D6-Final SRS

#fr #nfr #srs #ieee-830 #reqs #er

This document presents the Contents (structure) of your Final Requirements document (based on the IEEE 830 format)

Note: in a typical SRS it is not common to use so many different techniques, so we have to tweak the typical IEEE format to accommodate the Requirements Engineering course contents.

DO NOT DO THIS: All the lines must be numbered.

Cover

Must contain the following elements:

- Software Requirements Specification
- <Project title>
- <Document version>
- Requirements Engineering 2023/24
- Authors list: <first, last name>, <student number> (one per line)
- University of Coimbra 2023

Note1: the Project Title is the team name.
Note2: please use a style as close as possible to the IEEE 830 standard.

Adopt *the Mumbai underground midnight ride* criteria* (if someone finds your SRS on the floor during a crowded midnight ride in the Mumbai underground, just by looking into your cover it must be able to know what this document is about).

Back-cover

- Project name and date
- The link to the project repository on Gitlab (publicly accessible).
- The tool used to edit the LaTeX document.
- The revision history table
- One single contact email address.

Executive abstract

A short (up to **one** page) overview of this document. Managers only read this section, so you should be able to provide a complete perspective of the system to built. It is here that you clearly state your *problem* and the proposed (in the remaining text) *solution*, as well as your *motto* (the product metaphor).

A good executive summary is one of the most difficult sections to write because it has the potential to get the project approved or liminarly rejected. *There's no second opportunity to cause a first good impression.*

Table of Contents

The table of contents should be up to three layers deep. Only on exceptional circumstances should go deeper. Make it 'clickable', i.e. if I click an entry or page number I should be redirected to that section.

Table of Figures and Table of Tables

All figures and tables must be numbered and come with a caption. In tables the legend is placed *above* while in figures the legend must be placed *below*. Never, never, *ever* throw a table or figure to the reader without referring it in the text and describing it's contents. At least the most relevant aspect(s), i.e. the point you are using the table/figure to illustrate.

1.Introduction

The introduction of the SRS should provide an overview of the entire SRS. It should contain the following subsections:

1.1.Purpose

This subsection should:

1. Delineate the purpose of the SRS;
2. Specify the intended audience for the SRS.

1.2. Scope

This subsection should:

1. Identify the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);
2. Explain what the software product(s) will, and, if necessary, will not do;
   → *this is the 'scope' of the problem statement.*
3. Describe the application of the software being specified, including relevant benefits, objectives, and goals;
   → note that this refers to the problem to be solved, not the technical solution.
4. Be consistent with this template (which is based on the IEEE-830 standard).

1.3. Definitions, acronyms, and abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This content can be delivered using alphabetically sorted tables with all the acronyms&abbreviations used in the SRS and another table for the definitions.

Definitions
Acronyms and abbreviations

1.4. External references

This subsection should:

1. Provide a complete list of all documents referenced elsewhere in the SRS;
2. Identify each document by title, report number (if applicable), date, and publishing organisation;
3. Specify the sources from which the references can be obtained.

This information may also be provided by reference to an appendix or to another document.

1.5. Document overview

This subsection should:

1. Describe what the rest of the SRS contains;

2. Explain how the SRS is organised.

Normally this is done in the same sentence, e.g. *'In section 2 we provide an overview of the product*".

2.Overall description

This section of the SRS should describe *the general factors* that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements (i.e. *the context*), which are defined in detail in Section 3 of the SRS, and makes them easier to understand.

This section usually consists of six subsections, as follows:

2.1. Product perspective

If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. → *the use-cases context diagram can be placed here.*

Relate to existing solutions

This subsection of the SRS should also put the product into perspective with other related products, so this is where the most relevant observations and insights from its main competitors are presented. Link your observations to the Annex where your survey / state-of-art is presented.

This section may include a table with a list of attributes/requirements of other solutions against which yours is going to be compared. It can also serve as source of requirements for your product in your requirements list.

Don't reinvent the wheel. First, take a look around.

