Sprint 4: Risk Management Log

By:

Camil Bouzidi (40099611)

Radley Carpio (40074888)

Bicher Chammaa (40096200)

Nimit Jaggi (40032159)

Matthew Kevork (40063824)

Cedric Martens (40086877)

William Morin-Laberge (40097269)

Adrien Tremblay (40108982)

Pierre-Olivier Trottier (40059235)

A report submitted in partial fulfillment of SOEN390.

Concordia University

April 7th, 2021

Introduction

In this report, an initial risk analysis for the EPIC Resource Planner software product is described and explored. Given that this is the first development sprint related to the project, many of the risks outlined in this document are subject to change. This preliminary exploration of risk analysis has the goal of setting a baseline for upcoming sprints and meetings with stakeholders. First, different categories of risks will be explored. Then, for each of these categories, risks will be exposed through different elicitation techniques. Each of these risks will then be assigned qualitative values, which will then be turned into quantitative values to be used to rank these risks and see which will be tackled first. In order to tackle the risk, one of the techniques outlined in the risk management plan will be applied.

The following information is to be maintained for every risk (all of which is explained in the Risk Management Plan):

- Description
- Date of identification
- Probability
- Impact
- Rating/Priority
- Response Strategy
- Status
- Owner (to be defined in later sprints)

Updates

- Sprint 2: Risk 1.2, 1.3, 1.4, 2.2 and 2.3 were marked as Happening. All of them are being addressed (see the individual logs), and the effects of the response strategies will be collected between the 2nd and 3rd sprints.
- Sprint 3: Risk 1.2, 1.3, 1.4, 2.2 and 2.3 were all updated. Risk 2.5 was identified for the first time and mitigated quickly.
- Sprint 4: Risk 1.2, 1.3, 1.4, 2.2 and 2.3 were all updated.

Identification

Risk Categories

After deliberation and discussion with the product owner, the stakeholders, the development team as well as an exploration of the problem domain, the following risk categories have been chosen:

- Administrative or Process: This has to do with staff management, meetings, stakeholders and any other situation where an error in the code or the problem domain is not to blame.
- Code or QA: This has to do with any framework, API or coding defect or issue. This includes the bugs that arise as a result of testing.
- Problem Domain: While it is not the software development team's responsibility to run the business that uses the EPIC Resource Planner, the team should still analyze the problem domain and highlight any risks born out of the software's interaction with the problem domain.

Risks

For the previously mentioned categories, the following risks have been found:

Administrative/Process Risks

- **Risk 1.1**: Developer drops out of the team.

- Description: for any reason, a developer drops out of the team and does not work on the project anymore.
- Date of identification: January 17th, 2021
- Probability: High (50%)
- Impact: Medium (0.35)
- Rating: 0.175
- Status: Potential
- Response Strategy:
  - Mitigation: Developers shall perform as much knowledge sharing that they can to avoid losing knowledge when a developer leaves the team.
  - Transfer. The SOEN 390 administration will have to change the scope of the project if necessary.

- **Risk 1.2**: Developer must be relieved of work because of illness or other circumstances.
  - Description: Should a developer need some time away from the project due to an illness or circumstances outside of their control, their share of work must be handled.
  - Date of identification: January 17th, 2021
  - Probability: Very High (80%)
  - Impact: Medium (0.35)
  - Rating: 0.28
  - **Status: Happening**
  - Response Strategies:
    - Mitigation **(CHOSEN)**: Developers should have high visibility on others' work through weekly meetings. Should very complicated features need to be implemented, pair programming will be implemented.
      - **Result:** Work for backend setup was taken over quickly by the front-end team. Additionally, the focus was shifted towards core features which made it so that, even if a developer was relieved of work, no one was massively slowed down.
    - Transfer: The SOEN 390 administration will have to change the requirements of the project if necessary.

- **Updated: 02/13/2021**. Given that most developers had other engagements due to midterms, the status was changed to happening. Some issues were shared between front-end and back-end developers.
- **Updated: 03/17/2021**. This risk is still happening as there was a March break and the entire team needed some downtime. The rest of the sprint was however well managed.
- **Updated: 04/07/2021.** This risk is still happening. Given that the end of the semester is nearing, most of the developers had assignments or final projects to complete.

