RSL-IL Excel Template: A System Requirement Specification based on the RSL-IL Language [[1]](#footnote-1)

1

Alberto rodrigues da silva, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, alberto.silva@tecnico.ulisboa.pt

Daniel Serrão, Instituto Superior Técnico, Universidade de Lisboa, daniel.serrao@tecnico.ulisboa.pt

Tiago Catarino, Instituto Superior Técnico, Universidade de Lisboa, tiagomcatarino@tecnico.ulisboa.pt

Most of the research carried out in Requirements Engineering area focuses on requirements modelling, and forgotten that most business stakeholders continues to document the requirements using natural language. The writing of such documents in natural language leads to errors such as ambiguity and inconsistency. This article describes and shows the activity carried out in the Requirements Engineering area, in order to deepen the knowledge about it and develop a template that would support stakeholders and goals of any software system during the validation requirements and until the end. To mitigate the problem described above, it was used in this research work RSLingo approach. This article presents a description in detail of the investigation at Instituto Superior Técnico.

**Key Words and Phrases**: Requirements Engineering, System Modeling, RSLingo, **SRS**, RSL-IL, Template, Requirements Specification

# INTRODUCTION

In the requirements engineering (RE) community, “system requirements specification” or “requirements specification” (SRS) is a document that describes multiple technical concerns of a system, typically software systems but also blended software and hardware based systems [2,3].

A SRS is used throughout different stages of the project life-cycle to help sharing the system’s vision among the main stakeholders, as well as to facilitate the communication and the overall project management and system development processes. A good SRS provides several benefits, namely [4,5,6,9,10]: establish the basis for agreement between customer and supplier; provide a basis for estimating budget and schedule; provide a baseline for validation and verification; and serve as a basis for future maintenance activities. A SRS is sometimes also integrated in legal documents surrounding project’s Request for Proposals or Contracts. A SRS tends to follow a previously defined template that prescribes a given document structure (with chapters and sections) with supplementary practical guidelines. By definition, SRS templates should be adapted and customized to the needs of the organization involved. In any case, these templates prescribe the use of multiple (RE specific) constructs and models – corresponding to different views and perspectives of the system, for example goal-oriented, context, domain or use case models – that can be considered “modular artifacts”, in the sense of their definition and reuse. However, there are several dependencies among and even intra these modular artifacts, which are important to minimize or prevent (to some extent).

In a prevous work we discussed the result of our experiences in looking for combinatorial effects (CE) at different levels in the RE-process [1]. The examples covered CE at the RE level based on the adoption of notations and techniques such as UML (classes and use case diagrams), DEMO/EO, and BPMN models. Those examples are relatively straightforward, but enough to show the omnipresence of such instabilities in the RE level. As a result, we described the need for a research agenda focusing on the systematic research into CE and related issues at the RE domain in order to build enterprises and their information systems that are able to exhibit new levels of agility. Then, in a recent paper [???] we compared the modular structures of three SRS templates in terms of the extent to which they prevent CE from happening; and we proposed a set of practical recommendations to define a SRS template that would mitigate the referred problem.

This technical report introduces and discusses the proposed “RSL-IL Excel Template” that is a SRS Excel template based on the multi-view architecture defined in the RSL-IL language [???]. The proposed template includes the following files:

RSLIL-ExcelTemplate.xsl (the Excel template to be configured and used in a project basis);

RSLIL-Example-BillingSystem.xsl (a companion simple example that illustrates the use

Requirement Engineering (RE) refers to the process of defining, documenting and managing requirements [1]. The main goal of the RE is to provide development team with sufficient insight of the problem under consideration. This way it’s better to develop a software system that is more aligned with to the needs of the stakeholders. Business stakeholders express themselves in free natural language, and the development teams have to make an extra effort to understand and then to develop a solution that complies with these requirements [2]. With RE is possible to mediate the communication between stakeholders and development teams, which is required to reduce the project failure risks associated with the lack of quality of the documentation that contains the requirements specification. The main purpose of this SRS is to help and guide to specify the requirements of software systems. This document describes the views of this template, namely glossary, stakeholders, goals, actors, structural, use cases and requirements views. This template encourages business stakeholders to actively authoring requirements in natural language. All sheets are arranged in the same format, allowing this template to be used for modeling any systems. From the information contained in these sheets it’s possible to prepare (manually or automatically) the diagrams to help visualize the respective systems.

