Assessment 2: Brownfield Development

Risk Assessment

Cohort 3 Group 9

Oliver Barden

Connor Burns

Sam Goff

Milap Keshwala

Lewis Mitchell

Jeevan Singh Kang

Harry Thomas

Titas Vaidila

Introduction

As developing a system can be a very versatile process, prone to changes and adversities, being able to predict and mitigate risks is very important. If an issue arises that could have a big enough impact, it could completely halt development of the project. We are working within a certain timeframe, with a hard set deadline, and so this happening could be detrimental, and could result in us not meeting our targets in time, which could be a great hindrance on the customer, as they may require the software to be able to function properly by a certain point.

To cope with this, we followed a simple risk management process we found to be appropriate for a small scale project. First we identified plausible risks by brainstorming as a group using the knowledge of our requirements, tools, schedule and assessment scope. We then analysed each risk by estimating the impact and likelihood of each risk using a 3 level scale. For risks which landed higher on the scale we planned for how we would mitigate these risks and actionable steps we could take if they were to happen. Throughout the development of our project we continued to monitor our risks by reviewing and updating them as seen fit with a designated owner attached who was held responsible for monitoring the risk and making changes.

To do this, we needed a way to effectively organise and plan for risks. As such, we have decided to do our risk register in the form of a table. With this, we can record each risk, its likelihood (how likely the risk is to occur), impact (how severely it would impact us meeting our requirements), consequences, mitigations, contingency plan and ownership in a very clear, concise and easy to understand manner. Using a table helps us to review risks in a meeting, check mitigations are complete and update based on evidence we've discovered throughout the week such as issues in implementation or testing. As a result, any issue that arises will be less likely to negatively impact the team and the development of the project.

Key:

L = Low
M = Medium
H = High

Risk (ID)		Likelihood	Impact	Consequences	Mitigation	Contingency Plan	Owner
Team member stops contributing (R000)		L	M	Aspects of the project will not be completed, as everyone has already been assigned specific tasks to complete, causing progress to slip and an additional workload for the remaining team.	Conduct weekly check-ins to ensure everyone is coping. Also, assign a shadow member to ensure the work is being done and to help out where possible.	Redistribute the abandoned workload fairly across the rest of the team. If needed, defer non-essential tasks to meet the deadline.	Lewis Mitchell
Remote repository goes down (R001)		L	M	Temporarily, no one will be able to access files related to the project i.e. code or planning documents until the remote returns which slows coordination.	Keep backups of a working clone on everyone's local machine, push changes frequently and make backup copies of documents.	Find the most up to date local copy on someone's machine, work on that in the meantime, then consolidate changes when the repository is back up.	Sam Goff
Development software technology/tools become unusable due to update or misconfiguration (R002)		L	M	Development on the implementation would slow or halt, and those coding would not be able to meet deadlines.	Regulate tool usage, avoid unplanned updates, document setup environment and ensure code is always executable.	Revert back to known working versions, switch to a familiar alternative tool and rebuild the code on a fresh setup.	Implementation Team
Hardware in use stops working (R003)		M	L	Affected team members may be unable to contribute to the project, resulting in task delays and reassignments.	Frequent pushes to the repository, regularly share documentation and keep at least one accessible back up environment (campus machine) to resume work on.	Reassign tasks temporarily and make use of publicly available hardware in the meantime (i.e., library computers, university machines, etc) while team members regain a working setup.	Connor Burns

Customer changes their mind on a certain aspect of the product or clarification on a requirement is received late (R004)		L	H	We would have to stop development on anything that doesn't meet the new requirements and rework them, potential loss on additional features outside the core requirements.	Confirm requirements early and clarify any queries we have. Agree to freeze any development on any unnecessary features until the prototype is back on track.	IImmediately stop production on those parts that do not fit the new criteria, and start planning on how to work within the new constraints by assessing any changes and the impact they may have. Reprioritise work to target core requirements, update any documentation, and defer non-essential work.	Titas Vaidila
Feature implemented but isn't up to clients standard (R005)		L	M	Time spent implementing would go to waste if it doesn't work or improve gameplay, rework needed and optional features may be dropped to protect requirements.	Regularly meet with the customer and create simple acceptance criteria with the customer to determine what is good. Collect frequent feedback and log changes.	Take customer feedback and clarify exactly what they want and how they want it, immediately start planning how to tweak if possible or completely remove it if it still doesn't meet the acceptance criteria.	Jeevan Singh Kang
Development library in use for the development becomes incompatible and blocks development till resolved (R006)		L	M	The code fails to build and loses time trying to fix it, resulting in a potential delay to the deadline.	Pin dependency versions in place and avoid updating any, especially late into the project. Document the known working environment setup.	Revert tools and technologies to the last known working version, prioritising getting a working build first, and replacing the dependency only if required, followed by refactorisation of code.	Implementation Team
Unlicensed or unattributed third-party assets or other products are used (R007)		L	H	We may need to remove or replace assets already implemented in causing rework and potential of failing to meet the submission if not corrected.	Maintain an asset list with reference to where it was sourced from, attribute where need be and aim to use open licensed resources.	Identify anything falsely used and remove it, replace it with an appropriately licensed alternative and retest it. Update attribution documentation.	Milap Kewshala

The quality of code produced is poor due to unfamiliarity with the inherited codebase (R008)		M	H	Increased bugs and slowed development due to rework, less time to test and polish towards the end of the project	Regularly review the code and make small changes in regular intervals using consistent formatting. technologies.	Priorities research into unfamiliar libraries or if found to be too difficult/time consuming, refactor code to use familiar libraries and methods. Freeze new feature implementation if bugs arise and focus on stabilizing core gameplay.	Sam Goff
Deployment goes poorly (missing assets, etc) causing final version not to run as expected (R009)		M	H	The software may run incorrectly and add time consumption close to deadline fixing the submission and less time to test and refine product.	Create a release checklist and produce at least one practice build to be run on a clean machine and document the steps taken.	Fix submission using the release checklist.	Harry Thomas
CI build fails or has errors and bugs (R010)		M	H	Users unable to access/play the game without issues would likely result in loss of the player base and negative reviews.	The build should be blocked by the CI pipeline, preventing it from progressing further, meaning this build will never reach deployment.	If these pipelines fail and the build gets deployed, we should roll back the deployment and then work on fixing the issue.	Oliver Barden
Testing results in a false negative (R011)		M	M	Lead to bugs which would be extremely hard to find, and could be exploited by users.	Ensure we follow a mix of unit & manual tests, prioritise tests for core gameplay and regularly review code.	We and users can report bugs they find that would help the team discover and remove them. Freeze any implementation of new features and focus on stabilising the game.	Implementation and Testing team
Inherited project has flaws in the deliverables or hidden bugs in the codebase (R012)		M	M	Would lead to pre-existing bugs that we had not programmed and therefore extra time debugging and increased time	Make sure testing is not limited only to new features that we add and that we thoroughly read through the	Similar to R011, users can contact the team to report bugs and flaws.	Implementation and Testing team

				to implement new requirements. More chance of an unstable final project.	deliverables we were given to be in order to be familiar with the project.		
The game is too easy/hard to play/complete (R013)		M	L	Players may find the game too frustrating/boring leading to reduced enjoyment and negative user evaluation findings.	Ensure thorough testing and user evaluation to notice this as soon as possible. Keep difficulty components configurable (e.g. movement speed so improvements don't take large code rewrites.	Adjust difficulty by nerfing/buffing events and re-evaluate.	Lewis Mitchell