- **Risk 1.3**: Feature will not be delivered on time.
  - Description: If a feature is planned for a sprint and cannot be delivered by the end of the sprint, it is late.
  - Date of Identification: January 17th, 2021
  - Probability: High (50%)
  - Impact: High (0.5)
  - Rating: 0.25
  - **Status: Managed**
  - Response Strategies:
    - Mitigation **(CHOSEN)**: Features will be ranked by size to get a better idea of how they should be prioritized. Once prioritized, if a feature is not delivered at the end of the sprint, it will be implemented by the end of the next sprint.
      - **Result:** All issues/features have been ranked and prioritized. A few features were not implemented due to time constraints. This is due in part to poor estimations, and unexpected dependencies between issues. They have been moved to the next sprint.
  - **Updated: 02/13/2021**. Two features (Product manufacturing, and Parts Manufacturing) were not completed in time for sprint 2. As a result, they have been moved to sprint 3.
  - **Updated: 03/17/2021**. This risk is still happening, but in a slightly lesser capacity because the team has delivered features that were considered to be late.

- **Risk 1.4**: Poor Estimations
    - Description: The estimation of size and effort needed for various features and user stories are off, leading some features to being pushed to other sprints.
    - Date of Identification: January 28th, 2021
    - Probability: High (50%)
    - Impact: High (0.5)
    - Rating: 0.250
    - **Status: Happening**
    - Response Strategies:
        - Mitigation **(CHOSEN)**: As the project goes on and the development team becomes more comfortable with the technology and development tools, the estimations made by the development team will be more accurate both in terms of size and length. In order to accelerate this evolution, any task whose estimation was wrong will be reviewed by the development team at the end of the sprint in order to understand why or how it was misestimated.
            - **Result**: because the team was ahead in the development of the project, the features that were not necessary for the sprint were pushed back for the next one. This was only the case for a few features.
    - **Updated: 02/17/2021**. Given that the team was already ahead, a lot of activities planned for this sprint went beyond the requirements set out by the SOEN 390 administration (TAs and Professor). Status was changed to Happening to reflect this. In order to mitigate this risk, the team took another look at the present tasks and focused on the more important ones (which were specifying the backend and adding buttons to add any entity such as material, part, product, etc.).
    - **Updated: 03/07/2021**. This risk is still happening, the team tried to mitigate it by meeting early in the sprint and letting everyone declare how much they could or could not take on.
    - **Updated: 04/07/2021.** This risk is still happening. Some issues required more effort due to a change in library dependencies. Other issues took longer or more work than expected. This risk is being mitigated by having more than one person participate in the development of the scheduled features.

- **Risk 1.5**: Poor risk management

- Description: The initial plan for the risk management plan is unclear or leads to a poor risk management strategy where some risks go completely untouched and slow the production of the team.
- Date of identification: January 20th, 2021
- Probability: Medium (35%)
- Impact: Medium (0.35)
- Rating: 0.1225
- **Status: Happening**
- Response Strategies:
  - Mitigation **(CHOSEN)**: The risk management plan is to be revisited as soon as a problem is noticed by the development team. A meeting will be called in order to develop new metrics or methodologies related to risk management.
    - **Result:** For the 2nd sprint, a "Updated" section was added for every risk that has changed from one sprint to the next. This allows for better risk tracking and risk management.
- **Updated: 02/13/2021**. It was noticed that there was not a good way to keep track of changes to risks. This risk was changed to happening. Additionally, a section "Updated" was added to the risk registers in order to better keep track of any possible changes.
- **Updated: 03/09/2021**. This risk is still happening, but in a slightly lesser capacity because the team has a better understanding of the given risks for the project.
- **Updated: 04/07/2021.** This risk is happening in a lesser capacity. The previous sprints have helped the team to better plan the sprint and handle risks and issues more accurately.

- **Risk 1.6**: Sudden growth in requirements
  - Description: It is possible that stakeholders ask for the development team to suddenly develop new features and have additional requirements.
  - Date of identification: January 20th, 2021
  - Probability: Low (20%)
  - Impact: Very High (0.8)
  - Rating: 0.160
  - Status: Possible

- Response Strategies:
  - Mitigation: Meetings with product owners are scheduled for every sprint. The development team will also constantly refer to the requirements document to verify whether or not a new requirement falls within the guidelines previously outlined.
  - Accept: Should requirements not fall within the previously agreed upon guidelines, the software development team will tell product owners or stakeholders that the feature will not be implemented.

