CS261 - Requirements analysis

Group 4: Maddie Amza, Param Bhatia, Matthew Jack-Kee, Rahul Kundi, Yasmine Mountaser, Brindan Yasodaran

1 Introduction

Studies have shown that a substantial number of software development projects fail to meet its objectives, such as schedule, budget, quality, or functionality goals and often do not even reach completion. The effects of software development projects failing can be devastating and include cost overruns and damage to the reputation of the organisation with investors and clients. To prevent failure, proper planning and risk management must be carried out.

We have been tasked to design a system which will improve and save time in the risk management process by automating risk quantification and identification. This will provide information that will help project managers adjust the project as needed to keep it on track. We realise that the consequences of our system providing inaccurate information could be devastating to the success of a project. The system will therefore be thoroughly justified with research and will consider a wide range of factors concerning the project to provide an informed and unbiased quantification of risk.

2 Definitions and market research

Definition of risk: In the design of this system, we have interpreted the "risk" of a software development project to be the probability of the project not being operational at the time of its final deadline and within its budget, considering hard metrics such as team size, levels of activity and completion of deliverables (metrics that can be analysed objectively), and soft metrics such as team morale, health or communication (metrics which involve more subjectivity). This means that tracking the success of a project beyond its completion is out of scope, and that our system will focus mostly on the impact of the team's internal operation throughout development of the final product.

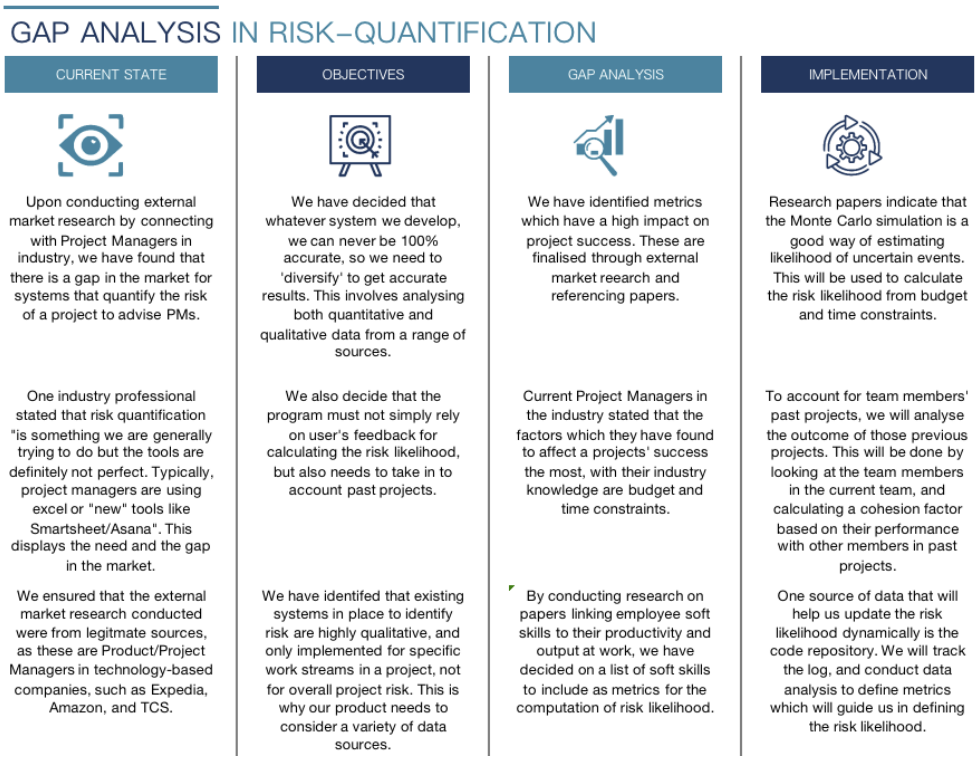

Figure 1: GAP analysis

3 Requirements

3.1 Functional requirements

The following requirements have been ranked according to the MoSCoW[2] hierarchical classification, and color coded based on the area of functionality they concern: orange for application input, pink for risk function computation and its output and blue for auxiliary system functions. The numbers were added for referencing in the Planning and Design document.

