Method selection and planning

Group 17

Team Loading

Joel Warren
Charlie Wilson
Jake Keast
Hari Bhandari
Ishrit Thakur
Joshua Hills

Method selection and planning 4. a)

Software engineering methods

Software engineering has been done primarily using agile methods, intertwined with waterfall methodologies as well. This incorporated rapid development and iteration on the project through frequent collaboration and communication between team members working on the implementation, while using frequent pushes of code to the Github repository.

The waterfall approach was also used to plan our project in a structured way, by first acquiring our stakeholder requirements and then incorporating them into development. The project has also been well documented throughout, following a linear direction.

After researching the two methodologies, agile methods appeared to be better suited given the relatively small scale of the project. Furthermore, agile development suited our co-located team using informal communication. The incorporation of waterfall methodology was also used to help give the project a clear structure. The combination of both these methodologies was used to maximise productivity and ensure the deadline set by our stakeholders was met, while also keeping a sense of linear direction during development.

Development/collaboration tools used and their fitness

- GitHub (development / collaboration)
    - Suitable for agile development through facilitating rapid development, collaboration and version control. Other features like pull requests, issue tracking and file storage also made GitHub suitable for agile development.
- LibGDX (development)
    - Simple to use and automated back-end development supports agile development.
- Google shared drive (collaboration)
    - Instant file access/sharing and real time collaboration made google shared drive suitable for agile development, and for the project as a whole. Storing files like user requirements also aided in waterfall development.
- Discord (collaboration)
    - Instant messaging, voice channels and file sharing meant discord was well suited to agile development.

Alternatives considered

Alternatives to Github considered:

- Gitlab
    - Supports agile development with features like issue tracking and collaboration, however, all team members had prior experience with Github, making GitLab less favourable.
- Jenkins
    - While highly configurable and compatible with Git repositories, the project's smaller size better suited Github.
- Jira

    - Well suited to agile development and highly integrable with other software like slack. However, since slack was not used, and Jira lacks communication and collaboration tools, Github was better suited, which could be integrated with discord using git-logs.

Alternatives to LibGDX considered:

- jMonkeyEngine
    - Suited for java games. However, libGDX is more suited for 2D games.

Alternatives to Discord considered:

- Slack
    - Greater integration options.
- Trello
    - Greater project management features.

Ultimately, Discord was chosen due to all team members having previous experience with it.

Alternatives to google shared drive were not considered due to the team's pre-existing knowledge with it.

Method selection and planning 4. b)

Approach to team organisation

In general, the team's approach to organisation has been through self-management, that decisions are made collectively as a whole, with no specific management roles assigned to team members. Work would be split between two groups, consisting of 3 members each. For organisation, internal deadlines through a gantt chart were then set for work to be completed by.

Documentation was mainly done by Joel, Ishrit and Josh, while implementation was carried out by Jake, Charlie and Hari.

Specifically, Discord, an online messaging software, has been the primary approach to team organisation. We created a Discord server, which allowed us to organise work by creating different text channels, such as having a text channel for each deliverable. The Discord server also allowed us to effectively communicate with instant messaging, and plan/host meetings as well.

Justification of approach

This general approach was appropriate for the project because it supported our primarily agile development methodology. Collaboration was enhanced by having members work on the same deliverable. Furthermore, the internal deadlines set by the gantt chart also supported the waterfall approach, ensuring that each phase of the project was completed before beginning the next phase. This approach also suited the team due to the small size, communication and decision making between 6 members was relatively quick.

This work allocation was suitable for the team because Jake, Charlie and Hari were well experienced in coding before the project, and hence were well suited to the implementation and website development. Joel, Ishrit and Josh preferred documentation, hence this work allocation suiting them.

Discord was appropriate for the team because all members had prior experience with the software prior to the project, meaning setup time was very quick. The project is also relatively small, meaning Discord’s features were sufficient in supporting the team's organisation. Discord was also integrable with GitHub, specifically git-logs, which suited the project.