Code or QA

- **Risk 2.1**: Poor End-user Engagement
  - Description: The product does not incorporate enough user feedback into its design. As a result, the end-user is unlikely to be satisfied with the product and is unlikely to adopt it.
  - Date of Identification: January 17th, 2021
  - Probability: Medium (35%)
  - Impact: Very High (0.75)
  - Rating: 0.2625
  - Status: Monitoring
  - Response Strategies:
    - Mitigation: More user testing, surveys, focus groups should take place to gain a better understanding of the user's requirements and expectations. Users should also be given working prototypes for their feedback.

- **Risk 2.2**: Poor Quality code
  - Description: The codebase is of poor quality and puts the project into technical debt reducing the agility of the project. Code may be of poor quality because it is hard to read, inefficient, lacks good tests, etc... Poor quality code can lead the codebase to becoming needlessly hard to maintain, expand upon and a product that is of poor performance or bug prone.
  - Date of Identification: January 17th, 2021
  - Probability: Medium (25%)
  - Impact: Medium (0.35)
  - Rating: 0.0375

- **Status: Happening**
- Response Strategies:
  - Mitigation **(CHOSEN)**: More code reviews, software testing, static verification should be done. More software testers could potentially be onboarded for the project. Additionally, a clear coding standard could be put in place for all developers to follow.
    - **Result:** Now, reformatting documents and running ESLint is part of the activities that a team member must do before submitting a document. The pull requests are now much cleaner.
  - Transference: Software QA could potentially be outsourced to another development team.
- **Updated: 02/21/2021**. Multiple pull requests had to be reviewed and this risk's status was set as Happening. It was enforced that developers must format documents through their IDE of choice and that they must run ESLint in order to make sure that declaration and use of variables and components is as efficient as possible.
- **Updated: 03/17/2021.** The code quality has increased, and the team is looking at new ways to make it more efficient as the project is over halfway done.
- **Updated: 04/07/2021.** Improvements to code quality have been increasing. Some documents were refactored to improve the quality of the codebase. Each pull request is being thoroughly reviewed to ensure code quality.

- **Risk 2.3**: Compromising on Designs
  - Description: The developers rush software design and architecture to jump straight into coding. This can result in a less agile, extensible, and performant end-product. This often leads to a lot of work going to waste because of constantly changing software architecture or design.
  - Date of Identification: January 17th, 2021
  - Probability: Medium (30%)
  - Impact: Medium (0.35)
  - Rating: 0.105
  - **Status: Happening**
  - Response Strategies:

        - Mitigation **(CHOSEN)**: Software architecture and design experts could be onboarded. The development team should seek approval for architecture and design from tech leads or management. An Agile approach to software design could be implemented where software is designed iteratively.
            - **Result**: There is now better communication between the design owners and the implementers in the team. If a feature is not implemented exactly as designed, it is pushed if and only if the basic implementation is similar to the design. Any additional details are then set as separate tasks that can be pushed to another sprint.
        - Transference: Software Architecture could potentially be outsourced to another development team.
    - **Updated**: **02/21/2021**. There were some designs that initially displayed much more information, but they were found to be somewhat bloated and made the display of information complicated on mobile. The status was changed to Happening. Over time, by implementing some of the features, it was noticed that some information was not necessary for some catalogs and the design was revisited.
    - **Updated**: **02/24/2021**. No naming conventions for certain class attributes, schema items, interface names, etc... were implemented. The result was each individual developer making up their own convention, leading to inconsistencies in the code. For the future, a naming standard shall be collectively decided upon.
    - **Updated: 03/17/2021.** While a final naming convention has not been universally agreed upon for minor variables and components, multiple team members were reminded of the different names for the major schemas & components in the backend and now understand how to name different front-end components that interact with them.
    - **Updated: 04/07/2021.** This risk is still happening. Early design choices like the chart library were revisited because of limited functionality.

