Risk Assessment and Mitigation

There are many risks that can affect a project or product at every stage of development. It is important to identify these risks as early as possible in order to plan how to minimise their impact or avoid them completely. Conducting a risk assessment can also highlight any problematic requirements so they can either be removed or altered.

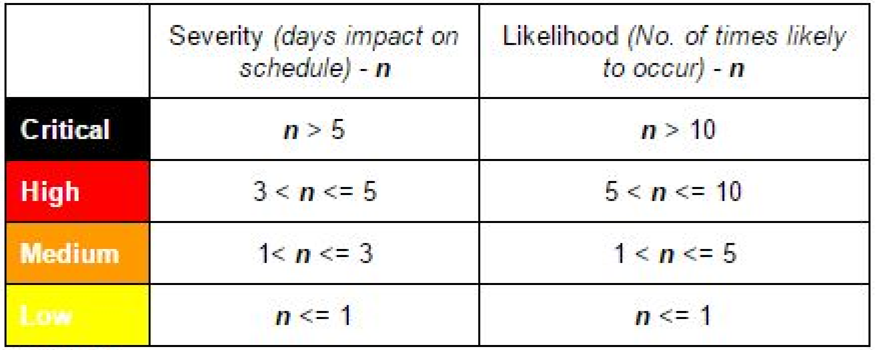
As a group we discussed various problems that might arise during the project and added them to a risk register (shown later). We later classified the risks, analysed how severe the impact on the development would be, the likelihood of the problem arising and looked at ways to mitigate them. We focused our efforts on mitigating the effect of the risks as opposed to preventing them as often some can be unpreventable. Using a custom algorithm which uses the level of severity and likelihood as discerning factors, we have ranked the risks on their need of addressing. In brief, it assigns a number to each rank for both likelihood and severity and adds them together to give an overall 'risk score'. Each risk is also assigned a unique ID to distinguish between one another. Additionally, each risk is assigned to a specific team member, it being their responsibility to manage and monitor their assigned risk(s).

Risk Classification

Risk classification is considered as an economical way of analysing risks and their causes by grouping similar risks together into classes[1]. One aspect of classification of risks is that it helps highlight classes which are going to have the most potential to cause problems. It can also inspire further identification of risks upon analysing the elements within classes. In addition to discussing an overview of any risks that might occur, we also used a taxonomy based risk identification method similar to [2] which is based on the SEI taxonomy. We used a revised version of the questionnaire in [2], removing any irrelevant questions, to give a concise outlook into the risks. As a group, we all read through each question while we were each assigned a role from 'de Bono's Thinking Hats' [3] to look at every possible perspective for each question. This enabled us to gather a more concise risk register which raises key areas of development that could cause the most harm to the project. We also considered the questions raised by the questionnaire [2] to help influence our ideas on mitigation of the risks we had already identified, allowing for a more useful and comprehensive mitigation plan.

Severity and Likelihood

To classify the severity and likelihood of the problem described in the risk register we have chosen four levels; '**Critical**', '**High**', '**Medium**', '**Low**', with 'Critical' being the worst-case scenario. The definitions of each level have been described in tabular form in *Figure 1.* We chose these boundaries for each level as they represent the difficulties faced by small projects such as ours well. This scaling down seems appropriate compared to those commonly used in industry due to the short timescale given to us. Where they may commonly deal with delays of months, we must work with shorter delays of days or even weeks.



	Severity *(days impact on schedule)* - ***n***	Likelihood *(No. of times likely to occur)* - ***n***
Critical	***n*** > 5	***n*** > 10
High	3 < ***n*** <= 5	5 < ***n*** <= 10
Medium	1< ***n*** <= 3	1 < ***n*** <= 5
Low	***n*** <= 1	***n*** <= 1

(Figure 1)

Key

- Additions we made

Risk Register

ID	Description	Class	Element	Severity	Likelihood	Rank	Mitigation	Ownership
01	Hardware failure, unable to use machine.	Development Environment	Development System	**Low**	**Low**	2	Many machines available on campus as alternative to home computer. Chances of complete university hardware system failure low enough to not require further mitigation.	HC
02	Use of development environment, possible bugs and limitations.	Development Environment	Development System	**Low**	**Low**	2	Use a well known and used development system where adequate support exists. Use existing experience from staff members to determine appropriate development environment.	TF
03	Failure to manage user expectations	Program Constraints	Program Interfaces	**Low**	**Medium**	3	Ensure regular contact with the client, providing e.g. prototype systems or models to ensure they know our progress, and to enable frank discussion of features they may ask for.	CH
04	Inadequate coordination during distributed development.	Development Environment.	Development Process	**Medium**	**Low**	3	Ensure we keep clear changelogs on the repository, and track who has created each part of the project to avoid repetition of task completion. Prior planning of work to be assigned can help minimise this.	CH
05	Personnel unfamiliarity with new technology	Development Environment	Code and Unit Test	**Medium**	**Low**	3	Allocate time prior to implementation to allow for training in new technologies otherwise look into ways of adapting the new technology to accommodate the way a developer is used to working. Look into alternative technologies which may be more suitable.	TF
06	Underestimating the need of planning the project	Development Environment	Management Process	**Medium**	**Low**	3	At the planning stage, look at every angle of the project - how it will affect certain users, how scalable is it, what level of detail is needed etc. Revisit and revise the plan as needed as the project progresses.	CH
07	Lack of customer involvement	Program Constraints	Program Interfaces	**Medium**	**Low**	3	Revisit the customer with any questions regarding development as well as keep them up to date with an major steps. Have periodic showcases and reports.	WH
08	Insufficient documentation on tools	Development Environment	Work Environment	**Medium**	**Low**	3	Spend time researching possible tools, evaluating their usefulness. If few tools are available allow time to learn to use the tools.	AP
09	Faulty external libraries, including	Development Environment	Development System	**Medium**	**Medium**	4	Research the user created libraries we want to use, minimising use of obscure third party libraries unless adequate support is available. Ensure adequate testing of libraries. If critical issue, contact library	HC

