Requirements Engineering Diagnostic and Process Improvement

Dr. Naoufel Boulila (naoufel.boulila@siemens.com)

Software Requirements Specification Template Documents for Requirements Engineers, Business Analysts, Product Managers, and Developers

Software Requirements Specification Template

Table of Contents

Requirements Engineering Diagnostic and Process Improvement 1

1.0 Introduction 4
1.1 Purpose 4
1.2 Scope 4
1.3 References 4
1.4 Overview of Document 4

2.0 System Overview 4
2.1 Project Perspective 5
2.2 System Context 5
2.3 General Constraints 5
2.4 Assumptions and Dependencies 5

3.0 Domain Model 5
3.1 Class Diagrams 5
3.2 Class Specifications 6

4.0 Prototyping 6

5.0 Requirements 6
5.1 Use Case Requirements 6
5.1.1 Actor List 7
5.1.2 Use Case Diagrams 7
5.1.3 Use Case Specifications 8
5.1.3.1 Use Case Standard Template 8
5.1.3.2 Essential Use Case Specification Template I 9
5.1.3.3 Extended Use Case Specification Template II 9
5.1.4 Extended Use Case Specification Template III 10
5.2 Activity diagrams 11
5.3 Business Use case 12
5.4 Business Rules 12
5.5 Non-Functional Requirements 12
5.5.1 System Requirements 12
5.5.2 Usability Requirements 13
5.5.3 Performance Requirements 13
5.5.4 Security Requirements 13
5.5.5 Delivery Requirements 13

5.5.6 Legal Requirements....13
5.5.7 Interoperability Requirements....13
5.5.8 Scalability Requirements....13
5.6 Interface Requirements....14
5.6.1 Machine Interfaces....14
5.6.2 External System Interfaces....14
5.6.3 Human-Computer Interface Considerations....14
5.6.4 Input and Output Requirements....14
6.0 Project/Release Issues....14
6.1 Projected Development Effort....14
6.2 Proposed Project Schedule....15
6.3 Conversion / Load Requirements....15
6.3.1 Data Population Load....15
6.3.2 Reference tables and Baseline Data Load....15
6.3.3 Data Conversion Strategy....15
6.3.4 Data Conversion Assumptions and Constraints....15
7.0 Appendices....15

1.0 Introduction

The Introduction section provides an overview of the Requirements Specifications and the scope of the system.

1.1 Purpose

- Define the purpose of the Requirements Specification document and identify who should use this document.
- This document describes the software requirements of a given system, that is, it describes the what, not the how, of the capabilities of the system.
- This document also serves as the basis for the design and implementation of the system, which can be documented in the corresponding document such as the Software Design Description (SDD) document.

1.2 Scope

- Identify the software artefacts to be produced (including delivered software).
- Explain what the current system will and will not do.
- Describe relevant benefits, objectives, and goals accurately.
- The description of scope should be consistent with other documents that may refer to this document (e.g. project plan).

1.3 References

List any documents referenced to create this Requirements Specification document, e.g.:

- Project Plan
- Documentation of workshops outcomes and decisions
- Change requests

Include the version number of each document and eventually the URL of any document repositories

1.4 Overview of Document

The overview provides a summary of the contents of each section of this document.

2.0 System Overview

The System Overview section presents the system context and design, and also discusses the background of the project

2.1 Project Perspective

The Project Perspective describes the context and origin of the system by defining whether the system is:

- a subsequent member of a system family
- a replacement for an existing system(s)
- a new self-contained system

2.2 System Context

The System Context describes the resulting software within the business case, including strategic issues in which the system is involved or which it specifically addresses. This section must provide a clear context for the system, in particular for the stakeholders who may not have knowledge about the system.

2.3 General Constraints

General Constraints identify any business or system constraints that will impact the way in which the software is to be:

- specified
- designed
- implemented
- tested

2.4 Assumptions and Dependencies

List any assumptions that have been made during the start of the project. In addition, list any dependencies that may impact its success or the anticipated result

3.0 Domain Model

The Domain Model section includes Class Diagrams and Class Specifications. A Domain Model includes both graphical (e.g. UML Models) and written (textual) descriptions of objects within the problem domain or the software application. Domain Models also describe how the classes are structurally related to one another.

3.1 Class Diagrams

Class Diagrams use classes and associations to describe the static structure of a system (e.g. UML Classes).

Classes represent abstractions of objects with common characteristics, and may be definitions of software classes and not necessary real-world concepts. That is, they can model domain concepts or software classes.

Associations represent the structural relationships between classes with named, including multiplicities.

