: applying the standard object modeling language*. 2nd ed., Object technology series UML distilled. Place of publication not identified: Addison Wesley Professional.

Jacobson, I., 1998. *The unified software development process*, Addison-Wesley object technology series. Reading, Mass. ; Harlow: Addison-Wesley.

Ltd, A.A., 2021. *Dynamic systems development methodology*. UK Essays. Available from: https://www.ukessays.com/essays/information-systems/dynamic-systems-development-methodology.php.

Matsudaira, K., 2018. How to get things done when you don't feel like it. *Communications of the acm*, 61(12), pp.52–54.

Olan, M., 2003. Unit testing: test early, test often. *Journal of computing sciences in colleges*, 19(2), pp.319–328.

Processes, 2021 [Online]. Available from: https://www.agilebusiness.org/general/custom.asp?page=ProjectFramework_06_Process.

Rubin, K.S., 2013. *Essential scrum : a practical guide to the most popular agile process*. 1st edition. Place of publication not identified: Addison Wesley.

Scope change request, 2020. Available from: https://uplandsoftware.com/psa/resources/glossary/scope-change-request/.

Singh, V., 2021. *What is crystal method in agile and how it is different?* Available from: https://www.toolsqa.com/agile/crystal-method/.

Yasvi, M., 2019. Review on extreme programming-xp.