Software Development Project – Requirements Analysis

Overview

In this activity, students will identify users, use cases, features, and quality requirements **for the specific software that each team will develop.**

Each team will produce a use case diagram, a features list, an abuse cases diagram, a utility tree for quality attributes and scenarios, and will prioritize the features and quality requirements.

Each team member will fairly contribute to the software requirements analysis activity. The requirements analysis lead, together with the project manager, are responsible for the coordination and timely completion and submission of the required deliverables for this activity. The effort spent in this activity needs to be recorded, and project progress needs to be tracked against the plan. Risk management has to be performed throughout this activity.

Description

Use the provided *LMS SRS Example* requirements and the UMD ELMS system as a starting point/inspiration to define the features and quality requirements of the specific software **your team** will develop in this course. Keep in mind, though, that your software will **not** be a replica of the UMD ELMS. Your software should operate in a different learning environment, serve a different type of users, will have a smaller scope, different attributes, and - most probably - a different design and implementation than the UMD ELMS.

For the software that you will develop in this class project:

1. Name the product and provide a brief description
2. Create a context model, using UML Class diagram
3. Create a use case model, using UML Use Case diagram
4. Identify a list of functional features derived from the use cases. For each identified feature:
    a. Specify the role/perspective of its users (e.g., student, instructor, admin…)
    b. Provide a brief description (one-two sentences)
    c. Assign a number from 1 to 3 (where Low = 1, Medium = 2, High = 3) for its utility (importance) to the users
    d. Assign a number from 1 to 3 (where Difficult or high risk = 1, Medium difficulty or risk = 2, Rather easy and low risk = 3) for the estimated difficulty or risk to implement that feature
    e. Compute the priority score (utility * difficulty). This score will help you decide what features will be within the scope of your software for design and implementation.
    See the provided Requirements Analysis template.
5. Document the bi-directional traces between these features and use cases, in a matrix
    See the provided Requirements Analysis template.
6. Create an abuse case model, using UML Use Case diagram

7. Identify security scenarios derived from the abuse cases
8. Document the bi-directional traces between these security scenarios and abuse cases, in a matrix
   See the provided Requirements Analysis template.
9. Describe the two most critical abuse cases that appear in your abuse case diagram: one abuse case description in textual/natural language (using the template provided in class) and the other abuse case description in graphical format (using UML activity diagram).
10. In addition to security scenarios and requirements identified above, also identify the other quality attributes and respective scenarios for the software to be developed. For each identified scenario:
    a. Provide a brief description (one-two sentences)
    b. Assign a number from 1 to 3 (where Low = 1, Medium = 2, High = 3) for its utility (importance) to the users
    c. Assign a number from 1 to 3 (where Difficult or high risk = 1, Medium difficulty or risk = 2, Rather easy and low risk = 3) for the estimated difficulty or risk to implement that scenario
    d. Compute the priority score (utility * difficulty). This score will help you decide what scenario will be within the scope of your software for design and implementation.
    See the provided Requirements Analysis template.
11. For the top three highest priority quality scenarios, provide their description using the SEI Template (Source of stimulus, Stimulus, Environment, Artifact, Response, Response measure).

For creating the UML diagram, you will use a UML tool.

Deliverables and Schedule

The deliverable for this activity is a pdf file containing:

- A brief introduction and description of your software product
- The context model
- The use case diagram (use cases' description is **not** required)
- The table of features, with brief description, utility, difficulty, and priority scores
- Bi-directional traceability matrix between features and use cases
- The abuse case diagram
- Bi-directional traceability matrix between security scenarios and abuse cases
- The description of the two most critical abuse cases (one abuse case described using natural language the template provided in class, and one abuse case described graphically, using UML activity diagram notation)
- The quality "utility tree" (it can be in a table format), with scenarios names, brief description, utility, difficulty, and priority scores
- The list of functional features and quality scenarios with high priority that will be further designed and developed
- Description of the top three high priority quality scenarios, using the SEI template.

This document must look professional, that is, will have a cover page (including the product name and team members' names), table of content, section numbers, will be free of spelling and grammar errors.

This file must be submitted through ELMS (Canvas), **by the deadline specified in ELMS**.

Peer review will be performed by the respectively assigned *Reviewer Team*. The Reviewer team must record issues and questions in ELMS, and discuss them with the *Author Team*.

Guidance: The grading criteria contained in the rubric associated with this assignment in ELMS should be used as review criteria for the Team to Team (T2T) review.

For developing and reviewing use cases and abuse cases, use the Formation Rules and the Heuristics in Chapter 6 of the class textbook by C. Fox:

Use Cases Formation Rules
Every use case diagram must have:

- At least one use case
- At least one actor
- At least one actor associated with each use case
- At least one use case associated with each actor
- No association line between actors
- No association line between use cases
- Name every actor and use case
- Not label any association line

Use Cases Heuristics:

- The product (software that you are developing) shall not be an actor.
- Actors should be named with noun phrases.
- Use cases should be named with verb phrases.
- An actor's goal should be achieved in one use case.

Additional use case review considerations:

- Review the stakeholders-goals list to make sure no actors are missing.
- Review the needs list to make sure no uses cases are missing.
- Review constraints and limitations to make sure they are no violated.
- Check that the collection of use cases covers all externally visible behavior.
- Check the diagram against the use case heuristics above.