3.2 Class Specifications

Class Specifications, or Definitions, define and describe each class in a textual way.

4.0 Prototyping

Prototyping (also known as Proof of Concept) is a process designed to reduce risk by validating or deriving the final system requirements. Prototypes are developed from an internal specification, delivered to the customer for experimentation and can evolve to a final system or simply are discarded such as in Throw-Away Prototyping.

Throw-Away Prototyping begins with the requirements that are unclear or poorly understood.

The focus of the Throw Away Prototyping may vary from project to project, depending on which aspect requires validation. Examples are:

- How will poorly understood business functions be implemented in the system
- How will complex business rules be supported by the system

The Throw-Away Prototyping section states whether or not a prototype is required, and if so:

- what business functions it is to demonstrate
- what look and feel standards it is to follow
- how the user will interacts with the prototype system
- where the prototype will be deployed

For the last point, it is expected that the prototype will deployed and demonstrated on the stakeholder (e.g. customer) hardware and software.

5.0 Requirements

The Requirements section defines the Functional and Non-Functional Requirements of the proposed system.

5.1 Use Case Requirements

Specify each individual functional requirement that must be supported by the system. It is expected that these requirements have been processed as following:

- Gathered
- Analyzed
- Abstracted
- Categorized
- Prioritized

This means that, where appropriate, the gathered functional requirements should have been abstracted to a higher level and categorized. Conflicted

functional requirements should have been negotiated between different stakeholders, converging to one consensus that is acceptable to all.

5.1.1 Actor List

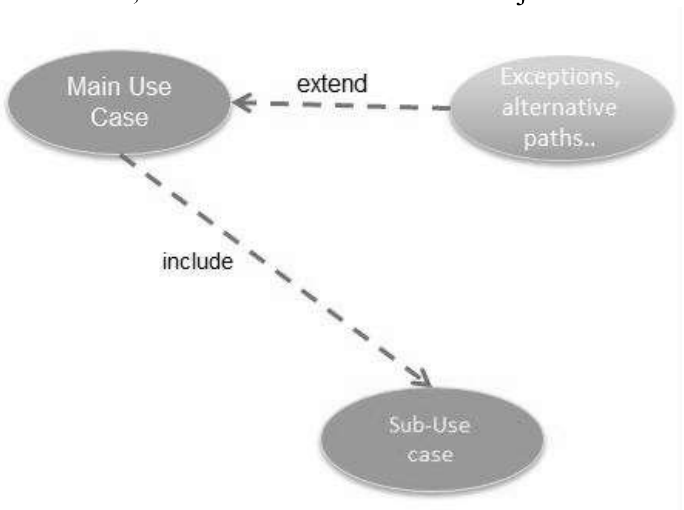
- An Actor is anything with behaviour; examples are system, a company, an organization, or a role played by a person.
- If the actor is a role played by a person (not the person's current job title) the Actor name should describe a category of persons that can play that given role.
- During requirements (resp. Use Cases) analysis, non-human actors (i.e. external automated systems) may be identified. An example is a credit card payment system; these non-human actors still need to be documented in the Actor List.
- For each Actor, describe briefly its responsibilities in relation to the system.

5.1.2 Use Case Diagrams

Use Case Diagrams shows actors (i.e. user roles) and Use Cases. They also describe how the actors interact with the system and the relationships between Use Cases.

Good practices consist of using "include" and "extend" relationships in using cases.

- **Include**: Use Cases may contain the functionality of another Use Case as part of their normal processing
- **Extend**: One Use Case may be used to extend the behavior of another, similar to UML Classes and Object



The deployment of Use Case diagrams or not, during the developments, is determined according to several criteria including the type of change occurring, as summarized in the following table:

Type of Change	When to define use case	Elements concerned with change
Minimal change	• Use cases are not needed • change has to be documented	• Layout • UI • DB
Medium change	• individual judgement is needed	• Layout • UI • DB
Big change	• New Use cases are needed • Update existing use cases	• UI • Application • DB

For more details see document "when to define use cases".

5.1.3 Use Case Specifications

Every Use Case must have a corresponding textual specification, helping the reader to focus on what is essential in terms of functional requirements. The specification:

- describes the Use Case
- lists the actors that directly participate
- Includes the steps that are involved in the individual Use Case.

The specification focuses on functional requirements, free of design details such as graphical user interface behaviour, or implementation details. There might be references to data elements, but these should be business-oriented names instead of database table or column names.

The resulting specifications should also be included in the Requirements Specification document.