2.2. Product functions

This subsection of the SRS should provide a summary of the **major functions** that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires. Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (Use-cases at level 1) that allocates particular functions to the software product. Note that for the sake of clarity:

1. The functions should be organised in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time (e.g. by role).
2. Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product, but simply shows the logical relationships among variables → *this can be illustrated using some (or all) BPMN diagrams.*

2.3. Product Context

In this section of the SRS you present your **Contextual Design** research.

Identify the actual users you interviewed, and their profile (on what relates to the project), the location and dates of contextual interviews.

Contextual Design

Include a presentation of your most relevant stakeholders (namely the client and the end-users), and all the relevant models:

1. Information flow
2. Cultural model
3. Sequence (only of core tasks)
4. Physical (optional)
5. Artefact (optional)

If a specific model does not apply, present the reason why.

Finish this section with two tables:

1. The *uniquely Identified* affinity notes you collected during the consolidation phase; if you have it, include a picture taken in an Annex.
2. A table with all the breakdowns identified. This table shall be extended in 'Breakdown Resolution' (Annex B-Traceability matrix)

Personas

This is where you put your **personas** link. You should describe the general characteristics of the intended users of the product including educational level, experience, and technical expertise.

Always include a face in your personas. You can get a computer generated one here.

2.4. Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include:
(only use those that apply to your project)

1. **Regulatory policies (e.g. GDPR);**
2. Hardware limitations (e.g., signal timing requirements);
3. Interfaces to other applications;
4. Parallel operation;
5. Audit functions;
6. Control functions;
7. Higher-order language requirements;
8. Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);
9. Reliability requirements;
10. Criticality of the application;
11. Safety and security considerations.

*This is where you put your **legal** and **regulatory** constraints.*

2.5. Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect your requirements. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

Examples:

```
- An API to an updated list of medical doctors is available and publicly
accessible.
- A reliable currency converter is publicly accessible.
- The parts list of the internal stock management system is accessible for
CRUD operations.
- A reliable Zigbee protocol library is available.
```

3. Specific requirements

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems.

As this is often the largest and most important part of the SRS, the following principles apply:

1. Specific requirements should be stated in conformance with all the characteristics described in 4.3.
2. Specific requirements should be cross-referenced to related documents.
3. All requirements should be uniquely identifiable (i.e. use unique IDs).
4. Careful attention should be given to organising the requirements to maximise readability.

3.1. Functional requirements

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs.

3.1.1. User Stories

Present *at least three* **user stories** of the most relevant functional requirements of your application.

Include, *for each user story*, a set of possible clarification comments and/or validation checks.

Example:

As a credit card holder, I want to view my statement balance, so that I can pay the balance due

- Display statement balance upon authentication
- Display Total Balance
- Show “Payment Due Date” and “Minimum Payment Due”
- Display Error message if service not responding/ timeout

3.1.2. Workflows

In this section present the workflows that represent **the core processes** needed to achieve the goals of the *problem* you are addressing, using BPMN. The BPMN models must explicitly describe the activities of the different actors including any external systems, and the artefacts and data stores required (if applicable).

Please **name each workflow** with the name of the process you are representing. Also *uniquely Identify* each step in the workflows (you shall need this for the traceability matrix).

3.1.3. Use Cases

- Provide a diagram with the Use-Cases Context diagram (level 0); → beware: this is NOT Contextual Design.
- Provide a brief description of all actors (which are presented in the context diagram)
- Provide a second diagram with the top-level Use-cases (level 1). These must be numbered (UC001, UC002,...).
- Present at least three use-cases (sea-level). These use cases MUST correspond to (at least) one activity in the workflow models above.
- Use cases must be presented in tabular format with the following fields:

Use-case fields:

- Use-case ID+Name
- Use-case level using the terminology {cloud/kite/sea/fish/clamp}

- Source: identify from where did you get this use-case (client/team/survey/...)
- Short use-case description
- Minimal guarantees (if the use-case fails to complete)
- Success guarantees (if the use-case terminates through the 'happy path')
- Trigger
- Main actor → that initiates the use-case
- Secondary actors (if applicable)
- Preconditions → this is the setup phase of the future acceptance tests.
- **Main success scenario**: numbered description of each use-case step
  → use the *two-column* format.
- Exceptions:
  → use numbered steps as for the main scenario.
- Post-conditions → this is the draft of the future acceptance tests.