# RSL-IL

RSLingo is an information extraction approach based on linguistic patterns. The RSLingo approach follows a multi-language strategy based on two languages: RSL-PL (Pattern Language) and RSL-IL (Intermediate Language) [4].

The goal of RSL-IL is to force the projection of natural language requirements representation, i.e. mitigate requirement defects through the formalization of requirements specification documented by business stakeholders in natural language. RSL-IL is a formal textual language specifically designed to address the most of concerns of RE toward the development of software-intensive systems. This language plays the role of an *intermediate language* in the context of RSLingo approach but can be use by requirements engineers to specify software requirements. RSL-IL can be regarded as back-end language for formally documenting/storing software requirements.

The RSL-IL is organized into two abstraction levels: Business Level and System Level (figure 1).

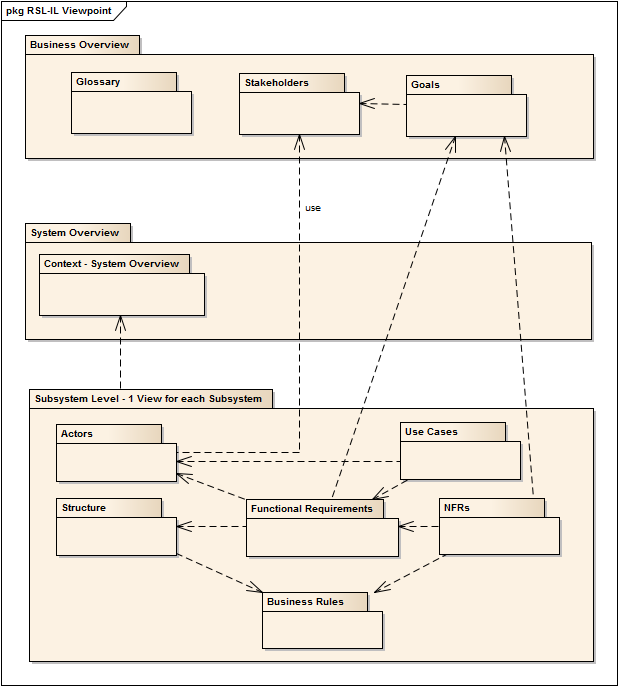


Figure 1: View dependencies and their respective level

The rationale of this organization is to ensure that RSL-IL covers all the abstraction levels related with requirements documentation, connecting both upstream with the business context, and downstream with the system context. This language provides several constructs that can arranged into viewpoints according to the specific RE concerns they address (figure 2).



Figure 2. Classification of RSL-IL viewpoints: levels versus interrogatives

## Business-level Views

### Glossary view

Glossary view includes all the domain and system terms used throughout SRS. The main purpose of this view is to reduce the negative effects of natural language’s imprecision through ambiguity resolution techniques. The little effort required to identify and properly define the most relevant terms of the Universe of Discourse of business stakeholders brings significant advantages, because these terms refer to the most important real world notions to be considered and they help everyone to use consistently the same terms with the same meaning while constructing the system.

This view is not dependent from any other view but other views should use the terms identified in it.

### Stakeholders view

Stakeholder defines the most important source of requirements, identified as stakeholders. Without stakeholders identification we cannot ensure that the delivered software is a suitable solution because either: (1) vital information might be missing, since an important Stakeholder (or a group of them) was not identified early in the project; or (2) one is unable to determine the reason why a requirement was originally stated, or even to clear out its true meaning in subsequent phases.

This view have only one dependency, the chose roles for the stakeholders need to be present as a term in the glossary. The views Actors and Goals depend of this view.

