Software Engineering Group 5 Report

Contributed to and reviewed by all group members – 09/04/25

Our report highlights our team's achievements and presents some of the lessons learnt and challenges faced throughout this project. Overall, we have had a successful project and exceeded our expectations for "Property Tycoon".

Key Achievements

Produced a Working Game:

Our proudest achievement is the final version of the game we were able to produce, the climax of this project. We have successfully included the vast majority of the requirements we interpreted from the brief, with many of the core components of the game working smoothly with each other. To our knowledge, the game satisfies Mr Raffles' vision based on the feedback we have had from his representatives, which is very pleasing for our group.

We were able to showcase our product by live-streaming YouTube videos during testing, showing off the prototype ability for AI player agents to play against each other for long periods. Multiple videos demonstrate this, and we have also played the game as a team with it working smoothly.

For us, successfully creating a game that closely matches the requirements means we've avoided repeating the software crisis, which is the reason software engineering was invented in the first place. Therefore, we can proudly say that we've learned the most important thing in software engineering: delivering what's needed on time.

Effective Planning:

One of the concerns towards the start of the project was whether our planning would be sufficient and doable. As this was a large project and multiple group members had never experienced production on this scale, we weren't sure how it would go. However, as we know, developing software from scratch is always considered a highly demanding task (Haigh, 2010). Therefore, the emergence of software engineering is to ensure that software can be built on time. So, we all agreed to work as early as possible and make a good plan. We collectively devised a plan and updated it when necessary, allowing us to finish on time and produce a strong project. We did get some time allocations wrong, however, we knew this could happen and allocated slack time to handle any of these delays. Our efficient and effective planning was a highlight of the development.

Lack of Team Issues:

We are proud that as a team we had very few issues, and any issues that did materialise were dealt with maturely and fully. Our team formed good relationships throughout the project, and the issues we did have were all minor, and had more to do with the timings of meetings and deadlines. We voiced any concerns and came up with solutions that benefitted the whole team.

We communicated daily via our Discord chat and had weekly group meetings, as well as contact with each other in lectures and breaks etc. This allowed us to all be on the same page and informed about the ins and outs of the current development stage, to-dos and problems.

Overall, we are happy and proud of our team's cohesion and ability throughout the project, successfully eliminating any doubts we may have had beforehand.

Documentation Guide

Our project's documentation is primarily organized within the 4. Design Evidence folder. To create a comprehensive design document for our game, we opted for one of the widely adopted documentation generators, Sphinx, using a theme provided by Read the Docs.

The documentation provides concise descriptions of each class used in the game's code, along with detailed explanations of their respective functions. We began by drafting the intended content in .rst files, which also include hyperlinks to the corresponding source code.

In addition, based on resources provided during seminars and available through campus materials, we developed the necessary diagrams for each class. These diagrams were constructed as simplified textual representations of the code structure and formatted according to the required diagram types.

To generate the diagrams, we utilized plantuml.jar. During the build process, PlantUML converts the textual diagram definitions into PNG image files, which are then automatically embedded into the website. These diagrams form one of the key components of our design documentation.

The original. rst source files used to generate the full documentation can be found in 4. Design Evidence/Sphinx Source Code/

What Went Well

Rapid Iteration:

Throughout the development process, we rigorously adopted a rapid iteration methodology. This agile approach allowed us to continuously enhance the game through frequent cycles of building, testing, evaluating user feedback, and refining features. By embracing short development sprints and regular assessments, we ensured a development lifecycle that was both stable enough to guarantee consistent progress and flexible enough to adapt quickly to emerging challenges, new ideas, and testing outcomes.

Software Versioning and Source Control

To complement our rapid iteration approach, robust software versioning was implemented from the very first prototype, regardless of its initial simplicity. We established a clear numbering system using the format build DD.MM.YYYY.

This systematic versioning enabled the meticulous tracking of every update, facilitated clear communication within the team, and allowed for the maintenance of comprehensive documentation detailing progress, changes, and milestones via detailed update logs for each build. Clear versioning proved essential for effective reporting, enabling us to reference the specific build used during testing phases or analysis precisely. This ensured that issues identified in one build could be tracked and verified as resolved in subsequent versions.

We utilized GitHub as our central repository for code storage and collaborative development. While every commit pushed to GitHub receives a unique identifier and serves as a crucial, frequent backup, these individual commits do not necessarily represent stable releases. Indeed, a new push might inadvertently introduce new bugs. Therefore, our formal build DD.MM.YYYY versions specifically denote releases that have reached a milestone of stability deemed suitable for formal testing, feedback gathering, and documenting in reports.