- **Risk 2.4**: Gold Plating
    - Description: Gold plating is when developers waste time implementing features not specified in the functional requirements or waste time writing code based on a personal unapproved codebase decision they have taken.
    - Date of Identification: January 17th, 2021
    - Probability: High (60%)

- Impact: Low (0.2)
- Rating: 0.12
- Status: Potential
- Response Strategies:
  - Mitigation: Developers who do Gold Plating should be severely punished to deter future instances of this occurring.
  - Mitigation: To provide developers with a clear understanding on which features are to be implemented and how, each feature should be made into a user story, and each user story should be broken into substories.
  - Acceptance: Developers will always make decision while coding. There is no clear line between what decision is acceptable to be taken by a developer on the fly, and a decision that needs to be approved by other developers. A few instances of gold-plating will always be unavoidable and is acceptable.

- **Risk 2.5: Improper Team Member Setup**
  - Description: With agile projects, while documentation pertaining to the customer and product owner can often be modified, it often happens that setup documents for the development team (which should help any new member set up their station) falls behind in terms of recency and accuracy of information.
  - Date of Identification: March 12th, 2021
  - Probability: Very High (80%)
  - Impact: Low (0.2)
  - Rating: 0.16
  - **Status: Managed**
  - Response Strategies:
    - Mitigation **(CHOSEN)**: In-house documentation about setup should be updated when major setup changes are made by someone.
      - **Result**: The team has a better idea of what to expect when setting up for the first time or refreshing their setup.
    - Mitigation **(CHOSEN)**: Pair programming will be used when a setup refresh is required.
      - **Result**: The person setting up for the first time gets a better understanding of every piece of the setup since someone is there to walk them through any unclear parts.

        - Acceptance: The person setting up for the first time will simply have to figure out for themselves what is going wrong.
    - **Updated: 03/12/2021**. Someone was refreshing their setup in order to fix an issue and it was noticed that there was a small discrepancy between the in-house documentation for the setup and the actual setup. Over a voice call, another programmer helped this person through the different part of the setup and the rest of the team was warned.

Problem Domain

- **Risk 3.1**: Market shifts
    - Description: The market shifts resulting in a change in requirements or product viability. For EPIC Resource Planner this could mean that bicycles become less popular, or certain types of bicycles shift in popularity.
    - Date of Identification: January 17th, 2021
    - Probability: Low (5%)
    - Impact: Medium (0.35)
    - Rating: 0.0175
    - Status: Potential
    - Response Strategies:
        - Acceptance: When the change to the market is relatively small not much can be done, and the risk should simply be accepted.

- **Risk 3.2**: Poor domain understanding
    - Description: The software team does not understand the problem domain well enough, and this is reflected by incomplete or inaccurate software requirements.
    - Date of Identification: January 17th, 2021
    - Probability: Medium (40%)
    - Impact: High (0.5)
    - Rating: 0.2
    - Status: Potential
    - Response Strategies:
        - Mitigation: More market research should be done to build a better understanding of the domain problem. Additionally, domain experts could be hired.

- **Risk 3.3**: Legal risk
    - Description: The project needs to abide by laws or regulations. If these laws arise during the project or are not identified an early stage of the project development, there could be a substation determent to the product in terms of needing to reevaluate requirements.
    - Date of Identification: January 17th, 2021
    - Probability: Low (15%)
    - Impact: Low (0.2)
    - Rating: 0.03
    - Status: Potential
    - Response Strategies:
        - Mitigation: The law should be thoroughly examined to find any laws that may have bearing on the project.

- **Risk 3.4**: External hazards
    - Description: External hazards are catastrophic events such as floods or earthquakes that may severely impact project assets or staff.
    - Date of Identification: January 17th, 2021
    - Probability: Low (1%)
    - Impact: Very High (0.8)
    - Rating: 0.008
    - Status: Potential
    - Response Strategies:
        - Mitigation: There should be stringent safety measures in place in the event of any catastrophic event to minimize impact caused by such an event.

Qualitative Risk Analysis Table

Impact	Very High	1.6, 3.4	2.1		
	High	2.4	3.2	1.3, 1.4	
	Medium	3.1	1.5, 2.2, 2.3	1.1, 1.2	
	Low	3.3, 2.5			
		Low	Medium	High	Very High

Probability