### Goals view

Goals view intention is to answer 2 fundamental questions: (1) “Why is this software system needed?” and (2) “How can coarse-grained, high-level business objectives be decomposed into concrete, realizable requirements”. Also, it allows to establish a bridge from the system-of-interest’s capabilities to the business context.

This view is dependent to the Stakeholders view, each goal in this view need to have a stakeholder that can be identified at the Stakeholder view. Other views are dependent from this one because they need to satisfy the goals, views like “Functional Requirements”, “Non-Functional Requirements – Sub-system Level” and “Non-Functional Requirements – High-Level”.

## System-level Views

### Actors view

Actors view shows the actors from the system, these actors represents the role played by a Stakeholder and they interact with the system.

This view is dependent to the Stakeholders view because each actor need to be associated with an identified stakeholder in the Stakeholder view. Views that depends from this view his “Use Cases” and “Functional Requirements”.

### Structural view

Structural view shows all the informational entities and relationships between them. It shows the data perspective of the requirements and what information the system will need to deal with. Is also possible to represent restrictions of the System-of-interest, such as putting a value range limit into an attribute.

### Use Cases view

Use Cases view shows one of the most versatile techniques to specify complex functional requirements. Despite is simplicity, Use Cases provides a practical way to describe and analyze part of the behavior of the system-of-interest based on the concept of scenario.

This view is dependent to the Actors view because each Use Case need to have at least an actor, and each actor need to be identified in the Actors view. It don’t exist any view dependent to this one.

This view is dependent to the Requirements view because each Use Case need to specify a requirement.

### Requirements view

Requirements view focus on the specification of individual requirements, and aims to provide a kernel that can be used to convey information about requirements based on the traditional requirements documentation approach. Some of this view advantages is to provide means to: (1) classify individual requirements according to their type (i.e., functional, quality or constraint); (2) specify their attributes (either intrinsic to them, or required for management purposes); (3) traceability relations; and (4) the rationale history. This view provides the “glue” required to connect the Business Level with the System Level because the requirements are there to satisfy the Business goals in the Goal view.

## Other views to consider

We still could consider other views, namely Business Rules View and Non-Functional Requirements View. **Business rules view** define the conditions or constraints imposed to the system-of-interest, some examples of rules are decisions, calculations or triggers. One important method for enforcing business rules is to define and to apply domain restrictions. By restricting and validating an attributes values, you can implement important business rules such as ensuring that a checking account maintains a positive balance, or preventing the entry of invalid phone numbers. **Non-Functional Requirements View** but in our understanding this view only makes sense in cases that the system is very complex in a way that the number of measurable requirements would be very high, in our case we only used a simple billing system as an example and most non-functional requirements could easily be represented in the Domain Model View.

# RSL-IL Excel Template

## Configuration and Homes Sheets

home Sheet

The home sheet has the index of template sheets, each line have one link for one sheet.

Using this sheet is easier to see and to access to sheets that exists in the template.

config Sheet

The config sheet lists the terms that are frequently used in the SRS. These terms can be referred in any other sheet. We can also see in this sheet which classes or terms are used in which views or sheets. During the explanation of the business level sheets and system level sheets, we will explain most of the classes and terms that are identified here.

rslil home Sheet

The RSLIL-Home sheet describes the System and Sub-Systems, we can also define the decomposition of the system and define which systems are part of which systems. The data in this sheet can serve as input to create a visual model of a System Architecture or a context.

## Business Level Sheets

Glossary View (rslil.glossary sheet)

This sheet includes the terms used in this SRS and has the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Id | Unique identifier of the term. |  |
| Class | The type of the term. A term can have more than just a type. | stakeholder, actor, entity, architectural, stakeholder.actor, stakeholder.entity, stakeholder.actor.entity |
| Word | The name of the term. |  |
| Definition | Define what is and the purpose of the term. |  |
| Synset | Is a synonym set or a group of synonims and corresponds to an abstract concept that can be used together with a software (e.g. TextBob) and with a Wordnet Database to define relationships with other words. | Is-a, Part-of |
| POS | Means part-of-speech and define if the term is a noun, adverb or verb. | noun, adverb, verb |
| Term relation | Id of another term used in the glossary to define a relation with the term of the current line. |  |
| Term relation type | Defines the type of the relation of the chosen term above. | antonym, hypernym, synonym. |
|  |  |  |

