CS 350 Task 3: Project Software Requirements Specification Elicitation

This task lays out the framework for a proposed Software Requirements Specification (SRS) related to our project. It relies on the breadth and depth of understanding developed in Tasks 1 and 2 to establish and bidirectionally cross-reference the following hierarchy:

- Scenarios
- User Stories
- Questions
- Requirements
- Specifications
- Requirement Verification
- Specification Verification
- ~~Validation~~

Lectures 16 onward cover them in more detail. Note that the final component, validation, is not in play because we have no good definition of what the customer's needs are. We (intentionally) glossed over this step early in the development process. Now we would be in trouble. Think about the first panel in The Cartoon.

The color coding highlights the cross-references here for clarity, but it is not part of the solution.

Part 1: Scenarios

A scenario is defined for this task as a general end-to-end story narrating one complete action, as done on the board in Lecture 17 for a previous project on rockets.

Given our scope from Tasks 1 and 2 and lecture, select two of the following scenarios, then title three of your own, all numbered 1 through 5, and finally narrate the reasonably complete process of performing a corresponding story action. Do not use any from lecture as yours.

- Climbing to an altitude.
- Turning to a direction.
- Landing.

Form of Solution

<scenario_num>: <scenario_title>

<narrative>

Example

1: Dropping water balloons from a tower

The holding mechanism at the top of the tower releases the balloon. The balloon falls straight down under gravity. It impacts the ground.

Part 2: User Stories

A user story is a specific part of a scenario from the user's perspective in the form:

As an <actor> I want to do <action> so that <achievement>.

For each of your scenarios from Part 1, title and state a user story, numbered 1.

Form of Solution

<scenario_num>.<user_story_num>: <user_story_title>

<narrative>

Example

1.1: Aiming at friend

As a friend I want to drop a water balloon on another friend so that he gets wet.

Part 3: Questions

A question elicits a targeted response to an aspect that needs further clarification.

For each of the user stories from Part 2, state and justify two representative W5H questions, numbered 1 through 2, that would be important to clarify.

Form of Solution

<scenario_num>.<user_story_num>.<question_num>: <question>

<justification why>

Example

1.1.1: How is the water balloon secured?

Holding onto the water balloon properly will ensure that it is dropped at the right time.

Part 4: Requirements

A requirement states the form of the solution that is expected. It does not answer a question.

For each of the questions from Part 3, state and justify two representative requirements, numbered 1 through 2, that are important to satisfy.

Form of Solution

<scenario_num>.<user_story_num>.<question_num>.<requirement_num>: <requirement>

<justification why>

Example

1.1.1.1: Balloon shall contain water

It is a water balloon.

Part 5: Specifications

A specification constrains the form or scope of the solution for a requirement.

For each requirement from Part 4, state and justify two representative specifications, numbered 1 through 2.

Form of Solution

<scenario_num>.<user_story_num>.<question_num>.<requirement_num>.<specification_num>: <specification>

<justification why>

Example

1.1.1.1.1: Balloon shall contain at least one liter of water.

This amount produces the best effect.

Part 6: Requirement Verification

Requirement verification defines how to determine whether a requirement has been satisfied; i.e., did we build the product right?

For each requirement from Part 4, briefly describe a go/no-go check, numbered 1.

Form of Solution

<scenario_num>.<user_story_num>.<question_num>.<requirement_num>.A.<verification_num>: <description>

Example

1.1.1.1.A.1: Is water present in the balloon?

Part 7: Specification Verification

Specification verification defines how to determine whether a specification has been satisfied; i.e., did we build the product right?

For each specification from Part 5, briefly describe a go/no-go check, numbered 1.

Form of Solution

<scenario_num>.<user_story_num>.<question_num>.<requirement_num>.<specification_num>.<verification_num>: <description>

Example

1.1.1.1.1.1.1: Is at least one liter of water present in the balloon?

Part 8: Validation

Validation defines how to determine whether the solution satisfies the customer's needs; i.e., did we build the right product?