Software release life cycle

Our development process encompassed all key stages of a complete software production cycle, from initial development through Alpha and Beta testing, culminating in packaging for deployment as version 1.0.

Beta testing commenced during Sprint 4 and proved to be a particularly crucial stage. During this phase, we actively solicited feedback from group mates, conducted rigorous level testing, and meticulously assessed system stability. This iterative process allowed us to refine various gameplay aspects and identify areas for improvement. We delayed writing documentation, such as design documents, until after the initial Alpha testing phase.

The insights gained from these early tests resulted in significant changes, including improvements to the user interface and the addition of new features that testers found desirable. For example, we implemented a feature to display player-owned properties with star indicators on the board during each turn, supplementing the existing player card information.

A particularly significant outcome of Alpha testing was the realisation that we needed to overhaul the property development system (building houses and hotels) due to inherent flaws. This major revision highlighted the effectiveness of our testing methodologies.

Following Alpha testing, we moved into Beta testing, and our final targeted release, v1.0, was planned for the last week of the development cycle. Further details and specifics of the testing phase are comprehensively documented in the dedicated testing reports. However, we acknowledged that some deep-seated bugs might remain unresolved within the project timeframe. These issues often originate from foundational design choices made early in the development process.

Game Feel

We prioritised enhancing the "game feel", and the sensory and emotional experience of playing while maintaining a lightweight implementation. This involved incorporating subtle details that significantly improved the overall user experience.

For example, we implemented fade-in/fade-out visual animations when launching the game and included helpful prompts at the start of gameplay to guide new players.

Several hundred lines of code were dedicated to implementing a music system designed to enhance player immersion. The music subtly adjusts to the game state, creating a more engaging and atmospheric experience. We also integrated a feature suggested by Owen that allows players to pan and zoom the game board in all directions (left, right, up, down, zoom in, and zoom out). This feature addresses usability concerns, particularly on smaller screens, by enabling players to closely examine small text elements and details.

These enhancements required relatively little development time but significantly boosted the overall user experience and engagement.

Within the timeframe of our project, we aimed to personalize the game as much as possible. In the settings page, we introduced a screen-scaling system and a font customization option. Several distinct fonts were made available for users to choose from, including pixel-style fonts, clean line-based styles, and even the British Rail typeface, which perfectly complements the game's theme. The inclusion of railway stations and various place names makes this font particularly appropriate, as it was originally designed for clarity and readability in transportation signage.

We also offered a font inspired by the "University of Sussex" typeface, providing players with a diverse and thematically relevant aesthetic experience. We believe these personalized touches significantly elevate the overall enjoyment and accessibility of the game, catering to a wider range of player preferences.

What Could Have Been Better

Game Development System Planning

Early in development, our understanding of all the game's nuances was incomplete. This resulted in inadequate initial planning regarding core gameplay mechanics, particularly the Development system that allows players to build houses and hotels on their properties.

This oversight proved detrimental. After we believed the Development feature was complete, numerous issues surfaced. The timing of the Development feature's implementation was flawed, negatively impacting the overall development schedule.

We initially estimated a week for debugging; however, we ultimately spent over two weeks reworking this feature. By that point, it was too late to redesign the system fundamentally, and the outcome didn't meet our initial expectations.

Reworking and relocating the activation conditions for the Development feature significantly impacted the entire game flow. The hurried implementation introduced several minor bugs, further hindering the debugging progress. Regrettably, even in its current state, the feature still exhibits unresolved issues.

AI Player UI Interaction

We initially underestimated the complexity of designing effective AI opponents, particularly in creating a seamless and intuitive interface that communicated the AI's actions to the human player.

We initially believed that focusing on the AI's decision-making process, using basic random number generation to simulate behaviour, would be sufficient. We consciously avoided complex algorithms at this early stage, assuming the AI would be relatively straightforward to implement. However, the most challenging aspect proved to be the AI's interaction with the user interface. During the AI's turn, the UI needed to indicate that it was the AI's turn and provide feedback on its actions. Certain buttons (such as the Development, Dice Roll, or Auction buttons) also needed to be disabled or hidden during the AI's turn to prevent human player interference.

Implementing these UI interactions proved to be far more complex than initially anticipated. After integrating the AI decision-making code into the game, we encountered a multitude of issues related to UI management.

This led to several weeks of UI redesign and rework, resulting in a very unstable initial AI build plagued with bugs. The game required numerous UI elements to be dynamically enabled and disabled at specific times while preserving the underlying functionality.