OBS: each use-case description should be in a singe A4 page. Adopt a template so that all use-cases are presented in a consistent way. It should be possible to deliver one of these A4 pages to a developer to start working.

3.1.4. Navigation layout (UED)

Present, based on the Workflows and the Use-Cases above, the UED for your whole application, i.e. its *navigation layout*. Don't forget to include the connections between the *focus areas* and describe the most relevant aspects of your UED diagram.

Remember: *all figures and all tables must be numbered and have captions.*

3.2. Quality Attributes

In this section you need to present:

1. The *quality attributes* that apply to your project. **Define** each one of them.
2. (at least one) *Concrete Scenario* for each identified quality attribute.

Discuss whether there are conflicting quality attributes and how do you intend to address that conflict.

Finally, every new requirement that emerges from this dimension must be uniquely Identified using the prefix "NF" (Non-Functional).

3.3. Business constraints

Your product may involve user registration and payments to be economically sustainable. It can be a free application, one single payment upfront, or a subscription and regular payments. Identify and present any new requirements that emerge from your business constraints.

Every new requirement that merges from this dimension must be uniquely Identified using the prefix "BZ".

3.2. Legal and Regulatory constraints

Regulatory and legal issues can have a huge impact in your system features and on its internal design. In this section you must identify, and present such issues.

Every new requirement that merges from this dimension must be uniquely Identified using the prefix "LG".

3.3. Wireframes and usability checking

Select at least **three** most relevant focus areas from your UED and design the mock-ups / wireframes for each one of the selected focus areas. These may also be use-cases, so if that is the case identify to which specific part of your requirements each wireframe refers to.

→ **Each screen/wireframe** of each selected focus area must be assessed using Nielsen's usability heuristics. This is a checklist so you state either 'no' or 'check'. The total number of

checks is the usability rating of that screen, from 0-10.

Note1: *Design the wireframes using low fidelity prototyping, don't put effort into visual design, that's the job of graphic designers.*

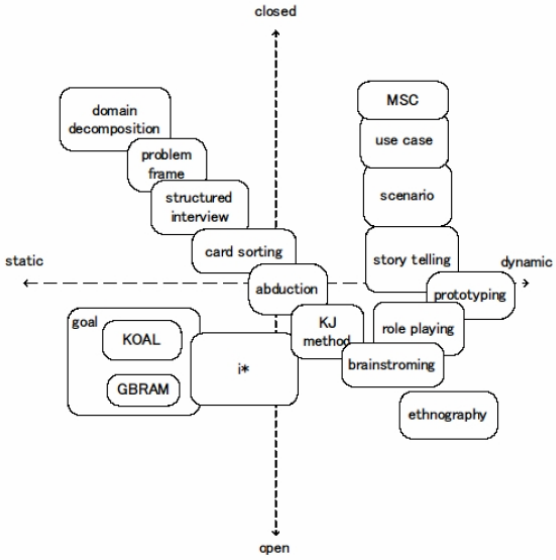
Note2: the focus areas selected for the usability check *should be* the ones whose interaction you detailed in the use-cases section. These must also be your most relevant focus areas.

Note3: each image/graphic should be in a singe A4 page. Adopt a template/style so that all images are presented in a consistent manner. If you have very large images that are difficult to read in A4, either break them (do it wisely) or replicate them in annexes, in A3 format.

4.RE Techniques assessment

In this section you:

1. Identify the nature of your problem domain (stable/open) and the technique adopted (systematic/exploratory) placing your project in the right quadrant of the plane presented in class (Week 12: A framework for RE techniques).