No action is required for this part because we cannot do it without having formally defined the customer's needs. Therefore, we have no disciplined way to determine whether we have satisfied them.

Deliverable

Submit one PDF document with each part in a separately named section in order as indicated below (presented in columns here only to save space). Do not nest the sections or use columns.

Part 1: Scenarios

1 blah blah
2 blah blah
3 blah blah
4 blah blah
5 blah blah

Part 2: User Stories

1.1 blah blah
2.1 blah blah
3.1 blah blah
4.1 blah blah
5.1 blah blah

Part 3: Questions

1.1.1 blah blah
1.1.2 blah blah

2.1.1 blah blah
2.1.2 blah blah

3.1.1 blah blah
3.1.2 blah blah

4.1.1 blah blah
4.1.2 blah blah

5.1.1 blah blah
5.1.2 blah blah

Part 4: Requirements

1.1.1.1 blah blah
1.1.1.2 blah blah
1.1.2.1 blah blah
1.1.2.2 blah blah

2.1.1.1 blah blah
2.1.1.2 blah blah
2.1.2.1 blah blah
2.1.2.2 blah blah

3.1.1.1 blah blah
3.1.1.2 blah blah
3.1.2.1 blah blah
3.1.2.2 blah blah

4.1.1.1 blah blah
4.1.1.2 blah blah
4.1.2.1 blah blah
4.1.2.2 blah blah

5.1.1.1 blah blah
5.1.1.2 blah blah
5.1.2.1 blah blah
5.1.2.2 blah blah

Part 5: Specifications

1.1.1.1.1 blah blah
1.1.1.1.2 blah blah
1.1.1.2.1 blah blah
1.1.1.2.2 blah blah
1.1.2.1.1 blah blah
1.1.2.1.2 blah blah
1.1.2.2.1 blah blah
1.1.2.2.2 blah blah

2.1.1.1.1 blah blah
2.1.1.1.2 blah blah
2.1.1.2.1 blah blah
2.1.1.2.2 blah blah
2.1.2.1.1 blah blah
2.1.2.1.2 blah blah
2.1.2.2.1 blah blah
2.1.2.2.2 blah blah

3.1.1.1.1 blah blah
3.1.1.1.2 blah blah
3.1.1.2.1 blah blah
3.1.1.2.2 blah blah
3.1.2.1.1 blah blah
3.1.2.1.2 blah blah
3.1.2.2.1 blah blah
3.1.2.2.2 blah blah

4.1.1.1.1 blah blah
4.1.1.1.2 blah blah
4.1.1.2.1 blah blah
4.1.1.2.2 blah blah
4.1.2.1.1 blah blah
4.1.2.1.2 blah blah
4.1.2.2.1 blah blah
4.1.2.2.2 blah blah

5.1.1.1.1 blah blah
5.1.1.1.2 blah blah
5.1.1.2.1 blah blah
5.1.1.2.2 blah blah
5.1.2.1.1 blah blah
5.1.2.1.2 blah blah
5.1.2.2.1 blah blah
5.1.2.2.2 blah blah

Part 6: Requirement Verification

1.1.1.1.A.1 blah blah
1.1.1.2.A.1 blah blah
1.1.2.1.A.1 blah blah
1.1.2.2.A.1 blah blah

2.1.1.1.A.1 blah blah
2.1.1.2.A.1 blah blah
2.1.2.1.A.1 blah blah
2.1.2.2.A.1 blah blah

3.1.1.1.A.1 blah blah
3.1.1.2.A.1 blah blah
3.1.2.1.A.1 blah blah
3.1.2.2.A.1 blah blah

4.1.1.1.A.1 blah blah
4.1.1.2.A.1 blah blah
4.1.2.1.A.1 blah blah
4.1.2.2.A.1 blah blah

5.1.1.1.A.1 blah blah
5.1.1.2.A.1 blah blah
5.1.2.1.A.1 blah blah
5.1.2.2.A.1 blah blah

