Method Selection and Planning
Eng1 Group 29

Adam Hewlett
Ani Thomas
Dan Kirkpatrick
Dominik Hagowski
Matthew Crompton
Niko Chen

Software Engineering Methods, Development Tools, and Collaboration Tools

Plan-driven methods provide a structured way to create software by predicting and anticipating all future features that may be implemented. Planning of the project can be promoted by providing comprehensive explanations of tasks, workflows and responsibilities [1]. There is an emphasis on continued review and risk management analysis. This detailed level of planning requires the project to be developed in a stable environment and is mainly used in projects that have a long working period [2]. Plan-driven methods that were researched include Capability Maturity Model and Team Software Process (TSP). The focus of TSP is to create an environment that supports disciplined work and maintaining a self-directed team, that would help to establish an effective management and structure to the project. However, with TSP requiring the need for experience of managerial skills and Plan Driven Methodology not catering for the project's need for flexibility, the use of Plan-Driven methods was not chosen.

Soft System methods look to break down complex problems in order to truly understand the problem outlined by the stakeholder, and use this to develop multiple conceptual models that can be compared to the original real world problems[3]. The time taken to understand this problem and discuss potential solutions is a lengthy procedure that takes weeks. The model has limitations in comparing the conceptual model to the real world problem and it does not provide a framework for how the design should look, instead focusing on the search process for the best solution[4]. The time-constraint and potential need to integrate other models meant this plan was not a rewarding model to use this project[5].

Agile methodology focuses on the "concept of ongoing waves or sprints of project planning and execution" enabling the project team to adapt the design, scope and execution of the project deliverable. Code is distributed in minor releases with short periods, where the project team works together with the customer in close cooperation[6]. The approach is easy to understand and its core principles were aligned to the scope of the project and the stakeholders requirements. The focus on frequent software delivery allows the development team to produce the product in a timely manner, and this allows iterations and innovative development of functionality of the system. Therefore this methodology was chosen, with sprint lengths consisting of weeks.

Development and collaboration tools

One of the development tools we have chosen to use is libgdx. Libgdx is a java game design framework that we have picked for its rapid prototyping capabilities and its wide platform compatibility. We are an inexperienced team and a game project can be very complex, this makes being able to quickly iterate vital in producing a product that fits together in the end. We also wanted to target a wide range of platforms, since the game could be run on various different places such as Windows, Linux or MacOS we needed an easy way of accommodating this and libGDX provides the means to easily support a wide range.

The main collaborative tool we use is git and github. Git acts as our version control among the team and allows us to collaborate on the project at the same time without any frustrating merge issues that plagued other version control systems. Github is our online git repository and its where all of our code is pushed to and where our website is run out of with the github

pages feature, which allows us to create a website easily in the same workflow that we use to work on the game itself minimising how many new things we have to learn just for the website.

For communication we will use discord to communicate what progress individuals have made as well as ask queries and seek assistance. Discord is optimal as it is well known to all team members and offers voice chat functionality as well as text chat. Furthermore it offers the ability to share screens that us very useful for when tackling tasks collaboratively

To create UMLs SmartDraw will be used to create the architecture diagrams for the game. It allows for early discussion of basic entities to be conceptualised and the relationships between them before the implementation process is started. The tool is easy to learn, and is partially free.

PixelArt will be used in order to create game assets such as the salad bowl. PixelArt will be used due to the simplicity of it and the fact that the game did not require high quality assets. Also PixelArt is free and one of our team members was familiar with the program already.

Team Organisation

The team is organised with coordinators. These individuals play a role in organising the team and suggesting ideas as to what people should complete that week. If someone doesn't feel comfortable with completing the assigned task then there is some flexibility that allows tasks to switch among different members. This ensures that everyone in the group can play a role in completing the game and its deliverables, and hopefully ensures relatively even distribution of tasks among team members.

We have also created a weekly plan, this contains a rough outline of what should be done that week by everyone. This allows the team to easily check what should be done that week, and gives everyone an idea of if we are working to the correct time scale to ensure that we don't fall behind.

When someone gets stuck on a certain section, they can mention this in the practicals or on discord. Someone will then look at what they got stuck on and suggest ideas for what could be done. This is beneficial for the project as we hopefully won't get bogged down on a specific task, and allows the individual who got stuck time to move onto something else, ultimately reducing the time wasted.

Once an assigned piece of work is completed, the team members look over what has been completed and if they think something needs to change they will discuss this with the group, if the change is deemed necessary/appropriate then it can be completed. This is beneficial for the team as there is less pressure on what the individual has completed, and ensures that we don't miss any key points that need to be touched on in the assessment.

Planning

Assessment 1

Assignment 1 turn in date: Wednesday, 1st February, 2023 [9 weeks from 1 Dec]

Task priority was decided based on:
1）The time taken to compete.
2）The number of marks it was worth and the number of people it required, if related to documentation.

Assessment 1 Plan Proposal:

W1 - 28/11/22 - We each individually make a libgdx game to all become familiar with libgdx, java, git, etc. We start preparing the website and discuss what the website should contain and we make this work for us. The website is a key deliverable because it will determine the general direction of our team, and the problems that may arise in the future may appear around it, such as version incompatibility meaning some code may not run, etc.

