Module Code
COM00008I

BSc, BEng and MEng Degree Examinations 2019–20

DEPARTMENT OF COMPUTER SCIENCE

Software Engineering Project (SEPR)

Open Group Assessment

Module	Software Engineering Project (SEPR)
Year	2019/20
Assessment	1
Team	The Dicy Cat
Members	Michele Imbriani Daniel Yates Luke Taylor Isaac Albiston Martha Cartwright Riju De Sean Corrigan
Deliverable	Risk Assessment and Mitigation

For this part of the project, the team has relied heavily on the book “Software Engineering, Global Edition”[1] for the risk elicitation process. The textbook provides a detailed and in-depth description of how to effectively identify, assess and mitigate the potential risks involved over the course of a software development/engineering project. We did however also make extensive use of the lecture slides alongside the book in order to get as full a view on risk assessment as possible.

The tabular format used for the risk assessment is the one shown in the lecture slides. Given our small team, relatively long period of development and fixed deadline/budget, we concluded that this was the most concise and detailed way in which to represent the potential risks.
A fundamental part of the risk assessment was the classification of likelihood and severity for each risk. We decided to use the 3-value scale for both: Low (L), Medium (M) and High (H). This allowed us to identify which risks would be more likely to have a large impact on the project but also did not go into too much detail such that the risk assessment would take a long time with very minimal returns. Furthermore, as suggested by Boehm in the textbook “Software engineering” [1], we identified the top 5 risks to which the team will pay more attention than the rest (the literature suggests to identify a “top 10” risks, but it assumes the scale of the project to be bigger than what it actually is in our context: we therefore decided to reduce from 10 to 5). These are coloured in red in the table.

The factors we took into account when assessing the severity of a risk were multiple, but for the most part they revolved around how confident each team member felt in their skills for solving the issue at hand in a hypothetical scenario in which the risk has become a reality i.e. how well would the team be able to cope with the problem if a risk turned out to be true? The likelihood of a risk, on the other hand, had more to do with how much it is related to this SEPR project: it is unreasonable to classify a risk such as “The hardware on which the game is expected to be run does not support its size” as high likelihood, given that the aim of this SEPR project is to be specifically run on low-end computers, while it would also be equally unwise to categorise, for instance, “The size of the software is underestimated “ as low likelihood, considering that most of the team members have little to no experience in video-game development.

In our risk assessment table, we have assigned a single member of the group to be the owner of each risk. The team unanimously agreed on the importance of assigning each risk to one owner only, as it gives us a clear indication of where each group member should focus their attention more. We assigned the risk ownership based on the type of the risk, as this allows each group member to get accustomed to dealing with risks of a specific type. The risks were, as a matter of fact, divided into 6 categories, namely Requirements, Estimation, People, Technology, Organisational, Tools, in order to both facilitate the risk ownership division and to have a clearer idea of which specific area of the project requires more attention and carefulness than others.

Finally, for each risk we have identified a mitigation strategy. To identify these strategies, we had a group meeting where we discussed the possibilities and entered the agreed upon mitigation in the risk table. In this meeting we used examples given in the lecture slides and in the “Software Engineering, Global Edition” book to help us formulate our risk mitigations.

References:

[1] Sommerville, Ian. *Software Engineering, Global Edition*, Pearson Education Limited, 2016. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/york-ebooks/detail.action?docID=5185655.

ID	Type	Description	Likelihood	Severity	Mitigation	Owner
R1	Requirements	Changes in the project requirements	H	L	Agile, SCRUM software development method allows for quick changes to the requirements	Michele
R2	Requirements	Adding unnecessary features	M	M	Before implementing any features check they can be traced to a requirement	Martha
R3	Requirements	Not resolving a requirement	L	H	Regularly meeting with the stakeholders to confirm the software meets their requirements.	Michele
R4	Requirements	Misinterpreting a requirement	L	H	Writing requirements with enough detail to be clear and demonstrating features to the customer	Michele
R5	Estimation	Time required to develop the software is underestimated	H	M	Generously allocate time to each task in order to minimise the risk of running out of time	Martha
R6	Estimation	The rate of the defect repair is undefined	H	M	Rank tasks' priority by difficulty so that most requiring tasks can be assigned to more proficient members	Dan
R7	Estimation	The size of the software is underestimated	H	M	Generously allocate time to each task in order to minimise the risk of running out of time	Dan
R8	People	The task assigned to a developer cannot be completed because of his/her's lack of skills	M	L	Rank tasks' priority by difficulty so that most requiring tasks can be assigned to more proficient members	Sean
R9	Technology	Unable to find information needed on how to implement a wanted function	L	L	Refere to literature or change the underlying mechanisms of the feature	Riju
R10	Organizational	The organisation is restructured so that different managment are overseeing the project	H	L	By keeping up with clear documentation, change in leadership should be smooth through continuation of previously laid out plans.	Luke
R11	Organisational	Organisational financial problems force reductions in the project budget	M	L	While we do not have a set budget for the project, any key financial requirements we have that we can justify can be requested from the University.	Luke
R12	People	Key developers are ill or unavailable at critical times	M	H	Integrate time into our plan so that certian people are able to take on extra tasks if required	Sean
R13	Tools	Software tools or libraries are inefficient, inappropriate or do not work as expected	M	L	Test the libraries we will implement on our target devices to ensure that it runs as expected.	Isaac
R14	Technology	The hardware on which the game is expected to be run	L	L	Avoid implementation of computing-demanding	Riju

		does not support its size			features and periodical testing of hardware-compatibility	
R15	Technology	Relying on a library/libraries that prove to be missing a key feature needed for the project later in development.	L	H	Thoroughly research any libraries/APIs that will be heavily relied on to ensure they conform to details laid out in the software architecture and requirements.	Riju
R16	Technology	Unexpected bugs/issues with the game due to different members working on seperate sections of the program.	L	M	Ensuring team members communicate and follow the architecture. Create efficient unit tests to pre-empt and isolate any errors.	Riju
R17	Estimation	Failure to meet individual assessments deadlines	L	H	Focus on SCRUM principles of working sprint by sprint, focusing more on the current assessment	Dan
R18	Technology	Failure to effectively select an efficient code during code-swapping phase of assessment	M	H	Thorough review of each team's code and work towards an unanimous team decision before picking	Riju
R19	Estimation	Final product is not as efficient as estimated due to undefined reasons	L	H		Dan
R20	Estimation	Too much dependency of features refraining software's development	M	M	Structure software's architecture trying to avoid or minimise possible compromising features dependencies	Dan
R21	People	Misunderstandings between team-members on	M	M	Use of clear and technical gergon, emphasize communication between team-members	Sean
R22	People	Team-member's personalities might clash with each other in critical decision-making moments	L	L	Trying to always make decision prioritising the team's best outcome instead of individual. When needed, refere to code of ethics to solve disputes	Martha
R23	Requirements	Complaints from client that contradict initial requirements	L	L	Clear and precise specification of requirements at beginning of project	Michele
R24	People	Reluctance of team members to adopt new/unfamiliar technologies decided by the team	L	L	Work towards a unanimous decision of what software and tools to use throughout the course of the assessment	Sean
R25	Technology	Software testing structured in a faulty way, producing wrong results and compromising project's succesful development	M	H	Careful development of test units, crosscheck of tests units between team-members	Riju
R26	Technology	Storage tools failure	L	H	Multiple versions of the same files on every team-member's device	Riju

R27	Estimation	Minigame development turns out more time and effort demanding than expected	M	M	Choose a style for the minigame that every team member is comfortable with doing with relative ease: platform-based game (like supermario/flappy bird)	Riju