Part 7: Specification Verification

1.1.1.1.1.1 blah blah
1.1.1.1.2.1 blah blah
1.1.1.2.1.1 blah blah
1.1.1.2.2.1 blah blah
1.1.2.1.1.1 blah blah
1.1.2.1.2.1 blah blah
1.1.2.2.1.1 blah blah
1.1.2.2.2.1 blah blah

2.1.1.1.1.1 blah blah
2.1.1.1.2.1 blah blah
2.1.1.2.1.1 blah blah
2.1.1.2.2.1 blah blah
2.1.2.1.1.1 blah blah
2.1.2.1.2.1 blah blah
2.1.2.2.1.1 blah blah
2.1.2.2.2.1 blah blah

3.1.1.1.1.1 blah blah
3.1.1.1.2.1 blah blah
3.1.1.2.1.1 blah blah
3.1.1.2.2.1 blah blah
3.1.2.1.1.1 blah blah
3.1.2.1.2.1 blah blah
3.1.2.2.1.1 blah blah
3.1.2.2.2.1 blah blah

4.1.1.1.1.1 blah blah
4.1.1.1.2.1 blah blah
4.1.1.2.1.1 blah blah
4.1.1.2.2.1 blah blah
4.1.2.1.1.1 blah blah
4.1.2.1.2.1 blah blah
4.1.2.2.1.1 blah blah
4.1.2.2.2.1 blah blah

5.1.1.1.1.1 blah blah
5.1.1.1.2.1 blah blah
5.1.1.2.1.1 blah blah
5.1.1.2.2.1 blah blah
5.1.2.1.1.1 blah blah
5.1.2.1.2.1 blah blah
5.1.2.2.1.1 blah blah
5.1.2.2.2.1 blah blah

If the document were nested (simply by sorting the indices), it would produce this embedded structure. However, we do not normally want this form because it confounds the context of what we are looking at. For example, the Requirements section should be exclusively about requirements: nothing less, nothing more. If we need to understand where a requirement comes from or where it goes, then the cross-referencing leads us to that independent section.

