INTRODUCTION TO REQUIREMENT ENGINEERING IN SENG2021

AGENDA

- Why requirement Engineering
- Project Scoping(Problem Statement)
- Features and how to present them
- User Stories

DISCLAIMER

- Part of the material presented here are taken from the SENG1031 lectures presented by Sci Prof. Boualem Benatallah.
- No “Actual” software code was hurt during the creation of these slides.

SOME REFRESHERS (REF1 & REF2)

- **User Requirement**: criteria/constraints to satisfy (natural language, other artefacts / use cases), problem statements / customer needs
- **System Requirement**: detailed descriptions (functions, constraints), services and operational constraints, interface/contract between customer/development team (e.g., class diagrams, state diagrams, functions, pre/post conditions, business rules)
- **Functional Requirement**: the main functions provided by the software (e.g., send messages, receive messages, add courses)
- **Non Functional Requirement**: the constrains on the functions provided by the software (e.g., security, performance, reliability)

PROBLEM STATEMENT (PART1)

- Few sentences (e.g., one or 2) to describe specific user/customer needs
- Identify a problem from customer perspective
- Understand the current state of the world
- **Problem statements** are not bug reports (the problems may never been raised before, problems were not in the requirements of existing projects), focus on identifying new requirements (not bug fixes, which errors of implementation not missing requirements)

PROBLEM STATEMENT (PART2)

- **Problem statements** can be enriched with references, supporting materials (e.g., market research reports, videos, government policy statements), but should be made clear and separate from supporting artefacts
- **Problem statements** are input to what is called feature statements or scenarios. Features / scenarios describe the state of the world when the project is done (e.g., the project will support category, interests based search in addition to keyword search, the project will support search across 4 most popular MOOC platforms).

EXAMPLE OF A GOOD PROBLEM STATEMENT

- *There is no feedback system for a particular course.*
- *There is no reputation / popularity system for users or course instructors.*
- *There is no general forum for the students to interact with each other, only a forum for each individual course.*
- *There is no Intuitive notification system.*
- *Major social hubs are not integrated into the platform*

FEATURES (REF3)

- Describe a way to solve a problem mentioned in the problem statement
- Behaviour oriented : focus on what the software will do when delivered
- Customer / Engineering collaboration : requirements created jointly between customer and engineering team, using, controlled natural language
- Simple to understand, testable, have business value
- Used as input in software management tools, e.g., to review, to generate input/output test cases, to monitor project progress
- Problem statement VS feature: current state of the world VS the state of the world when the project is delivered.

FEATURE EXAMPLE

General project information : MOOC system (called myMOOC) provides access to courses by groups of people involved in those courses.

Feature: As a teacher, I can access myMOOC to search and manage courses

COMPLEX FEATURES

- Complex feature: As a student, I can search courses, enrol in a course or drop it (complex feature)
- Concrete/small features

1: As a student, I can search courses by tags

2: As a student, I can enrol in a course

3: As a student, I can drop a course

FEATURES NOTATION

- Based on **Connextra** notation, a feature is described as follows : **role**, **goal**, **task**

Feature **name**

As a **[the role of the user]**

So that **[I can achieve some goal]**

I want to **[do some task]**

FEATURE EXAMPLE USING NOTATION

Feature **Search Tutor by reputation**

As a **Student**

So that I can **find a qualified Tutor**

I want to **search Tutors by their reputation scores**

GOOD FEATURES

- SMART principles are one way to write good features
- **SMART** stands for:
  - **S**pecific
  - **M**easurable
  - **A**chievable
  - **R**elevant
  - **T**ime-Boxed

EXAMPLES OF GOOD VS BAD FEATURES (PART1)

- Specific

– Example of a vague story:

Student can search for a tutor

– Example of a specific story:

Student can search for a lecturer by their reputation score

EXAMPLES OF GOOD VS BAD FEATURES (PART2)

- Measurable
  - Example of a non-measurable story:

    Searching tutors should be fast.
  - Example of a measurable story:

    When searching for a tutor, the results should appear within 5 seconds.

EXAMPLES OF GOOD VS BAD FEATURES (PART3)

- Time-Boxed (time budget, may be complex, prioritize, reschedule, re-negotiate dead-lines, re-negotiate features, scale down)
- Achievable

– Realistic in terms of time assigned (can be implemented in one iteration. If not may need to decompose into smaller features)

EXAMPLES OF GOOD VS BAD FEATURES (PART4)

- Relevant:
  - resolving one of the problems in the problem statement
  - Adding business value
- Example:
  - Why searching for reputable tutors? To maximize quality of courses.

USER STORIES (REF1 &REF2)

- User story: feature + scenarios
- Stories: concept in HCI, paper cards, easy to understand and manipulate, used for brain storming, prioritizing
- Stories: few sentences written in non technical terms, yet ready to be used as input for various project tasks, focus on stakeholder perspective
- Will refine features in future phases of the project (e.g., use case diagrams, input/output interfaces, UI prototypes)

STORIES (CONTIN'D)

- Captures a feature rationale (not detailed requirements) n Business value (customer oriented, a feature that customer needs)
- Small (few person / month to deliver the feature)
- Independent functional requirements (no overlaps between stories)
- Testable by customer
- Effective project management (more accurate estimate of progress, iterative development cycle, manage priorities)

SCENARIO

- A scenario
  - Explain how a single feature is used.
  - consists of a set of (3 to 8) steps.
- Each step of a scenario starts with a keyword, i.e.,
  - GIVEN (pre-condition/state)
  - WHEN (Action/Event)
  - THEN (Transition / consequence of action)
  - AND (conjunction)

SCENARIO EXAMPLE

- Feature: student can search tutors by reputation score.
- Scenario: Search Tutors by reputation (cloud be used to generate UI interactions/mock-ups, state machine)

GIVEN I am on myMOOC home page

WHEN I click on “search for a Tutor”

THEN I should be on “search for a Tutor” page

WHEN I fill in “reputation” higher than” “0.9”

AND I press “Search” button

THEN I should see all tutors whose reputation score higher than 0.9

WRITING GOOD USER STORIES

- Write for broad audience: e.g., people who are not in your team. Will they understand?
- Complementary to other techniques, e.g., SMART user stories
- Reuse/analogy: e.g., search and analysis of similar services/products, reuse of feature descriptions (at least style) from previous projects.
- Avoid obscure language: e.g. technical jargon (unless internal project and agreed upon by team), mixing requirements with background/detailed design documentation

REFERENCES

- (Ref. 1) Scott Berkun. The art of project management. O'Reilly 2005
- (Ref. 2) Leszek A. Maciaszek: Requirements analysis and system design (2. ed.), Addison-Wesley, 2005
- (Ref. 3) Software Engineering. Ian Sommerville, Addison-Wesley, 9. ed., 2010,
- (Ref. 4) inGenius: A Crash Course on Creativity, by Tina Seelig

FIRST DELIVERABLE

- Describe few problem statements you are going to solve in your project
- Identify the features that you are going to implement to solve the problems
- Describe user stories
- Group effort.
- Meeting with mentor to refine ideas and get feedback.
- A detailed specification will be released soon.

Q&A