

Universidad Carlos III

Software Engineering 2022-23

Final Project   
Course 2022-23

Title

Date: **xx/xx/xx** - Version: **1/2/FINAL**

GROUP: **81/82/83** TEAM: **01/02…**

Members: **First Name/Last Name**

Document Status

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 0 | 30-09-2011 | Ana García | First Draft |
| 1 | 15-10-2011 | Juan Gómez | First deadline |
| 1.1 | 20-10-2011 | Ana García | Added fixes |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Time effort

|  |  |  |  |
| --- | --- | --- | --- |
| Name | INDIVIDUAL | TEAM | TOTAL |
| Ana García | 40 | 15 | **55** |
| Juan Gómez | 43 | 11 | **54** |
| Isabel López | 47 | 16 | **63** |
| Pedro Fernández | 50 | 18 | **68** |
| TOTAL | **180** | **60** | **240** |

Roles and responsibilities

|  |  |
| --- | --- |
| NAME | MAIN CONTRIBUTIONS |
| Ana García |  |
| Juan Gómez |  |
| Isabel López |  |
| Pedro Fernández |  |

It is also possible to use a RACI matrix: *responsible, accountable, consulted, informed*.

TABLE OF CONTENTS

1 introduction 5

1.1 Purpose of this document 5

1.2 How to use this document 5

1.3 References 5

2 dESCRIPTION 6

2.1 Product perspective 6

2.2 Scope of the product 6

2.3 Business case for the product 6

2.4 Restrictions 7

2.5 Types of users: roles and capabilities 7

2.6 Operational environment 7

3 REQUIREMENTS 8

3.1 Requirements classification 8

3.2 Template for requirements 8

3.3 Functional Requirements 8

3.4 Non-functional Requirements 9

3.5 Domain vocabulary 9

4 ARCHITECTURE 10

4.1 Conceptual model 10

4.2 Architectural model 10

4.3 Component specification 11

5 lists and tables 13

5.1 Requirements’ consistency matrix: overlapping, conflicts, etc. 13

5.2 Requirements traceability – Conceptual model (classes) 13

5.3 Requirements traceability – Development view (components) 13

REMARKS

* + The **first delivery** shall cover chapters 1-2-3 and a first version of the section 5.1
  + The **second delivery** shall cover chapters 4-5. Section 5.1 shall be updated and extended according to previous reviews.
  + The **final delivery** shall include a complete review of ALL contents according to the reviews. The first delivery update can be sent with the second delivery.

\*REMOVE THIS PAGE FROM THE FINAL VERSION OF THE DOCUMENT\*

# introduction

This chapter shall present an overview of the whole document.

## Purpose of this document

This section shall:

1. Define the purpose of this document (it is about the document no the Project).
2. Specify the target audience.

## How to use this document

This section shall:

1. Describe the structure and contents of the document.

## References

This section shall include a complete list of ALL standards, tools, documents, web pages, etc. cited and used during the development of the project. To do so the IEEE format for citing is recommend (see <https://www.ieee.org/documents/ieeecitationref.pdf>). It is also recommendable to use a kind of reference manager such as Zotero, Mendeley, Bibtex or the one included in the text processor.

* <https://www.zotero.org/> (standalone version as a plugin for the main text processor)
* <https://www.mendeley.com/> (on-line tool)

# dESCRIPTION

This chapter shall describe the problem statement (business objectives). It does not include any system requirement but the proper information to understand them. This section shall serve us as an introduction to the problem.

## Product perspective

This section presents a context for the development of the new system. A review of similar products shall be provided including: 1) the status of existing systems (if any) 2) the evolution of existing systems to the new one (if applicable) or 3) the replacement of an existing system to the new one. All reviewed systems shall be properly cited.

Furthermore, this section shall include a state of the art of similar systems in the market with the aim of establishing potential synergies, critical success factors or key features to be developed. Depending on the extension authors should evaluate the possibility of separating this section in a new Annex.

## Scope of the product

This section shall:

1. Enumerate and identify which software products will be generated.
2. Explain the overall objective of the system (what it is covered and what it is not).
3. Describe the expected benefits, objectives and goals of the project. A detailed description is expected but, please do not include requirements (see Chapter 3).
4. Ethics. It is important to include ethical implications (if any).

## Business case for the product

This section shall include a description of the services offered by the system. More specifically, a detailed justification of every service is expected. It is also necessary to describe the business process that motivates the project to have a better understanding of the system requirements.