1 blah blah	2 blah blah	3 blah blah	4 blah blah	5 blah blah
1.1 blah blah	2.1 blah blah	3.1 blah blah	4.1 blah blah	5.1 blah blah
1.1.1 blah blah	2.1.1 blah blah	3.1.1 blah blah	4.1.1 blah blah	5.1.1 blah blah
1.1.1.1 blah blah	2.1.1.1 blah blah	3.1.1.1 blah blah	4.1.1.1 blah blah	5.1.1.1 blah blah
1.1.1.1.1 blah blah	2.1.1.1.1 blah blah	3.1.1.1.1 blah blah	4.1.1.1.1 blah blah	5.1.1.1.1 blah blah
1.1.1.1.1.1 blah blah	2.1.1.1.1.1 blah blah	3.1.1.1.1.1 blah blah	4.1.1.1.1.1 blah blah	5.1.1.1.1.1 blah blah
1.1.1.1.2 blah blah	2.1.1.1.2 blah blah	3.1.1.1.2 blah blah	4.1.1.1.2 blah blah	5.1.1.1.2 blah blah
1.1.1.1.2.1 blah blah	2.1.1.1.2.1 blah blah	3.1.1.1.2.1 blah blah	4.1.1.1.2.1 blah blah	5.1.1.1.2.1 blah blah
1.1.1.1.A.1 blah blah	2.1.1.1.A.1 blah blah	3.1.1.1.A.1 blah blah	4.1.1.1.A.1 blah blah	5.1.1.1.A.1 blah blah
1.1.1.2 blah blah	2.1.1.2 blah blah	3.1.1.2 blah blah	4.1.1.2 blah blah	5.1.1.2 blah blah
1.1.1.2.1 blah blah	2.1.1.2.1 blah blah	3.1.1.2.1 blah blah	4.1.1.2.1 blah blah	5.1.1.2.1 blah blah
1.1.1.2.1.1 blah blah	2.1.1.2.1.1 blah blah	3.1.1.2.1.1 blah blah	4.1.1.2.1.1 blah blah	5.1.1.2.1.1 blah blah
1.1.1.2.2 blah blah	2.1.1.2.2 blah blah	3.1.1.2.2 blah blah	4.1.1.2.2 blah blah	5.1.1.2.2 blah blah
1.1.1.2.2.1 blah blah	2.1.1.2.2.1 blah blah	3.1.1.2.2.1 blah blah	4.1.1.2.2.1 blah blah	5.1.1.2.2.1 blah blah
1.1.1.2.A.1 blah blah	2.1.1.2.A.1 blah blah	3.1.1.2.A.1 blah blah	4.1.1.2.A.1 blah blah	5.1.1.2.A.1 blah blah
1.1.2 blah blah	2.1.2 blah blah	3.1.2 blah blah	4.1.2 blah blah	5.1.2 blah blah
1.1.2.1 blah blah	2.1.2.1 blah blah	3.1.2.1 blah blah	4.1.2.1 blah blah	5.1.2.1 blah blah
1.1.2.1.1 blah blah	2.1.2.1.1 blah blah	3.1.2.1.1 blah blah	4.1.2.1.1 blah blah	5.1.2.1.1 blah blah
1.1.2.1.1.1 blah blah	2.1.2.1.1.1 blah blah	3.1.2.1.1.1 blah blah	4.1.2.1.1.1 blah blah	5.1.2.1.1.1 blah blah
1.1.2.1.2 blah blah	2.1.2.1.2 blah blah	3.1.2.1.2 blah blah	4.1.2.1.2 blah blah	5.1.2.1.2 blah blah
1.1.2.1.2.1 blah blah	2.1.2.1.2.1 blah blah	3.1.2.1.2.1 blah blah	4.1.2.1.2.1 blah blah	5.1.2.1.2.1 blah blah
1.1.2.1.A.1 blah blah	2.1.2.1.A.1 blah blah	3.1.2.1.A.1 blah blah	4.1.2.1.A.1 blah blah	5.1.2.1.A.1 blah blah
1.1.2.2 blah blah	2.1.2.2 blah blah	3.1.2.2 blah blah	4.1.2.2 blah blah	5.1.2.2 blah blah
1.1.2.2.1 blah blah	2.1.2.2.1 blah blah	3.1.2.2.1 blah blah	4.1.2.2.1 blah blah	5.1.2.2.1 blah blah
1.1.2.2.1.1 blah blah	2.1.2.2.1.1 blah blah	3.1.2.2.1.1 blah blah	4.1.2.2.1.1 blah blah	5.1.2.2.1.1 blah blah
1.1.2.2.2 blah blah	2.1.2.2.2 blah blah	3.1.2.2.2 blah blah	4.1.2.2.2 blah blah	5.1.2.2.2 blah blah
1.1.2.2.2.1 blah blah	2.1.2.2.2.1 blah blah	3.1.2.2.2.1 blah blah	4.1.2.2.2.1 blah blah	5.1.2.2.2.1 blah blah
1.1.2.2.A.1 blah blah	2.1.2.2.A.1 blah blah	3.1.2.2.A.1 blah blah	4.1.2.2.A.1 blah blah	5.1.2.2.A.1 blah blah

Assessment

Each part and its contents will be evaluated with respect to the rest of the document. Your responses need only be reasonable and consistent, not 100% correct or optimal. You are not qualified in the subject matter to make such decisions. However, keep in mind that in industry, you would also not be qualified, but you would be expected to get everything right. Think about the roles of Task 1 in establishing the customer's needs and Task 2 in bridging the gap between knowing nothing and knowing something, but not enough. What would you do further?

Follow our process: read ➜ understand ➜ plan ➜ execute ➜ verify ➜ reflect