[CS350] 2018-Fall, Team Project

Project #2. Software Requirements Specification (SRS)

Open: 2018-09-21 (SEP. 21th, 2018)
Due: 2018-10-10 (OCT. 10th, 2018)

In Project 1, your team selected a topic and proposed some ideas and solutions. Also, some functional and non-functional requirements are identified. In Project 2, based on the elicited requirements, your team will refine and specify those requirements to articulate goals and scope of your project. Software Requirements Specification (SRS) will include (i) Title Page, (ii) Introduction, (iii) Overall Description, (iv) System Features, (v) Preliminary User Manual, (vi) Acceptance Criteria, (vii) Non-functional Requirements, (viii) Roles and Responsibilities, and (viiii) Acknowledgement.
(no more than **20 pages**)

1. Title Page

Include the name of the document, team name, team members, and date. You also prepare Table of Contents (including page numbers and so on) of your document.

2. Introduction

Identify the product whose software requirements are specified in this document. Describe the scope of the product that is covered by this SRS. A quick description of the project, to communicate how your 'product'(application) supports the need to users. The *vision statement* must articulate the goals of the product (or service): (i) the overall goal, (ii) strategies to lead your team to success, (iii) the most top-level requirements associated with the system goal.

3. Overall Description

A. Product Perspective

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of product family, a replacement for certain existing system, or a new, self-contained product. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

B. Product Features

Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Please highlight the features with the public data. Details will be provided in Section 4, so only a high-level summary is needed here. Organize the functions to make them understandable to any reader of the SRS.

C. Operating Environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

D. Assumption and Dependencies

List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change.

4. System Features

A. Functional Requirements

Specify functional requirements of your system. They could be defined hierarchically, and they should include specific actors and functional features. Also, the specified requirements should be prioritized based on your team's own criteria.

B. Use Case Diagram & Descriptions

Draw a use case diagram in which the product boundary (scope of the product) is clearly identified, and all the major use cases are shown.

Include actors relevant to your product, such as user, external system, DB etc. For each use case, give a priority for product development. For three main use cases, describe them based on the use case description template provided (See Appendix A and B).

- When you properly and correctly use *use-case relationships* such as <<include>> and <<extend>>, there will be extra credits.

C. Domain Model

In your domain model, make sure to include objects (5~15 objects) and relationships which are critical to your system domain. Build a domain model with UML class diagram (See Appendix C).

D. Sequence Diagrams

For three main use cases, which were selected for 4-B, show expected dynamic behavior (ordered interactions) of your product using a sequence diagram.

- If you use *sequence fragments* effectively, there will be extra credits.

*TAs will give a lecture about exercising UML diagrams (e.g., use case diagrams, sequence diagrams, class diagrams) on SEP 27th. The detailed time will be noticed in KLMS.

5. Preliminary User Manual

Identify who the user of the manual should be (State assumptions about the user). Give the user a good idea of the human interface. For a happy case scenario of your app, show some expected menus, command lines, interaction screens, and/or report formats.

- If you use some *images to describe preliminary GUI (Graphical User Interfaces),* there will be extra credits.

6. Acceptance Criteria

Based on the analysis of previous sections, describe how the user or evaluator can determine whether the end product actually performs the way it is described. This is best done as a *check list*.

It gives a way to determine whether the project has been completed and you have delivered what was promised. What tests should be performed to verify that it functions as described earlier? (Unless you have a complete understanding of the requirements of the software, this section is hard to write. Later, if specifications change, this section will have to be updated.)

7. Non-functional Requirements (Quality Attribute)

State important non-functional requirements. You may make reasonable assumptions about the scope of your project. You can refer to Appendix D for the detail about nonfunctional requirements.

8. Roles & Responsibilities

State roles and responsibilities for each member of your team. For this semester, you will develop your product based on the Agile development method which pursues fast and iterative software development. One of Agile development method is to have a small size of the team with detailed roles and responsibilities. One should be a team manager who is responsible to manage, promote, and accelerate your progress, and these are the main but not the only role of a manager. The manager should explain how you have managed your project, in the next report. The others are team members with specific roles and responsibilities. Please describe the detailed roles and responsibilities of your team manager and members. For example, you can divide roles by considering client part and DB part or you can assign exact objects or functionalities to each member.