Stakeholders View (rslil.stakeholders sheet)

This sheet define the stakeholders of the system we use the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Id | Unique Identifier of the Stakeholder. |  |
| Class | The class of the stakeholder. | group.organizational, group.business unit, group.team, individual.person, individual.external |
| Name | The name of the stakeholder. |  |
| Description | The purpose and/or what the stakeholder do with the system. |  |
| Role | The role of the stakeholder in this system context. |  |
| Is aggregated by | The Id of other stakeholder that is aggregated by the stakeholder of the current line. |  |
| Category | The category of stakeholder. | business.customer, business.customer.sponsor, business.champion, business.user.direct, business.user.indirect, business.advisor.expert, business.advisor.trainer,  business.advisor.regulatory |

Goals View (rslil.structural sheet)

This sheet define the goals of the system, we use the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Id | Unique identifier of the goal |  |
| Description | Describe the goal |  |
| Source (Stakeholder) | The Stakeholder who defined the goal. |  |
| Criticality | The level of importance and priority of the goal. | Very low, low, medium, high, very high |
| Dependent by | Here we write an existent goal in this sheet. The current line goal is in some form, dependent to the chosen goal. |  |
| Dependency type | Type of dependency of the current line goal | Requires, Supports, Obstructs, Conflicts, Identical |
| Decomposed by | Here we can write one or more existent goals in this sheet. The current line goal is decomposed by all the chosen goals. |  |
| Decomposition type | The decomposition has 2 types. The type “and” which means that all the goals in the “Decomposed by” need to be fulfilled to fulfill the current line goal. The type “or” which means that only one goal need to be fulfilled to fulfill the current line goal. | And, or |

## System Level Sheets

Actors View (rslil.actors sheet)

This sheet define the actors of this system, we use the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Id | At the blue line we define the system unique identifier (normally S1,S2,…,Sn) and at the other lines we define the actors unique identifiers. |  |
| Name | At the blue line we define the name of the system and at the others lines we define the name of the actors. |  |
| Responsibilities | Only used for the white lines related to the actors and define the responsibilities of the actor. |  |
| Specialization of | Id of an existent stakeholder. The current line actor will be the specialization of that stakeholder. |  |
| Stakeholder | Name of chosen stakeholder |  |

Structural View (rslil.structural sheet)

This sheet define the entities of this system, we use the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Id | Unique identifier of the entity. |  |
| Name | At the blue line we write the name of the entities and at the others lines we write the name of the attributes. |  |
| Description | At the blue line we write the description of the entity and at the other lines we write the description of the attribute. |  |
| Type | This column is only used at the white lines, and it defines the type of the attributes | Boolean, integer, decimal, currency, date, time, date time, enumeration, text, regex, ref, image |
| Default value | This column is only used at the white lines and it defines the default value of the attribute. |  |
| References to | Here we write the name of another entity if needed. The chosen entity will have some type of association with the current line entity. |  |
| Multiplicity | We need to select here the multiplicity of the association defined at “References to” if that happen. | 0, 1, …, \* |

Use Cases View (rslil.usecases sheet)

This sheet define the use cases of this system, we use the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Id | Use case unique identifier. |  |
| Name | Name of the use case. |  |
| Description | Description of the use case. |  |
| Accomplished goals | Goals accomplished by the use case. |  |
| Functional Requirements | Ids of the functional requirements that the use case fulfill and that exist at the Requirements View (rslil.requirements) |  |
| Actor initiates | Ids of the actors that can start the use case and that exist at the Actors View (rslil.actors). |  |
| Actors participates | Here we need to write the ids of the actors that can participate in the use case. |  |
| Pre-conditions | Conditions needed to start the use case. |  |
| Pos-conditions | Conditions needed to fulfill the use case successfully. |  |
| Include | Ids of the use cases that are included in the current line use case. |  |
| Extended By | Ids of the use cases that extend the current line use case and the related extension points. |  |