Please, pay attention to the difference between the business process and the software or application that will support it (see Section 2.6 Operational environment).

## Restrictions

This section shall describe any circumstance that can restrict or limit the development of the system. Again, general circumstances that motivate requirements are expected but not specific requirements.

## Types of users: roles and capabilities

This section shall describe the stakeholders. A table showing roles and services/functions/capabilities is expected to map users to capabilities. It is similar to a high-level use case diagram but a table is usually enough.

It is possible that many types of users will interact with the system in an operational environment. Different roles can be then found: end users, IT personal, etc. and their education, language, experience or technical background could affect to some of the system requirements.

Although a software system can be frequently used by some users, others will use the system just for specific activities. Frequent users will become expert users while others will face again and again a kind of new interface. That is why a classification of users is relevant to have an (real) estimation of the necessities for each type of user.

A simple description of the user interface is also expected including the type of environment (console, desktop, web or mobile application) and other relevant characteristics for a meaningful user interface such as the voice-based interfaces, etc.

## Operational environment

This section shall describe the environment in which the system will be deployed. It shall include both natural and artificial characteristics of the environment: other software systems, mechanical devices, natural environment characteristics, etc. This description can be done using a *context diagram* to represent external interfaces and *activity diagrams* to illustrate the interaction of the system with the external agents.

If the document describes a part of other system a description of the whole system is expected.

# REQUIREMENTS

This chapter shall provide a complete software requirement specification. Clients will validate the set of requirements while quality control methods will be used to verify the requirements. Requirements shall be written accomplishing with the quality rules established in standards such as IEEE-830.

A key enabler to create a good requirements specification lies on the creation of a domain vocabulary. Thus requirements will only include those entities and relationships that are part of a controlled vocabulary. Usually, both (requirements and the domain vocabulary) are created in parallel.

## Requirements classification

This section shall explain the criteria that have been followed to classify and group requirements. It is important to group requirements according to some criteria such as type of requirement (functional/non-functional), module, component, sub-system, etc. A plain list of requirements is not clear enough in most of cases.

Therefore, it is mandatory to sort and group requirements following the criteria of modularity explaining why the selected strategy is the proper one. More specifically, functional requirements shall be grouped in “thematic” modules (commonly in components/sub-systems/services). Non-functional requirements shall be grouped according to the criteria established in some standard such as ESA or the one available in the slides.

## Template for requirements

Requirements shall be presented in a table following some template and including/justifying the most relevant attributes.

A verification test plan is a mandatory item in this project. Verification tests shall be specified for every requirement as a field in the aforementioned table.

## Functional Requirements

This section shall contain the description of the functional requirements. The requirements will be specified using natural language. In general, text-based requirements are more comprehensive and human-readable although the use of natural language also implies some potential errors in terms of ambiguity, inconsistency, etc. It is necessary to ensure that text-descriptions do not contain the typical errors when we use natural language.

## Non-functional Requirements

This section shall contain the non-functional requirements. Again text-based requirements will be used to specify the constraints, the context in which the system will work. Non-functional requirements can affect just to a set of functional requirements or the whole system. It is then necessary to specify which parts of the system will be affected by the non-functional requirements (functionality, a module or the whole system).

## Domain vocabulary

The domain vocabulary (also known as “glossary”) can be divided into two groups:

### Conceptual model terminology

This set of terms defines the concepts, entities and relationships that appear in some domain. These terms can be just simple terms referring to some entity or capability. Usually the creation of the domain vocabulary starts during the requirements elicitation process. It is a building block to be able to write proper requirements and to create a conceptual model. It also serves us to bridge the gap between the text descriptions of requirements and models. Furthermore, we can use the terminology to evaluate some quality characteristics of the system artifacts such as completeness.

### Technical terminology

This set of terms includes all technical terms related to the design or technology that will be used to implement the system. They do not have any direct relationship to the conceptual although they can appear in some requirements. It is not necessary to define all technical terms (e.g. interface is something that is already known in the context of this course) just those that can be relevant to understand the set of requirements and further design decisions. The set of acronyms used along this document should be also included here.

Hint: the technical terms defined here do not appear in the conceptual model.

# ARCHITECTURE

## Conceptual model

This section shall include a model to specify the system architecture through techniques such as diagramming. The main objective of this section is to show the information architecture of our system.