Then,

2. Rank the techniques you used in the first five deliverables by the value they provided on your requirements elicitation work. Identify at least the *most* and *less* useful, and justify *why*.
3. Reflect on whether your 'best' technique(s) are aligned with the framework presented in 1.

***The analysis and rationale for your opinion in this section is one of the most relevant aspects of this submission.**

Annexes

Insert here the List of annexes.

- We have included four Annexes. Add more as needed.
- Each annex must start in a new page.

Annex A - Full list of requirements

In this annex you must provide a set of (up to five) tables with a full list of every requirements identified. One table for Category of Requirement (Functional, non-functional, Business-related and Regulatory) and one for the constraints identified .

Think of it as a 'checklist' that the development team must be tracking until all requirements are developed. Another perspective is to see it as the whole Product backlog in an Agile context.

Notes:

1. Category can only be of type:
    1. FR: Functional requirements
    2. NF: non-functional requirements (quality attributes)
    3. BZ: Business requirements
    4. LG: Legal and regulatory
2. ID numbering must start at 001 for each category.

Each table must have the following format:

FR | Functional requirements

ID	Description	Priority	Source
FR001	Only authenticated users can use the app	Must	Client
FR002	Users shall have one of two roles: *client* or *admin*	Must	Client
FR...	(...)	(...)	(...)

NF | Quality Attributes (non-functional requirements)

ID	Description	Priority	Source
NF001	Under normal load, Latency must be below 2s	Should	Team
NF002	Users can access any focus area in three clicks	Must	UX

BZ | Business requirements

ID	Description	Priority	Source
BZ001	Invoices must be submitted before Dec 28	Must	Accountant
BZ002	Premium clients have 15% discount	Must	Client (CFO)

LR | Legal an regulatory requirements

ID	Description	Priority	Source
BZ001	IRS bracket level calculated according to IRS code	Must	Accountant
BZ002	Users can request account delete	Must	RGPD

CONSTRAINTS

ID	Category	Description	Source
C001	LR	Clinical and user data cannot be in the same database	RGPD
C002	NF	The application must be developed using Java technology	Client (CTO)

Notes:

1. Constraints have no priority field as they are always mandatory.
2. Constraints must refer the category to which the constraint applies to.

Annex B - Traceability matrix

A Traceability matrix mapping the requirement ID (functional only) to where it shows up at the external interface (if applicable).

REQ.ID	Description	UED	UC	Wireframe
FR001	Clinical and user data cannot be in the same database	[3]	UC002	[7]
FR002	Only authenticated users can use the app	[5]	UC003	[5]
FR003	Users shall have one of two roles: *client* or *admin*	[3]	UC010	[2,5]

Breakdown resolution

Include a table where the list of breakdowns is mapped into the requirement(s) ID(s) that address/solve these breakdowns. If a breakdown could not be solved by your new system state is as N/A (not achieved).

BRK.ID	Description	REQ.ID
BRK001	Patients don't have reliable information to choose a doctor	UC001
BRK002	My financial data may be locked by a provider	FR032
BRK003	A sitter my steal my assets	FR012

Annex C - Larger resolution diagrams

In this annex you must put higher resolution diagrams whenever they are large and the size in the main text is considered not very readable. Create sub-sections for each category of diagrams (BPMN, Wireframes, etc.)

Annex D- Survey

Whatever is the problem we are trying to address, there is always someone, somewhere that has already thought of it. It might not be exactly what we intend, it might be somehow off-topic, but we need to find it. In same cases it is obvious, in others not so much.

Here you are required to survey the internet (and libraries and books and wherever sources you consider appropriate) and find what has already been done to address your problem.

Thus, your goal is to find the current, cutting-edge, and most advanced work and **competing solutions** related to your problem. It refers to the current, real-world methodologies, techniques, tools, and standards that are commonly used in a particular industry, profession, or field of practice related to your specific problem.

SRS Checklist (for the reviewers)