Table 1: Functional requirements

	Requirements and justification
M1	C: The system must allow both Project Managers and Developers to create user accounts. D: The system must implement a registration and a login page, that would prompt the user to input their username, password and account type (manager or developer).
M2	C: The system must allow a project manager to create and track multiple projects D: The system database must contain separate project and project manager entities and link a manager to all the projects they create.
M3	C: The system must allow a project manager to assign different developers to different projects; team developers must be able to partake in multiple projects at a time. D: The system must be able to track assignments of developers to projects within the database.
M4	C: The system must allow only the project manager to give information on overall project status. D: Deadlines and budget should be provided by the project manager in best/average/worst case estimations, which will be used in Monte Carlo simulations. D: Other project specific metrics should be provided through a standardised point / ranking system, to manage ambiguity and possibility of input error.
M5	C: The system must track soft metrics such as enthusiasm and health at an individual (team member) level D: The system should allow project members to input updates to personal progress, through a standardised input method (slider, scale rankings etc), in order to decrease ambiguity and possibility of input error.
M6	C: The system must give a first evaluation of risk based on the data received from the users and how it compares to documented benchmarks D: The project manager should be able to view the risk of a project in a red/amber/green zone estimation, based on the internally computed risk function. D: Both the user input and the pre-existing knowledge base of the system must be factored into the risk computation Justification: benchmarks may come from academic literature, earlier projects from the application database or both, to inform a reliable baseline for detecting elevated risk of a project failing
M7	C: The system must update its risk evaluation every time the project manager and team members provide new/updated information D: The project database must be updated dynamically as data is received from the user interface, and the risk function evaluation should be refactored based on the updated metrics
M8	C: The system must update its risk evaluation periodically to account for updates in the code of each software project within its database D: A set time interval should be chosen for recomputing code-dependent metrics for each project (based on data collected from its associated repository), as performing updates every time a line of code is changed would be too computationally expensive.
M9	C: The system must identify elevated risk areas and notify the project manager of what those are. D: The system should keep track of partial results used in the cost function estimation, rank them in order of severity (level of impact) and output the most impactful ones.
M10	C: The system must provide a way of tracking the level of authority each user has over a specific project D: Tracking authority can be done by linking each project to its manager in the database and by creating separate accounts (and database objects) for project managers and developers. Justification: Project developers should be allowed limited access to a project – they should be able to provide personal updates, but not see the overall risk assessment or make changes to the project itself, as this might negatively impact team morale.

S1	C: The system should provide suggestions to reduce the overall riskiness to the project manager D: When areas of risk are identified, standard risk-reducing advice could be provided from a pre-computed knowledge base; for budget and time, the system should output how much they should be extended to reach a low-level of risk.
S2	C: The system should account for how well members of the team have collaborated in the past D: The system should be able to derive the successfulness of collaboration between two team members from the database, by subtracting the number of failed projects they both worked on from the number of successful projects they both worked on (the baseline cooperation factor is 0). Those numbers are computed for all pairs of developers in the team, then factored into the risk function
C1	C: The system could make use of past projects for reassessing risk benchmarks. D: The system could use a machine learning algorithm to update the weightings for hard and soft metrics that are used to evaluate the overall riskiness.
C2	C: The system could keep track of code metrics for monitoring overall completion of deliverables and team member performance. D: We can accomplish this by linking the project's code repository to the website. Justification: We can take code metrics into account for tracking completion of deliverables, and potentially the effectiveness of team member interactivity.

3.2 Nonfunctional requirements

1. C: The system should : Remain responsive whilst being able to process the risk assessment data
   D: The website should be dynamically updated by making use of AJAX.
   D: The system should also use multithreading for time-consuming computations (Monte Carlo simulations).
2. C: The system must ensure confidential data is stored and processed in a secure manner
   D: The application code should include libraries for encryption of passwords.
3. C: A regular user should have a clear understanding of how our application works after only using it once.
   D: May provide a tutorial or written instructions upon first use of the application, use sliders / other conventional rating systems for measuring metrics wherever possible, and name sections of the user interface appropriately
4. C: Have a consistent user interface, that follows accessibility guidelines
   D: Use similar themes and styling (e.g.: fonts, colours etc.), HTML tags that maximise accessibility, colours easily distinguishable by colourblind people etc.
5. C: Perform rigorous testing through formal validation techniques
   D: Each requirement should be verifiable through an easily implementable test, following the testing plan described in the design document