Method selection and planning 4. c)

Plan evolution

The initial and final gantt charts for the project are shown in the page below.

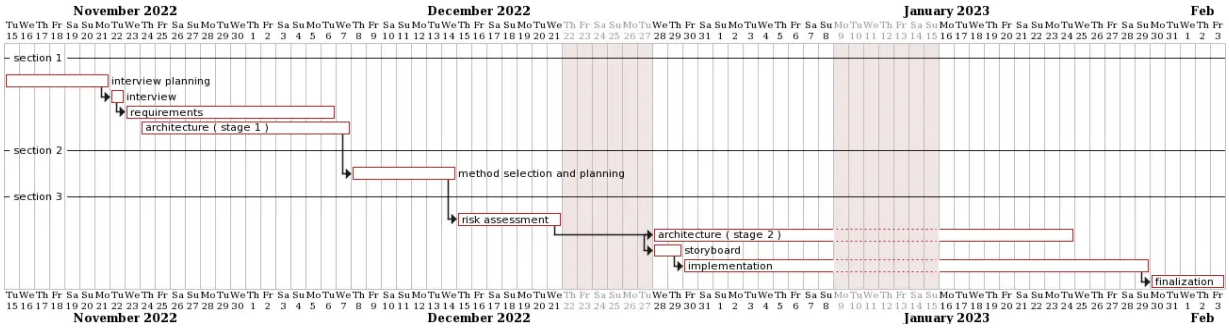
The plan has developed throughout the project. At the start of the project, planning was done week by week, with no holistic overview. Work was only planned for the coming week.

Shortly after, a gantt chart (shown below) was made. This involved planning when each deliverable was to be worked on, and then representing this using plantUML to draw the gantt chart. This plan was mostly kept throughout until the submission deadline. Week by week, the gantt chart was updated to show the current progress on each deliverable, with weekly snapshots of this on the website.

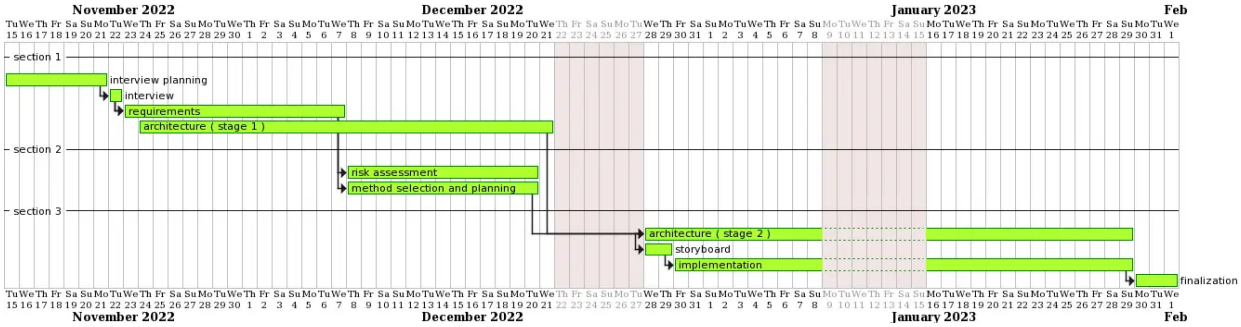
However, some changes were made. Both stages of architecture took longer than expected, and therefore the gantt chart had to be updated to extend the deadline for these tasks. After further consideration, the risk assessment and method selection were moved to be worked on together with the same deadline, since these tasks could be worked on in parallel and to take into account architecture not yet being completed.

Other minor changes such as arrows were implemented to more accurately represent key dependencies.

Initial gantt chart



Final gantt chart



Gantt charts showing a systematic plan for the project.

- Key tasks - denoted by rectangles with description
- Start / finish date - denoted by start and end date of rectangles
- Task priorities - denoted by order of tasks
- Dependencies - denoted by arrows
- Days not being worked on - denoted by greyed out dates