Our failure to adequately anticipate these challenges during the planning phase significantly impacted the development timeline. This oversight also influenced our decision to improve the AI's decision-making process. Rather than implementing more sophisticated algorithms for a challenging "hard" mode, we opted for a simpler approach that dynamically adjusted the AI's decisions based on human selection during the game to make it more engaging. Effectively, the AI still relies on randomized decisions rather than advanced algorithmic strategies.

Future Improvements

Bug Reduction and Rule Compliance

Despite rigorous Alpha and Beta testing phases and numerous adjustments, we recognize the need for further refinement to improve game stability and rule adherence. Investing additional time in bug fixing, improving code quality, and ensuring complete compliance with game rules is paramount. Reducing bugs and addressing edge-case inconsistencies will create a more polished and reliable game, ultimately enhancing player satisfaction. We underestimated the debugging time required. Ideally, a period of two to three times longer than the development time should be allocated for thorough testing and bug fixing. Given more time, this would be our absolute top priority.

Expanded Game User Research

As a game, playability and overall richness are crucial for success. While the physical version of the game has proven popular, its digital adaptation requires focused user research to ensure a positive reception. Further investigation into user preferences and gameplay behaviour is needed to align our design more closely with player expectations. This is especially critical concerning UI/UX elements, such as button placement and the overall game feel. Time constraints limited our ability to conduct comprehensive Game User Research during the development cycle, though extensive planning for this phase was completed. (See planning document / User Research Plan.pdf for further details.) In future iterations, we intend to prioritize this research to inform design decisions and optimize the player experience.

Enhanced Game Feel and Personalization

While our current game offers a reasonable level of personalization, there is significant potential for improvement. For example, the message log located in the lower-left corner of the game interface can be too long and verbose, potentially overwhelming some players. Moreover, not all players may find the message log useful, as they may not be interested in the actions of other players. Therefore, introducing an option within the settings menu to disable the message log would allow players to customize their experience according to their preferences. Further exploration of customization options and subtle enhancements to the game feel would contribute to a more engaging and enjoyable experience for all players.

AI Agents enemy

The game requirements mention the need to add "AI agents" as enemy players. While this might simply mean implementing basic, algorithm-driven AI opponents, we're intrigued by the possibility of leveraging truly advanced AI agents, reflecting cutting-edge developments in the field. This immediately sparked creative ideas, drawing parallels to the groundbreaking achievement of AlphaGo, which, in March 2016, defeated Lee Sedol in a five-game match, demonstrating the immense potential of AI. Imagine an AI opponent capable of strategic thinking, adaptation, and even a degree of unpredictability.

Therefore, it would be incredibly interesting if we could also integrate Large Language Model (LLM) powered AI enemies to battle players within the game. This goes beyond simple scripting and opens the door to dynamic and responsive AI behaviour. We could write the game rules into a prompt and then use structured output functionality, enabling the AI to not only output text for engaging in natural language conversations with human players, but also to generate appropriate and contextually relevant movement commands and actions within the game world. This would allow for

emergent gameplay and create unique encounters that are difficult to replicate with traditional AI techniques.

Such an approach would allow the game to perfectly meet the enemy capability requirements for players of all skill levels, from beginners to seasoned veterans. The AI could dynamically adjust its difficulty, adapting to the player's skill and experience to provide a challenging yet fair experience. New players would face more forgiving opponents, while experienced players would be tested by cunning and resourceful adversaries.

Currently, our AI system primarily relies on probabilities to simulate randomness in enemy behaviour. While this approach has its merits, it often results in predictable patterns and limited strategic depth. Once we genuinely utilize the full capabilities of AI agents, the game's replayability will be significantly enhanced. For example, players could potentially engage in dialogue with enemy AI characters, influencing their decisions and altering the course of events through conversation. A dedicated chat function could be implemented to facilitate these interactions. This would add a layer of social complexity and strategic depth to the game.

This would also greatly enhance the human player's overall experience, as they would no longer feel like they were simply facing mindless drones. The ability to interact with and potentially outsmart the AI would create a more engaging and rewarding experience. It would also allow human players to experience something akin to the strategic prowess demonstrated by AlphaGo right from their homes, providing a unique and compelling gameplay experience.

Although we know there are now many readily available resources and frameworks for learning about and implementing AI agents (e.g., https://github.com/microsoft/ai-agents-for-beginners , https://huggingface.co/learn/agents-course/unit0/introduction), and successfully integrating them into the game would undoubtedly better fulfil the game's requirements and elevate the overall quality of the product. Moreover, the Python language we have chosen as our primary development language is exceptionally well-suited for integrating AI functionalities, thanks to its rich ecosystem of libraries and tools.