That you describe here are roles and responsibilities for Sprint 1. The manager will be replaced by another member for Sprint 2, and if needed, you can also change the other's role.

Schedule: https://docs.google.com/spreadsheets/d/16F4I9BeGtmn9L4VuKetr_pN_U0lcg9BxYITeOEq98x4/edit#gid=141009320

9. Acknowledgement

Include authors of this document, by section, and editor of document.

[Scoring criteria]

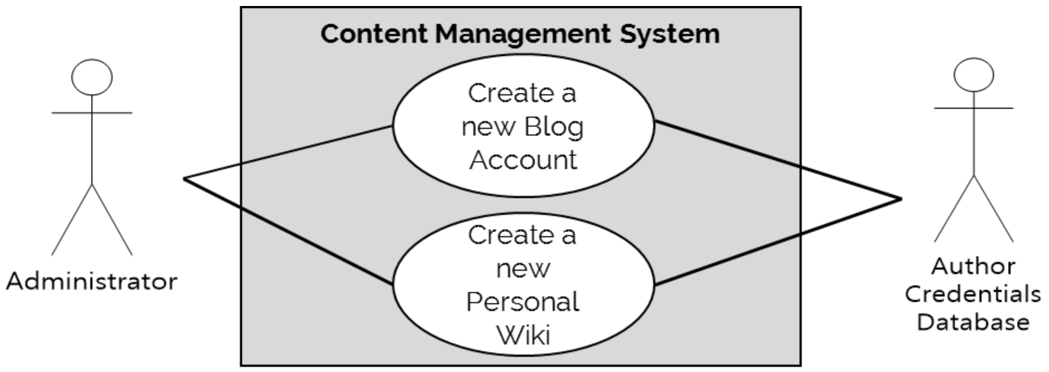
- **Completeness**: requirements, use-cases, use-case descriptions, sequence diagram
- **Consistency**: use-case diagram & use-case description & sequence diagram
- **Unambiguity of UML models**
- **Understanding of basic skills for UML modeling**

*After the submission, a *consulting session* will be held by TAs.
*Submit your report to *KLMS* (not by e-mail)

Appendix A. Use Case Description Template

Use case name	**Name of a use case to be described**
Related Requirements	Some indication as to which requirements this use case partially or completely fulfills.
Goal in Context	The use case’s place within the system and why this use case is important.
Preconditions	What needs to happen before the use case can be executed.
Successful End Condition	What the system’s condition should be if the use case executes successfully.
Failed End Condition	What the system’s condition should be if the use case fails to execute successfully.
Primary Actors	The main actors that participate in the use case. Often includes the actors that trigger or directly receive information from a use case’s execution.
Secondary Actors	Actors that participate but are not the main players in a use case’s execution.
Trigger	The event triggered by an actor that causes the use case to execute.
Main Flow	The place to describe each of the important steps in a use case’s normal execution.
Extensions	A description of any alternative steps from the ones described in the Main Flow.

Appendix B. Use Case Description Example
(<Create a new blog account> use case of Content Management System (CMS))


Content Management System
Create a new Blog Account
Create a new Personal Wiki
Administrator
Author Credentials Database
Use case diagram

Use case name	**Create a new blog account**	
Related Requirements	Requirement A.1.	
Goal in Context	A new or existing author requests a new blog account from the Administrator.	
Preconditions	The system is limited to recognized authors and so the author needs to have appropriate proof of identity.	
Successful End Condition	A new blog account is created for the author.	
Failed End Condition	The application for a new blog account is rejected.	
Primary Actors	Administrator.	
Secondary Actors	Author Credentials Database.	
Trigger	The Administrator asks the CMS to create a new blog account.	
Main Flow	**Step**	**Action**
	1	The Administrator asks the system to create a new blog account.
	2	The Administrator selects an account type.
	3	The Administrator enters the author's details.
	4	The author's details are verified using the Author Credentials Database.
	5	The new blog account is created.
	6	A summary of the new blog account's details are emailed to the author.
Extensions	**Step**	**Branching Action**
	4.1	The Author Credentials Database does not verify the author's details.
	4.2	The author's new blog account applications is rejected.