5.1.3.1 Use Case Standard Template

We define three types of Use Cases templates according to the roles within the organization:

Template	Who	Purpose
Essential Use Case Specification Template I	Business Analyst	• Provide high level description of the offered services • Serves as communication medium with the stakeholders
Extended Use Case Specification Template II	Developer	• Further system-oriented details that presents initial details for the implementation • Helps the transition to the sequence

		diagrams describing detailed system-user interaction
Extended Use Case Specification Template III	Developer	It is an extension of the Template II, and can be adapted according to the needs. Usually it is used for specific purposes in complex developments including a strong dependency on specific factors including hardware, specific commercials software, technology, performance etc.

5.1.3.2 Essential Use Case Specification Template I

Use Case: short active verb	
Use Case ID	Unique identifier
Requirement ID	The requirement to which the use case is related to
Actor(s)	One or many
Description	Brief description and flow of events

5.1.3.3 Extended Use Case Specification Template II

Use Case ID	**Use Case Name : short active verb**		
Requirement ID	The requirement to which the use case is related to		
Description	Goal description		
Rationale	The why of the existence of this use case		
Precondition	What must be true before starting the use case		
Post-condition	What must be true after accomplishing the use case		
Priority	How critical it is to the system, i.e., how important to implement in the first place		
Actor	Main actor role name		
Trigger	The action that leads to start the use case		
Main flow of events	Step	**User action**	**System response**
	1		

	2		
	…	…	
Exceptions	All other scenarios, branches, exception		

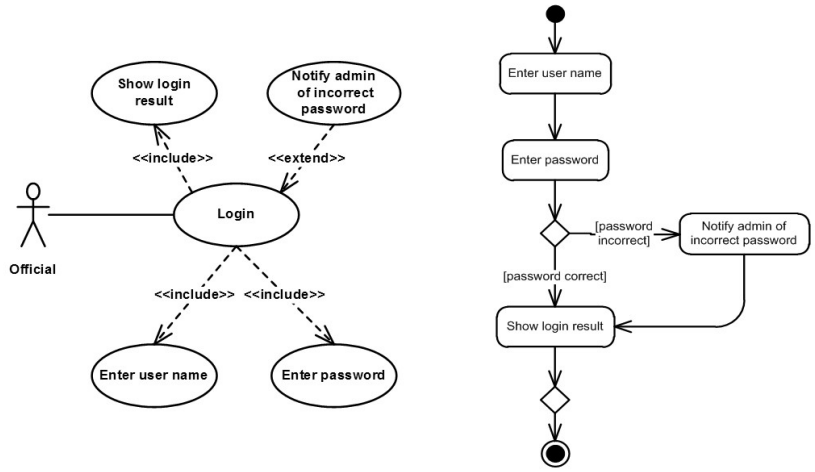
5.1.4 Extended Use Case Specification Template III

Use case name	**Use Case Name : short active verb**
Use Case ID	Unique identifier for the use case
Requirement ID	The requirement to which the use case is related to
Test ID(s)	Related test cases
Goal	Brief description of the purpose of the use case
Primary actor(s)	Names of the primary actor or actors who interact with the system
Brief description	Description of the use case
Preconditions	State of the system when the use case is triggered
Trigger	The action that leads to start the use case
Flow of events	Activities and interactions performed when the use case is performed
Post-conditions	State in which the use case leaves the system in
Priority	Relative development priority of the use case
Alternate flows and exceptions	Major alternatives or exceptions that may occur within the flow of events
Non-Functional requirement	Requirements such as performance, security, reliability, etc.
Assumptions	Any assumption that have been mandated, made
Issues	Open issues
Source	Meeting, interview, document, etc. the use case derives from
Technology and variation list	Input devices, methods, …
Frequency	How often it is expected to happen

Cost estimation	The estimated cost for implementing the use case in terms of manpower (hour, week, month-man)

5.2 Activity diagrams

Activity diagram: (UML term) is a diagram that describes a sequence of business or system activities. It is used to describe the dynamic behaviour of the system; i.e. what the system does and how it does it. Each Use Case can be detailed and described using activity diagrams (see next figure).



An activity diagram including partitions (swimlanes) also describes which object is responsible for each activity.

Activity Diagrams shows:

- sequential logic
- iteration
- decision making
- object states
- parallel threads
- an overview of threads as described by the sequence diagram

Diagram Type	Who	Purpose
Activity Diagram	Business Analyst	• business use case description or modeling • rough system behaviour description • workflows definition • processes description • software processes description • complex business rule • single use case description

		(modeling the logic captured by a single or usage scenario)
	Developer	• detail system use cases • single use case description (modeling the logic captured by a single or usage scenario)

5.3 Business Use case

A business Use Case is a business process representing a specific workflow in the business. It is an interaction that a stakeholder has with the business that achieves a business goal.