W2 - 05/12/22 - Having experienced programming with java and libgdx we can talk about the more detailed architecture of the game and how we will program the project, also working on things like the uml diagrams and documentation. As well as making a more detailed plan and more detailed division of labour. We will start to prepare the requirements and some questions to ask the "user", which will give us direction for the needs and requirements of the game. The requirements will be our first understanding of user needs. We need to understand user needs to a certain extent, and give some of our own understanding and ideas, and then communicate with users based on our understanding of whether the software we make meets the needs of users as consistently as possible.

W3 - 12/12/22 - Basic character movement and Map loading. The most fundamental things to do are probably character movement and map loading, once we can get a character moving around a blank map we can all work on top of it. We started to download and install libgdx, and started to get familiar with how to make a game. Some people also should prepare method selection and planning, as discussed in our plan, and also work on the risk management deliverable. Some problems we may encounter could be being absent from work due to illness, and making alternative plans, etc.

W4 - 19/12/22 - A way to handle items in the game, so cooks can carry things around in a holding stack and can take ingredients out of cupboards, etc. Alongside the basic infrastructure needed for item handling (e.g. making it easy to add new items to the game). Spending additional time on the assignment on the weekends and discussing anything we have to do, should ensure we can complete the week's tasks.

W5 - 26/12/22 - Add a framework for workstations where with an action and/or time an ingredient is converted to another ingredient (e.g. a cutting area that can transform lettuce into cut lettuce using a cut action). In addition to this we should do further work on the deliverables, to ensure they meet the requirements set out in the brief.

W6 - 02/01/23 - A counter where orders are handed in and food is handed out and an infrastructure for customers. This part of the code is key in meeting the brief, as the customers in the game need to receive the food that has been produced by the player. This depends on the previous work such as work on the ingredient handling and the work to add workstations that allow cutting, etc.

W7 - 09/01/23 - Exam week. Work will be paused during this week which will allow us to focus on our exams.

W8 - 16/01/23 - A main menu, end screens, a way to set up the scenario mode before starting the game, as well as things to support the eventual health system. The priority during this week should be the main menu and the end screen. The main menu should be inviting to encourage the player to start the game, and the end screen is key, we want the player to keep playing the game. Ensuring that this screen is rewarding would encourage them to come back and play the game again. This does depend on if the development of the game is completed, as we need to ensure that the end screen can be reached and that the menu screen can start the game.

W9 - 23/01/23 Space for bug fixing and potential stretch time if work is not done on schedule.

W10 - 30/01/23 - Further stretch time, and submission time. We will check over the code and other deliverables to ensure that they are up to date, and we don’t miss any key marking points. We have also allowed more time if work isn't completed to schedule.

Project Evolution
By developing a concise and well laid out project plan near the start of our project, few changes were made over the course of the project.

One part of the plan that had to change throughout the project was the overall delay in writing the code. This was because we were yet to finish the key deliverables that could aid us in the development period. Another factor that contributed to this delay was the risk, defined in Risk Assessment and Mitigation, R6 which is described as, team members missing sessions due to illness/prior commitments. Throughout the development period some time was missed due to illness, this time was however caught up when peoples illnesses cleared.

We ran into issues in W4 concerning the creation of multiple food stations, where it was hard to tell the difference between them apart. Due to this, Our development team decided to create a fridge instead of multiple food stations which the user could interact with to get multiple different ingredients.

Additionally, the plan had to change when we ran into a problem where the game wouldn't scale correctly, risk R13 (different display sizes (laptop/TV) could cause graphics to worsen and the game not look good.), during W9. During this week we were testing the game on different desktops to ensure the games behaviour was consistent. On some team members' desktops the game wouldn't scale correctly meaning that the graphics were cropped and the stations couldn't be reached. For example, the fridge, inventory, and burger making stations all couldn't be seen and therefore used. A fair amount of time was spent by the team working out a potential solution and this pushed the plan back a little.

During W5 we ran into an issue with merge conflicts, which relates to R14: Merging of branches on GitHub may not be completed correctly. Work was being completed on two different branches and when the time came to merge them both into the main branch, there was a problem with merge conflicts. One of the team members was assigned to work on this, and after some time had passed the merge was completed successfully. This didn't affect the plan too much as one of the developers could continue to program, whilst another worked on merging the branches.

Overall, the evolution of the plan and project didn't change too much. Most of the problems we ran into put us behind schedule, but because we created an in depth plan which included stretch time in W9 and W10 we were able to complete Piazza Panic within the time period assigned.

References

[1] H. Svensson, Developing support for agile and plan-driven methods, PhD dissertation, KTH, Stockholm, 2005, p.26

[2] No title
https://www.se.rit.edu/~se456/slides/PlanDrivenMethodologies.pdf

[3] S.Burge, “An Overview of the Soft Systems Methodology”, System Thinking: Approaches and Methodologies, 2015, pp.1-14

[4] (2018, September.11), Soft Systems Methodology (SSM).

[5] Soft Systems Methodology
https://www.imsl.edu/~sauterv/analysis/F2015/Soft%20Systems%20Methodology.html/htm

[6] No title
https://www.wrike.com/project-management-guide/agile-methodology-basics/