However, given our current project timeline and resource constraints, we simply lack the time to simultaneously learn about the intricacies of AI agents and reliably integrate this advanced functionality into the game within the existing schedule. The associated risks of incorporating a complex and potentially unstable AI system are simply too high at this stage. If we had the time and resources to fully explore and implement this technology, it would certainly be a fascinating and transformative addition to the game.

Sprint Evaluation

We completed four sprints in total, three lasting two weeks and one lasting three weeks. Each sprint was successful and a valuable learning experience. We found that the sprints gradually became more and more complex as the project developed and the tasks became less broad. This was not a problem, though, as all tasks in the sprints, except one, were completed on time and to a high standard. Each sprint had a thorough evaluation, allowing us to review our work and highlight changes to our methods in the future.

We had planning documents, a requirement analysis, a risk sheet and other documents to assist us with the long-term project targets, but the sprints were useful for structuring and accomplishing our short-term goals. They allowed us to work towards achievable goals and build upon previous features rather than trying to do it all in one go.

One mistake we did make was not updating the sprint as we went along during the first sprint. We started the sprint during the second week of the term but didn't begin producing the documentation for it until the third week, which meant we had to catch up on a week's worth of progress. Since that point, we made a conscious effort to update the sprint as we went along, even if it was just adding a few notes so we had good points to add that may otherwise be forgotten. It also enabled us to be more productive during the sprint as we could tick off tasks when they were complete, meaning we could move forward to the next task.

As a group, it was very pleasing to see the improvements in our prototype as the sprints progressed, allowing us to visually see just how far we had come. The difference in the prototype between sprints one and four is immense, highlighting just how much effort we had put into the development.

Our reviews of each sprint outlined achievements, future improvements and outstanding issues we had noticed. This allowed us to improve sprint on sprint in our approach to the tasks, which was very beneficial as the tasks became more complex.

During seminars and on the canvas discussion page, we were provided with feedback from Mr Raffles' representatives, and this was reflected in our sprints. The feedback was very useful, and we tried to implement it as best we could.

Summary

In conclusion, our coursework project has been completed successfully, and we are pleased with the entirety of the results. The sprints flowed seamlessly; our planning and documentation were effective, and the product produced exceeded all prior expectations.

We may have overrun our ideal deadline of having the project completed by the end of week 9, but this was within reason, and we knew this was a strong possibility. The two-week slack time at the end of the project was used to try and fix any outstanding bugs and issues identified, complete final documentation and double-check that all deliverables were ready, ensuring we had the best project possible to submit.

So overall, we, as a team, are incredibly happy with the work we have produced and have all gained valuable skills and experience in larger-scale group and extended group projects. We know what works well and what to do in the future to improve.

Useful Links

PERT Document Final.pdf and Risk Analysis Report .pdf

YouTube Channel: https://www.youtube.com/channel/UCfA_oAs-qoOGMWJ_UueYHng

GitHub repositories: https://github.com/Minosaji/Software-Engineering-Project

Design Documentation: https://minosaji.github.io/Software-Engineering-Project/

T. Haigh, *"Crisis, What Crisis?" Reconsidering the Software Crisis of the 1960s and the Origins of Software Engineering*, draft version presented at the 2nd Inventing Europe/Tensions of Europe Conference, Sofia, Jun. 17–20, 2010. [Online]. Available: https://www.tomandmaria.com/Tom/Writing/SoftwareCrisis_SofiaDRAFT.pdf

Peer Review

Group 5 Team Members and Assigned Points:

Eric Shi: 20

Stuart Baker: 20

Lin Moe Hein: 20

Duncan Law: 20

Owen Chen: 20

The total available points are 100 (5*20 per member).

Used Resources

Below is a list of resources that helped us create the game. This is only a portion of the references we used; we also appreciate other resources that are not listed here.

Fonts Source:

- https://www.strathpefferjunction.com/download/british-rail-fonts-rail-alphabet/
- Google Fonts
- https://www.1001fonts.com/ticketing-font.html
- https://fonts.google.com/specimen/Libre+Baskerville
- https://www.sussex.ac.uk/brand/designers/fonts
- https://www.strathpefferjunction.com/british-railway-typefaces-fonts/

Image assets:

Throughout the development of this game, we employed generative AI tools in a limited way for the creation of certain visual assets. This approach aligns with the permitted use of AI outlined in the module guidelines, which allows for the generation of art/multimedia/non-coding assets.

We utilized generative AI to create the game logo and the background images for the game. The rationale for using AI in these specific instances was to create more suitable, unique, and thematically relevant background art, ultimately enhancing the game experience.