5.4 Business Rules

OCL (Object Constraint Language) might be used for documenting complex business rules.
Projects without complex business rules may instead document business rules in the notes or comments section of the relevant Use Case

5.5 Non-Functional Requirements

- The non-functional requirements for a system are typically constraints on the functional requirements – that is, not what the system does, but how it does it (e.g. how quickly, how efficiently, how easily a task is achieved from the user's perspective, etc.).

- Other non-functional requirements may be required characteristics that are not part of the system's functionality, e.g., conformance with legal requirements, scalability, interoperability, etc.

- Specify each individual non-functional requirement that must be supported by the system.

5.5.1 System Requirements

Provide a comprehensive but high-level description of the technologies, if known, that will compose the target system environment. The intent is not to restrict the developer's options, but rather to avoid implementation-dependency issues at delivery time. System requirements include:

- applicable system standards
- required operating systems
- required commercial software
- hardware or platform requirements
- performance requirements,
- any environmental requirements

5.5.2 Usability Requirements

Describe the expectations regarding to how easy the system will be to use. This includes considerations such as the educational level, experience and technical expertise of the target user community.

5.5.3 Performance Requirements

Describe the requirements for system performance, in terms of the following:

Speed	Specify the time available to complete specified actions. These are sometimes referred to as response times.
Safety	Quantify any perceived risk of possible damage to people, property and environment.
Precision	Quantify desired accuracy of results produced by the system.
Reliability, Availability	Quantify the allowable time between failures, and the allowable down time of the system. The down time may be needed for back-up, etceteras.
Capacity	Quantify the volume that the application can handle and the amount of data that can be stored.

5.5.4 Security Requirements

Specify the requirements for protecting the system from accidental or malicious access, use, modification, destruction or disclosure. Requirements for data integrity and confidentiality should also be specified in this section.

5.5.5 Delivery Requirements

Provide details of the life cycle and approach that will be used to deliver the system. Refer to the corresponding delivery standards document if any.

5.5.6 Legal Requirements

Define any legal requirements that the system must support, explain whether the system falls under the jurisdiction of any law. This includes intellectual property rights and any standards with which the system must comply.

5.5.7 Interoperability Requirements

Define any requirements relating to interoperability with relevant systems.

5.5.8 Scalability Requirements

Define any requirements relating to scalability.
E.g. .cross platform- independent … or ... has the ability to add business areas as required…

5.6 Interface Requirements

5.6.1 Machine Interfaces

Describe any interfaces to other machines, computers or devices

For example: System1 needs to access the System2 reports server to …..

5.6.2 External System Interfaces

Describe any interfaces to other systems, products or networks.

For example: … needs to access the System3 to reference the spatial information for ..

5.6.3 Human-Computer Interface Considerations

Provide screen images and associated text, note and describe any principles concerning the Human-Computer Interface, such as:

- screen layout
- use of UI elements such as drop-downs lists, check boxes, radio buttons, etc.
- help context
- error handing
- navigation

5.6.4 Input and Output Requirements

Define the inputs and outputs of the system and or business process. The definitions include the source, medium and type as well as any enablers or guides that help with the process.

6.0 Project/Release Issues

6.1 Projected Development Effort

Provide a high-level description and estimate of the project development efforts. This estimate includes the estimated effort required to complete the following phases:

- Design
- Build
- Test,
- Implementation.

These estimates are based upon the requirements as specified in this document.

6.2 Proposed Project Schedule

Documents the high-level project schedule, which is based upon the effort described in the previous section.

Project development that span more than 9 –12 months, are probably confronted to various changes such as requirements, technology changes which will negatively impact the project schedule

6.3 Conversion / Load Requirements

This section provides a high-level description of the conversion and load requirements for the system.

6.3.1 Data Population Load

The data population strategy is to be specified in this section. Applications that are being developed to replace a current system are likely to have complex data conversion requirements.

Applications for new lines of business likely will not have any data conversion requirements but will have a requirement to populate reference tables and some baseline data.

6.3.2 Reference tables and Baseline Data Load

Describe the strategy for populating reference tables and baseline data.

6.3.3 Data Conversion Strategy

Describe the strategy for populating the application with data. If data is being converted from a legacy system, describe the requirements and approach to be followed.

6.3.4 Data Conversion Assumptions and Constraints

Identify any assumptions or constraints that may limit or constrain the data conversion requirements.

7.0 Appendices

Provide any additional information or documents that might be useful during the System Development Life Cycle.