University of York Department of Computer Science
Engineering 1

Methods Selection & Planning

Cohort 3 Team 5 - alltheeb5t

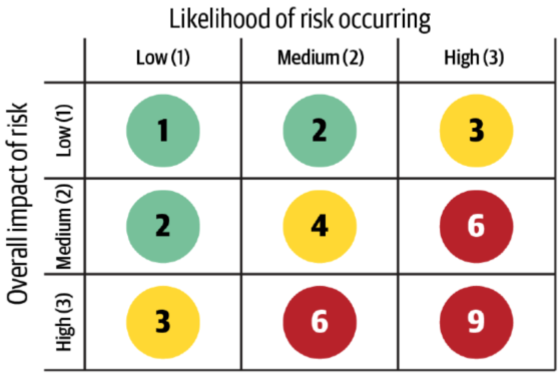
Aaron Heald
Alex Gu
Arun Hill
Jade Stokes
Maksim Soshchin
Meg Tierney
Will Hall

This document outlines a comprehensive risk assessment for our group. We have evaluated the risks based on the severity and likelihood to make them in a low (Green), medium (Yellow) and high (Red) categories [1].

Risk Types

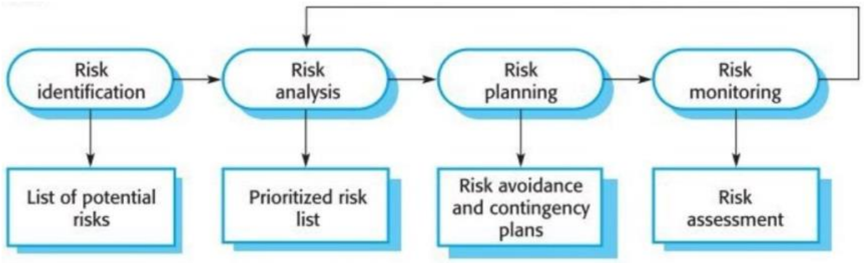
- Schedule Risk – Time related risk or project delivery related planning risks. It can be a member of the team who did not finish a task on time or underestimated the time needed to complete a task.
- Team – Situations like a teammate falling ill, withdrawing from studies, missing group meetings, or not completing their assigned work can strain team relationships and hinder overall performance.
- Client requirements – This can be the final product that we produced that does not match the Client's requirement or missed requirements from the product brief.
- Users – Players may lose interest if the game doesn't meet their expectations or if they encounter issues running it on their devices.
- Code – This risk arises when client requirements cannot be met due to coding issues. Additionally, coding problems can cause the game to crash or fail to report errors accurately.
- Tools - A tools-related risk could stem from the limitations of the software used in the project, such as generating inefficient code or lacking compatibility. This may be indicated by outdated software or recurring errors during program compilation.

Risk Matrix



		Likelihood of risk occurring		
		Low (1)	Medium (2)	High (3)
Overall impact of risk	Low (1)	1	2	3
	Medium (2)	2	4	6
	High (3)	3	6	9

This risk matrix from the book *Fundamentals of Software Architecture: An Engineering Approach* [2], has proven very helpful. It provides a single number to represent the significance of each event, making it easy to grasp the level of importance at a glance. We've decided to incorporate this approach into our own risk table.



[3]

This diagram outlines the software risk management iterative process, involving the identification, analysis, planning, and monitoring of risks to proactively address potential issues throughout the project.

ID	Description	Risk Type	Likelihood	Severity	Risk Matrix	Mitigation	Ownership
R1	One of the team members is ill and won’t be to do work	Team	M	L	2	Other team members will need to take over the work that need to be done	All
R2	One of the team members withdraw from studies	Team	M	M	4	Need a meeting to separate the work that the previous teammate was doing	All
R3	One of the team members unable to complete the work on time	Team	M	L	2	Workload can be switched between team members and deadlines extended in conjunction with new work	All
R4	Project became more complex than planned	Schedule Risk	L	M	2	Remove unnecessary features when designing. Frequent updates between development team	Alex
R5	Project does not meet client expectations	Client requirements	M	H	6	Frequent updates and meetings on progress with client	Jade,Aaron
R6	Poor quality code leading to bugs and logical issues	Code	L	H	3	Frequent testing of new code and how they work together. Run Larger test.[4]	Arun, Will
R7	Users are dissatisfied with the overall scope of the project	User	L	H	3	Publish beta test models before the final version, also Conduct surveys and implement feedback given by users	Maksim
R8	Requirements changed once implementation begins	Client requirements	L	L	1	Updating requirements document if new requirements are found or if old ones are no longer needed and redesign the game if needed	Jade
R9	Team needs to learn new game engine software and libraries	Tools	M	M	4	Meeting to help newcomers understand the software	Arun, Will

R10	Team members might be unfamiliar with certain tools	Tools	M	L	2	Practise them before actually using them in the project	Meg
R11	The time required to develop the software is under estimated	Schedule Risk	H	H	9	Prioritise Key Deliverables. Increase communication between team members, and see if help is needed	Alex
R12	Merge Conflicts and Code Overwrite	Code	M	M	4	establish a clear branching strategy, encourage frequent small commits, assign code ownership, and implement peer code reviews.	Will, Arun
R13	Error caused by tools not integrating together well	Tools	L	M	3	Check for updates on all the tools used, check the documentations of the tools	Aaron

References

[1] K. Rafalski, “Top Software Development Risks and How to Mitigate Them,” *Netguru*, Jun. 24, 2024. https://www.netguru.com/blog/software-development-risks (accessed Oct. 11, 2024).

[2]

Figure 20-1. Matrix for determining architecture risk

Richards, Mark, and Neal Ford. *Fundamentals of Software Architecture : An Engineering Approach*, O'Reilly Media, Incorporated, 2020. *ProQuest Ebook Central*, http://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=6029037.

Created from york-ebooks on 2024-11-06 21:34:14.

[3]

Figure 22.2 The risk management process

Sommerville, Ian. *Software Engineering, Global Edition*, Pearson Education, Limited, 2015. *ProQuest Ebook Central*, http://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=5185655.

[4]

“Larger tests might also be necessary where smaller tests fail. The subsections that follow present some particular areas where unit tests do not provide good risk mitigation coverage.”

Page 283, Winters, Titus, et al. *Software Engineering at Google : Lessons Learned from Programming over Time*, O'Reilly Media, Incorporated, 2020. *ProQuest Ebook Central*, http://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=6125411.