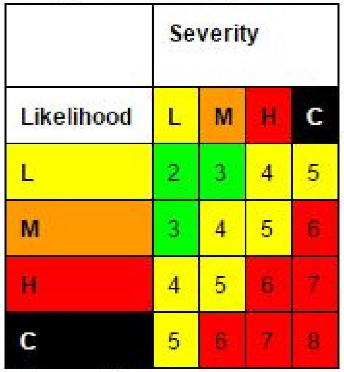
	malfunction and bugs.						admin for support.	
10	Unrealistic Schedules	Development Environment	Management Process	**Medium**	**Medium**	4	Research into how long similar projects have taken to complete. Also aim to meet milestones with incremental development. Reuse software were possible to reduce development time. Assign more people to a task which is behind on the schedule.	CH
11	Temporary loss of access to file hosting, unable to access data	Development Environment	Development System	**Medium**	**Medium**	4	Members should be allowed to take local copies of their sections in case of file system failure and then upload any changes after access is restored. Schedule should also allow sufficient time for temporary inability to access data for submission - i.e. not working right up to deadlines.	SD
12	Change of intended user	Program Constraints	Program Interfaces	**High**	**Low**	4	Not much that can be done here except have the word of the current user that the product will not be changed to target another user. Identify which changes are needed ASAP as this has the potential to affect every aspect of the project.	WH
13	Gold Plating (adding extra features that are not key to the product)	Product Engineering	Design	**High**	**Low**	4	When researching requirements, extract what is absolutely necessary and target those as the main goals when implementing the product. Gold plating can usually be done after the completion of the core concepts of the product, however they should still be noted.	TF
14	Lack of project management methodology	Development Environment	Management Process	**High**	**Low**	4	Use a well known management methodology to organise the team. If too far into the project, simply assign roles as a basic means for management.	CH
15	Requirement changes mid/post implementation	Program Constraints	Program Interfaces	**High**	**Low**	4	Not much that can be done here except have the word of the current user that the product will not be changed to target another user. Identify which changes are needed ASAP as this has the potential to affect every aspect of the project.	HC
16	Inadequate personnel programming skill level	Development Environment	Development Process	**High**	**Low**	5	Lots of time spent on practising require programming skills as early as possible. Perhaps follow a training course if unfamiliar with the language being used or assign a 'buddy' guide to the inexperienced developer.	SD
17	Underestimation of project scale.	Product Engineering	Design	**High**	**Medium**	5	Simplify project as much as possible and allow extra allocated time to what was originally expected.	TF
18	Temporary loss of staff due to illness or lack of	Development Environment	Management Methods	**High**	**Medium**	5	Schedule should have time margins to allow for catch-up. Other staff should be able to cover for some sections if necessary as stated previously - perhaps using a buddy system.	AP

	communication.							
19	Lack of commitment to the project (developers and users)	Development Environment	Work Environment	**High**	**Medium**	5	Praise good work and allocate fair amounts of work to each developer to encourage commitment.	CH
20	Unable to test scripts due to dependency issues and the way Unity works	Development Environment	Work Environment	**High**	**Medium**	5	If test tools are unavailable use other methods to test such as writing values to the console in order to determine the expected value is returned.	HC
21	Complete failure of file hosting, loss of all data	Development Environment	Development System	**Critical**	**Low**	5	Additional offline backups of critical elements to be taken. Chances of Google file system or GitHub repositories going down without backup are low enough to require no further mitigation.	SD
22	Misunderstanding/ambiguity of requirements	Product Engineering	Requirements	**Critical**	**Low**	5	Likelihood is low if lots of time is spent on the accurate gathering of requirements. This also ensures that the product is not incorrectly implemented later in development.	AP
23	Permanent loss of staff due to leave of absence or extended loss of contact	Development Environment	Management Methods	**Critical**	**Low**	5	At least two people should be assigned to do any section of the project. In the event of long term loss of personnel, immediate prioritisation and re-scheduling should occur.	CH
24	Inadequate amounts of testing to prevent bugs and error upon submission.	Product Engineering	Integration and Test	**High**	**High**	6	Test the individual modules of the game as thoroughly as possible, attempt to test several variations of the end product (exhaustive testing infeasible due to dynamic / random nature of game). Get external actors to playtest the game for a fresh perspective.	SD

Analysis of Risk Level

Upon analysing the likelihood and severity and calculating the rank, we produced a table (*figure 2*) which defines what we as a group, think are acceptable risks, manageable risks and extreme risks. Each displays the rank, while each colour describes how acceptable we feel each rank is.

Rank against Severity/Likelihood



	Severity			
Likelihood	L	M	H	C
L	2	3	4	5
M	3	4	5	6
H	4	5	6	7
C	5	6	7	8

(Figure 2)

- **Acceptable risks** - risks that we should be able to manage and only cause minor problems if they were to come to fruition.
- **Manageable risks** - we should be wary of these risk levels but should still be able to cope with using the correct mitigation.
- **Extreme risks** - these risks should be heavily managed using avoidance strategies and mitigation. Could potentially cause long delays all the way up to complete project failure.

References:

[1] C. Pandian, Applied Software Risk Management a Guide, United States of America Auerbach Publications, 2007.

[2] Carr. Marvin, Konda. Suresh, Monarch. Ira, Walker. Clay, and Ulrich. F., "Taxonomy-Based Risk Identification," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-93-TR-006 , 1993. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11847

[3] E. De Bono, Six Thinking Hats, Little, Brown and Company, 1985.