The process involved providing the AI tool with specific prompts designed to guide the generation of the desired assets. One example of the prompt used was: "pixel pc 2d game background, make it look very simple and not complex"

The AI tool generated various outputs based on these prompts. These outputs were then downloaded and integrated into the game as background images. In all cases, the AI-generated assets underwent significant selection and minor adjustments to ensure they met the specific artistic style and requirements of the game and were visually appealing. We acknowledge the use of the following generative AI tool(s) in the creation of these assets:

- https://deepmind.google/technologies/imagen-3
- https://labs.google/fx/

All prompts used and resulting image outputs have been meticulously documented and stored. This information is readily available for review upon request.

Sound effects Source:

- https://pixabay.com/sound-effects/dice-142528/
- https://tuna.voicemod.net/sound/719f1389-c249-4c39-a161-2cb267178036
- https://tuna.voicemod.net/sound/00a9071e-8ed6-4f29-af49-c9005466f41d
- https://pixabay.com/sound-effects/place-100513/
- Sound Effect by Modestas123123 from Pixabay
- https://pixabay.com/sound-effects/cash-register-kaching-sound-effect-125042/
- https://pixabay.com/sound-effects/coin-257878/

- https://pixabay.com/sound-effects/game-over-classic-206486/
- https://pixabay.com/sound-effects/cashier-quotka-chingquot-sound-effect-129698/
- https://pixabay.com/sound-effects/menuselect4-36147/
- https://pixabay.com/sound-effects/winsquare-6993/
- https://pixabay.com/music/video-games-game-176807/
- https://pixabay.com/sound-effects/logo-pianoremate-dm-257455
- https://pixabay.com/sound-effects/time-passing-sound-effect-fast-clock-108403/
- https://pixabay.com/sound-effects/logo-piano-125764/
- https://www.zedge.net/ringtones/484ea82f-4d9d-3660-b7d4-3017ecc0d46f
- https://pixabay.com/sound-effects/family-store-logo-13175/
- https://marketplace.visualstudio.com/items/?itemName=MS-SarifVSCode.sarif-viewer
- https://pixabay.com/sound-effects/success-bell-6776/
- https://pixabay.com/sound-effects/ding-small-bell-sfx-233008/
- https://pixabay.com/sound-effects/cry-131025/
- https://www.youtube.com/watch?v=6yrdS4tIP9U
- https://www.youtube.com/watch?v=GTXUWhmjCog
- https://www.youtube.com/watch?v=kpwNjdEPz7E
- https://www.youtube.com/watch?v=A8wK-vhuWog
- https://www.youtube.com/watch?v=wO9q4H49cGA
- https://www.youtube.com/watch?v=wUwYZ3

Code Scan Source:

- https://github.com/marketplace/actions/bandit-scan
- https://github.com/marketplace/actions/bandit-action

Others:

- Pygame Subreddit
- Pygame Official News
- Pygame Tutorial
- Pygame GUI
- Setuptools (PyPI)
- https://www.naukri.com/code360/library/window-in-pygame
- OpenPyXL Documentation
- Pandas Official Website
- Pandas (Wikipedia)
- Nuitka Compiling with MinGW64 on Linux(Reddit)
- Improving Performance in Pygame
- Pygame Image Documentation
- Why Do My Games Lag? (Reddit)
- First Weekend Writing a Python Game
- They_Comin (GitHub)
- r3frame (GitHub)
- Nuitka (GitHub)
- Nuitka Official Website

- Chinese Nuitka Compilation Guide (YouTube)
- Clang (Wikipedia)
- Pygame Web
- auto-py-to-exe
- PyInstaller
- https://www.sphinx-doc.org/
- GitHub Actions Documentation
- Nuitka GitHub Action
- Pygame Web Documentation
- Building and Testing Python with GitHub Actions
- How to Test an Elevator (Chinese)
- Pytest Documentation
- https://www.youtube.com/watch?v=n8EtFlaSRVM&t=371s
- How to Format Python Code on Save in Visual Studio Code
- https://marketplace.visualstudio.com/items/?itemName=ms-python.black-formatter
- Game UX Research Discussion(Reddit)
- Ruanyf Weekly (Chinese)
- HelloGitHub (Chinese)
- How to Play Monopoly Go (YouTube)
- Monopoly Go Online (Lagged) - Play Monopoly Go online for free.
- Monopoly in JavaScript (IntrepidCoder) - A fully functional Monopoly game in JavaScript, open-source on GitHub.
- Monopoly Game in Python (AniketSanghi) - A Monopoly game made with Python using Pygame and Tkinter.
- Pygame Monopoly-AW (YouTube) - A video example of Monopoly using Pygame.