Module	SEPR
Year	2019/20
Assessment No.	3
Team	Mozzarella Bytes
Members	Kathryn Dale, Emilien Bevierre, Ravinder Dosanjh, Elizabeth Hodges, Daniel Benison, Callum Marsden

Risk Assessment and Mitigation

N.B All changes made for Assessment 3 are in bold

The understanding of risks a project faces is an important part of planning in a software engineering project. A project may face risks of various different kinds and so developing an effective method of managing these risks is essential.

A widely used risk management approach is that of Boehm [1], where he proposes a model that considers the exposure a project has to individual risks and therefore prioritises the avoidance of exposure to these risks above lesser risks. Boehm also considers the management and monitoring of ongoing risks throughout the duration of a project, tracking risks at milestones and summarising and analysing the top-10 risks on a regular basis.

The foundations of this are built upon by Sommerville [2], who outlines the process of risk management, extending Boehm's model and providing an application in relation to modern agile software development environments, such as those used by this project. It is from this basis that the risk management of this project is constructed from - following a process of continually analysing, planning and monitoring risks.

In the initial phase a list of potential risks to the project will be made, based primarily on the different types of identifiable risk; estimation, organisational, people, requirements, technology and tools [2].

Thereafter will be a continued cycle through the rest of the process, moving first to risk analysis, where the probability of the risk occurring will be assessed on a scale of 'low', 'moderate' and 'high' where a 'high' probability indicates near certainty that the risk would arise. Following this, the effect or impact of the risk is identified on a second scale of 'insignificant', 'tolerable' and 'serious', where a 'serious' impact would indicate significant issues to the project, such as being able to deliver a product or feature. These two measures are combined, where each probability and impact level is assigned a value of the range 1 to 3, and these are multiplied for each risk, giving an indication of risk exposure, where with higher exposure values the risk priority is higher.

For each identified risk, avoidance strategies, minimisation strategies and contingency plans are determined to assist or alleviate the potential impact any given risk may cause should it arise, giving focus primarily on avoiding the risks where possible.

Finally, a continued process of monitoring and reassessing the given risks continues throughout the project (in this case at the beginning of each *sprint*), and where the potential for new risks arises, these are processed as with any other risk to ensure effective project management. This is managed most effectively by assigning each risk an *owner* - a member of the development team responsible for assessing and reassessing the risk. It is important that for each risk a

monitoring strategy is outlined and measurable, such to prevent unexpected incidents arising where resolution strategies or plans have been outlined. During the stage of reassessment, the probabilities and impacts should also be recalculated such to accurately reflect the risk likelihood and effect.

ID	Risk item	Probability 1(low)-3(high)	Impact 1(low)-3(high)	Exposure Probability x Impact (of 9)	Mitigation / Resolution Avoidance, minimisation and contingencies.	Monitoring How to measurably monitor the risk
R-01	***Estimation*** The time required to develop the software is underestimated	2	2	4	Minimisation - investigate technical requirements during the planning process and adapt the development timeframe to maintain accuracy. Contingency - plan for additional time at the end of the development phase to allow for overrun if necessary.	Regularly check progress against the initial plan, and where deviation has occurred, adjust the time frame to more suitably reflect the development time required. **Owner: Kathryn Dale**
R-02	***People*** Developers do not have the required skills.	2	3	6	Avoidance - gain an understanding of developer skill-sets during the planning phase, and select tooling based on existing skills. Minimisation - where individuals do not have the required skills, encourage them to learn these. Assign objectives to individuals able to complete them.	Regularly check on team members to ensure they are able to complete objectives, and where necessary encourage or assist them in acquiring skills or assigning different objectives. **Owner: Daniel Benison**
R-03	***People*** Developers are ill or unavailable. Poor productivity of team members during the project.	1	2	2	Contingency - usage of short sprints (a week or less) and regular scrums to ensure that assignments are not insurmountable if a task is not completed. Well- documented architecture and/or plans to allow for other developers to complete these assignments	Regularly check in with team members (such as with scrums) so the team are aware of any unavailability as quickly as possible. **Owner**: **Kathryn Dale**
R-04	***People*** Poor team dynamics.	1	3	3	Avoidance - interactive team management through regular contact and opportunities to identify potential arising issues. Contingency - process of four peer assessments and contact with lecturers to ensure issues can be handled effectively	Regularly check in on team members using scrums, identify issues if they arise and discuss these with the relevant individuals. For more serious cases, usage of the four peer assessments and contact with lecturers. **Owner: Kathryn Dale**
R-05	***Requirements*** Changes to requirements that require major design rework are proposed.	2	3	6	Avoidance – communicate with the client at the beginning of the project to understand their requirements fully. Contingency - use an easily extensible architecture to allow for adaptations if required.	Where changes to requirements are proposed, analyse and understand the impact on the project at the earliest opportunity. When developing architecture, consider how it could be extended if required. **Owner: Callum Marsden**
R-	***Requirements*** The customer fails to	1	2	2	Minimisation - regularly communicate with the client to prevent the need to change	Where changes to requirements are proposed, analyse and understand the

06	understand the resource impact of requirements changes.				requirements during or after implementation phases. Contingency - explain to the client the impact of requirements changes to the project, such as to the timeframe.	impact on the project and communicate this to the client. **Owner: Callum Marsden**
R-07	***Requirements*** Architecture specification is incomplete or contains conflicts.	2	2	4	Minimisation - re-evaluate architecture plan at the end of each implementation sprint to ensure it continues to meet requirements and is compatible with the project.	Prior to each sprint or implementation phase, analyse the proposed architecture specification, and add more detail and/or resolve conflicts. **Owner: Emilien Bevierre**
R-08	***Tools*** Software libraries use do not support the desired features. Technology Faults in reusable software components require repair before usage.	1	3	3	Avoidance - attempt to fully understand the featureset provided by a library prior to selection. Contingency - research alternative methods to perform required operations where needed.	Prior to implementation identify whether selected libraries provide the required features. Where these features are not available, perform research. **Owner: Daniel Benison**
R-09	***Tools*** Software development tools used do not work together.	1	2	2	Avoidance – verify tooling operates together as expected prior to selection	Check for issues in tooling by installing and performing test implementation operations, such as implementing a function and committing to the project. **Owner: Daniel Benison**

Colour Coding:
Red – high risk
Yellow – medium risk
Green – no/little risk
Orange – between high and medium risk.

The ownerships of each risk has been updated to reflect the change in personnel.

Bibliography

[1] B. W. Boehm, ‘Software risk management: principles and practices’, *IEEE Software.*, vol. 8, no. 1, pp. 32–41, Jan. 1991 [Online]. Available: 10.1109/52.62930.

[2] I. Sommerville, *Software engineering*, Tenth edition, Global edition. Harlow, Essex, England: Pearson, 2016.