Appendix C. Domain Model & Examples

a) Domain model

A domain is a collection of related concepts, relationships, and workflows. A domain model is a package containing class and activity diagrams[1], and it provides a common vocabulary to enable clear communication among members of a project team or stakeholders[2].

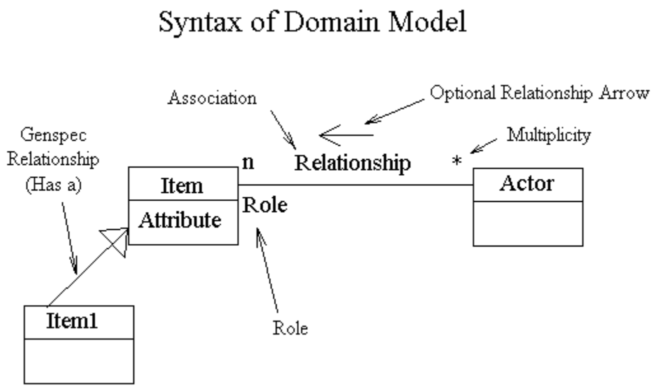
In the Unified Modeling Language (UML), a class diagram is used to represent the domain model. A UML domain model will relate objects in the system domain to each other. Objects in the domain model can be physical objects or abstract concepts[3]. Nouns can be taken from the requirements definitions and use case drawings. After the list of concept is complete, a domain model should be made by considering which simple items should be attributes of objects.

The domain model is a static model. Time flow, with sequence of events or information flow are not shown in the domain model (i.e., we have to avoid showing procedural relationships). This model does not include software, and the objects in the domain model are candidates for programming objects. There can be multiple relationships between objects in the domain model.

In the domain model, the following are shown[2]:

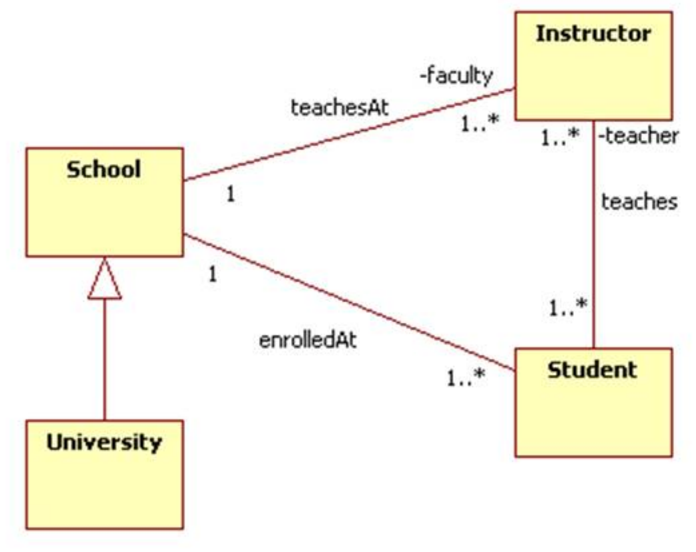
- Concepts (Objects)
- Attributes of Objects (Attributes must be simple attributes)
- Association between objects
- Multiplicity
- Optional direction of relationship arrow
- Optional role of object

Syntax of domain model:

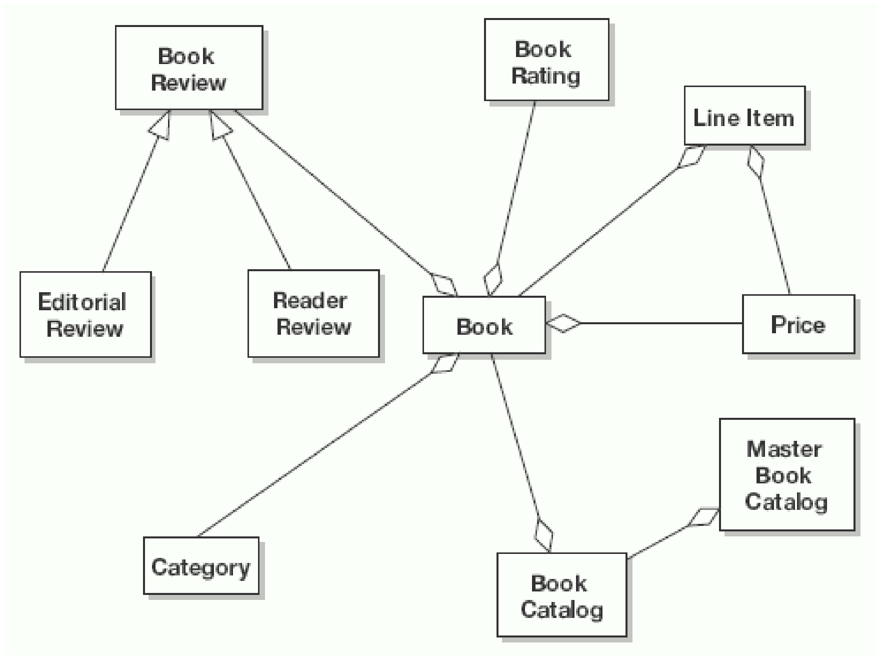


[1] http://www.cs.sjsu.edu/~pearce/modules/lectures/ooa/analysis/DomainModeling.htm
[2] https://iconixprocess.wordpress.com/iconix-process/requirements/domain-modeling/
[3] http://www.comptechdoc.org/independent/uml/begin/umldomainmodel.html

b) Domain model examples[4]



It contains the concepts of school, university, instructor, and student
as well as the relationships teaches-at, teaches, and enrolled-at



This is another example of a domain model diagram
to develop an online library (book store) management system

[4] http://www.cs.sjsu.edu/~pearce/modules/lectures/ooa/analysis/DomainModeling.htm

Appendix D. Non-functional Requirements

1. User Interface and Human Factors

- What type of user will be actually using the system?
- Will more than one type of user be using the system?
- What sort of training will be required for each type of user?
- Is it particularly important that the system be easy to learn?
- Is it particularly important that users be protected from making errors?
- What sort of input/output devices for human interface are available, and what are their characteristic?

2. Documentation

- What sorts of documentation are required?
- What audience is to be addressed by each document?

3. Hardware Considerations

- What hardware is the proposed system to be used on?
- What are the characteristics of the target hardware, including memory size and auxiliary storage space?

4. Performance Characteristics

- Are there any speed, throughput, or response time constraints on the system?
- Is there size of capacity constraints on the data to be processed by the system?

5. Error Handling and Extreme Conditions

- How should the system respond to input errors?
- How should the system respond to extreme conditions?

6. System Interfacing

- Is input coming from systems outside the proposed system?
- Is output to go to systems outside the proposed system?
- Are there restrictions on the format or medium that must be used for input or output?

7. Quality Issues

- What are the requirements for reliability?
- Must the system trap faults?
- Is there a maximum acceptable time for restarting the system after a failure?
- What is the acceptable system downtime per 24-hour period?
- Is it important that the system be portable (able to move to different hardware or operating system environments) ?

8. System Modifications

- What parts of the system are likely candidates for later modification?
- What sorts of modifications are expected?

9. Physical Environment

- Where is the target equipment to operate?
- Will the target equipment be in one or several locations?
- Will the environmental conditions in any way be out of the ordinary (for example, unusual temperatures, vibrations, magnetic influences, and so on)?

10. Security Issues

- Must access to any data or the system itself be controlled?
- How often will the system be backed up?
- Who will be responsible for back up?
- Is physical security an issue?

11. Resources and Management Issues

- What materials, personnel, computer time, and other resources will be required to build, install, and maintain the system?
- What skills or knowledge must the developers have to develop the system?
- What are the proposed intermediate and final deadlines for system development costs?
- What is the proposed budget for hardware, personnel, and other development costs?
- Who is responsible for system installation?
- Who will be responsible for system maintenance?