3.3 Out of scope requirements:

1. The system will not keep track of all interactions within a software development project such as delays between interdependent tasks, unpredictable technical challenges and success of the projects in regard to the business sector they're deployed in. Although these are factors contribute to the failure of a software development project, the system should be used alongside other standard risk assessment techniques to account for this.
2. The systems suggestions are not expected to provide a complete risk management plan but only to provide guidance on managing and mitigating the risks identified.
3. The system is not expected to differentiate between members working in different development roles within a single software development project.
4. The system is not expected to handle a project manager simultaneously contributing to other projects as a developer (the roles are mutually exclusive).
5. The system is not expected to handle security beyond validation of login information; all other input should be collected through a standardised method (rating, sliders etc), bypassing the need for sanitisation.
6. The system is not expected to integrate functionality of external systems belonging to clients.
7. The system does not actively ask developers to update their skills; it is assumed that when a developer gains a new programming skill, they will personally add it to their profile.

4 Project management

4.1 Methodology

Our team opted for an agile development methodology based on the Scrum model [5], as most of our software developers have similar levels of experience and one of our primary goals is to be able to observe ongoing development. In practice, as access to customer feedback is limited, we have organised our activity through the 4 main development stages (as listed in [1]) in the following way:

- Software specification (1) and design (2): These stages will make up the "Outline planning phase" of the scrum process.
- Software implementation (2) and evolution (4):
  - Software implementation will be organised in the form of sprint cycles.
  - Because we don't have a way of easily communicating with the customer, the features selected for development by each sprint cycle should be chosen based on the priority of the requirement that they fulfil: the higher the priority, the sooner it should be selected for implementation.
- Software validation (3) – this will be accomplished through unit / feature tests at the end of each sprint cycle, and a final system testing period during the "Project closure" stage of the scrum process.

4.2 Team Roles

Most of our team members have similar levels of experience, which further justifies the efficiency of using an agile methodology and implementing a relatively flat team structure. The roles and responsibilities of each member are listed in the table below.

Role	Responsibilities	Team member names
Project Manager	Manages deadlines, keeps the team on track and makes organisational decisions	Brindan Yasodaran
Product Owner	Conducts market analyses, based on which the product vision is defined.	Param Bhatia
Scrum Master	Organises Scrum events and ensures sprint goals are met	Yasmine Mountaser
Business Analyst	Identifies the utility of the project, documents its progress and analyses potential solutions	Brindan Yasodaran
Frontend developer	Handle the design of the user interface of the website	Maddie Amza, Matthew Jack-Kee, Param Bhatia
Backend developer	Handle the website's structure and overall functionality	Matthew Jack-Kee, Rahul Kundi, Maddie Amza
Software Tester	Designs the tests for the system, ensures implementation works as expected and all requirements are met	Yasmine Mountaser

Table 2: Team roles

4.3 Communication and meeting plan

Our team has resolved to meet at least twice a week for the duration of the project, once during the first half of the week (Monday or Tuesday) and once during the second half of the week (Friday or Saturday). Meetings are taking place in person, for about 1h, with the option open for hybrid meetings in case of emergencies, or online meetings on Microsoft Teams during the weekend.

For communication between meetings, we opted for using Discord, as most of our members have prior experience with it and it provides other useful functionalities, such as highlighting important messages by pinning them and allowing us to separate conversations by topics (e.g., Having separate channels for discussing backend and frontend coding). Otherwise, as we are already using teams for online meetings, we decided to make use of it for file-storing as well (code will be accessed through a GitHub repository).

References

[1] James Archbold. CS261 Software Engineering. https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs261/.

[2] Wikipedia authors. MoSCoW method. https://en.wikipedia.org/wiki/MoSCoW_method.

[3] Michael Bloch, Sven Blumberg, and Jürgen Laartz. Delivering large-scale IT projects on time, on budget, and on value. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value, 2012.

[4] Katie Terrell Hanna and Francesca Sales. Gap Analysis. https://www.techtarget.com/searchcio/definition/gap-analysis#:~:text=A%20gap%20analysis%20is%20a,assessment%20or%20need%2Dgap%20analysis.

[5] Ken Schwaber and Jeff Sutherland. The Scrum Guide. https://scrumguides.org/scrum-guide.html.

[6] Karl Wiegers. Writing Quality Requirements. http://www.processimpact.com/articles/qualreqs.html, 1999.