The information model, or conceptual model, shall be properly justified or derived from the set of requirements. It does not make sense to include classes, attributes or operations that have not been specified in the requirements. In the same manner, all relevant concepts, attributes or operations that appear in the requirements should be reflected in the conceptual model. We should use the domain vocabulary as a reference (see Section 3.4) to exactly know whether or not something is missing.

Since diagrams will be used to represent the different parts of the system, it is important to remark that the mere inclusion of diagrams does not imply a clear and understandable model. Diagrams shall be relevant and explanatory and a text description will be useful. It is not matter of including a lot of diagrams but the relevant ones.

The class diagrams shall help us to understand the system. We strongly recommend keeping diagrams as simple as possible. Sometimes they are only a kind of graphical view of the requirements. This means that a general class diagram specifying the main relationships will be expected while other specific class diagrams could be added to represent particular parts of the system (including specific information such as attributes).

Since this section will provide us the system architecture it is forbidden to include operations in the class diagrams. This part should be done in the design for implementation but not here.

Please include a reference to the tool that has been used to create the diagrams.

## Architectural model

This section shall include an explanation of the selected development view (e.g. subsystems and components) and shall specify the dependencies between the different system components. We will focus just on a first decomposition of the system into subsystems and components to clearly state the interfaces between them. The application of the 4+1 architectural view model is expected.

Again diagrams and textual description are expected to help readers to understand the contents of the section.

The decomposition of the system shall be justified according to some criteria such as simplicity, extensibility, efficiency, maintainability and others that can be relevant. It is necessary to pay attention to the non-functional requirements.

It is not mandatory to follow any architectural model such as the MVC (Model-View-Controller) or any other “standard” model but it is completely necessary to select the proper architecture and justify why the selection is appropriate. The architecture can be based on an existing standard or can be a hybrid between different architectural models but the important part is the selection of the architectural model according to the aforementioned quality criteria.

As it has previously mentioned, please include a reference to the tool that has been used to create the diagrams.

## Component specification

This section shall include a detailed description of all system components. A component will have a name and a unique, clear, concise and meaningful identifier. It is possible to use some kind of acronym or naming convention to ensure that all component identifiers are consistent and understandable.

The main task in the component specification lies on the description of the interfaces that must be the same that we have defined in the previous section.

### Component A (for all the components)

The description of the components shall follow the next template:

Type

There are basically two main types of components: executable and non-executable. Non-executable components can be configuration files, web page templates, etc. In order to specify the components we will focus on their type: executable components (e.g. processing X) and non-executable components (e.g. data). Each section shall be adapted to the required component description.

Purpose [Function, Processing, Data, Resource]

The purpose of this section is to define the aim of the component creating traceability links against requirements. A brief description is expected making an alignment to the requirements (implemented by). A detailed specification of traceability links is also expected in Section 5.3.

Interfaces (provided interfaces)

*“An interface[[1]](#footnote-1) is a classifier that declares of a set of coherent public features and obligations. An interface specifies a contract. Any instance of a classifier that realizes (implements) the interface must fulfill that contract and thus provides services described by contract.*

*Since interfaces are declarations, they are not instantiable. Instead, an interface specification is implemented by an instance of an instantiable classifier, which means that the instantiable classifier presents a public facade that conforms to the interface specification.*

*Any given classifier may implement more than one interface. Interface may be implemented by a number of different classifiers.*

***Interfaces realized by a classifier are its provided interfaces, and represent the obligations that instances of that classifier have to their clients. They describe the services that the instances of that classifier offer to their clients.****”*

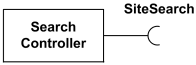
*E.g.* Interface realization dependency from a classifier to an interface.

To specify all (provided/required) interfaces it is mandatory to use the design-by-contract technique.

Dependencies (required interfaces)

*“Required interface specifies services that a classifier needs in order to perform its function and fulfill its own obligations to its clients. It is specified by a usage dependency between the classifier and the corresponding interface.*

*The usage dependency from a classifier to an interface is shown by representing the interface by a half-circle or socket, labeled with the name of the interface, attached by a solid line to the classifier that requires this interface.”*

*E.g* 

# lists and tables

## Requirements’ consistency matrix: overlapping, conflicts, etc.

## Requirements traceability – Conceptual model (classes)

## Requirements traceability – Development view (components)

1. See a detailed description here: <http://www.uml-diagrams.org/interface.html> [↑](#footnote-ref-1)