Scenarios View (rslil.usecases.Sn sheet)

This View is part of the Use Cases View. Each Scenario View must have only use cases related to a sub-system of the system-of-interest. A Use Case View can have 1 or more Scenarios Views. Each of these views have the following columns:

|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| Name | Name of the scenario. |  |
| Type | Type of the scenario. | MainScenario, AlternativeScenario, ExceptionScenario |
| Sequential | Boolean value, the steps of the scenario can be sequential or not. | TRUE, FALSE |
| Label | This is the number of the steps, in the case of sequential, the first steps are the ones with the smaller number at the label. |  |
| Text | Description of what happen at the step. |  |
| Next Label | Number of the next step after the current step. Most of the times we don’t need to write this number because the default value of the next label is the current label plus one. |  |
| Actor subject | Name of the actor that executes the current step. |  |
| Pre-conditions | Conditions that need to be fulfilled to start the step. |  |
| Pos-conditions | Conditions that need to be fulfilled to finish the step successfully. |  |
| Action Type | It defines the type of the step and who execute it. A step can also be 2 types at the same time. | ActorPrepareData, ActorCallSystem, SystemExecutes, SystemReturnResult, ActorPrepareData.ActorCallSystem,  SystemExecutes.SystemReturnResult |

Requirements View (rslil.requirements sheet)

In this sheet, to define the requirements of this system we use the following columns:

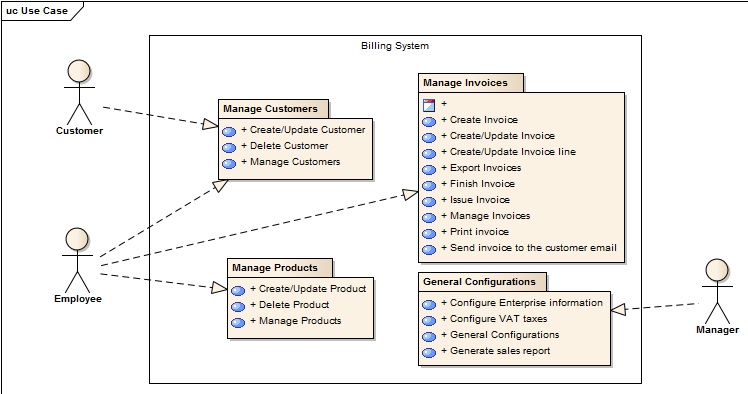
|  |  |  |
| --- | --- | --- |
| Column Name | Description | Values |
| FR Id | Unique Identifier of the functional requirement. |  |
| System Id | Id of the sub-system related to the current functional requirement. |  |
| Obligation | Obligations that the requirements must fulfill. |  |
| Proposer | Stakeholder who created or want the current functional requirement. |  |
| Creation Date | Date which the requirement was created. |  |
| Priority | The level of importance or priority of the current requirement. |  |
| Sentence | Description of the requirement. |  |
| Action Verb | Action Verb used by the requirement. |  |

# Example – THe Billing System

A billing software handle the tracking of billable products and services delivered to a customer or set of customers. These types of software automate much of what used to be a time-consuming process of preparing invoices or other related documentation. The modern digital structures provided by billing software is part of what has propelled businesses into the new digital era, allowing for more productivity and greater ease of business administration in general.

This example describes the goals, stakeholders, actors, entities, use cases, requirements and scenarios of the Billing System, providing a framework for requirements specification.

This example is a simple one and based on four groups of use cases, we can have a better idea with the following image.



# Guidelines for using the RSL-IL Excel

This Excel template can be used by any company that wishes to specify SRSs. The biggest advantage of this template, it should be noted the correct completion of each sheet. There are sections (such as descriptions of use cases, etc.) that can be written using natural language. On the other hand, there are sections where it is necessary to use a certain classifiers, this being defined in the config sheet. These classifiers are now defined so as to reduce errors sometimes occur when using the natural language for the RE process.

All classifiers used to define the types of glossary, stakeholders, goals, etc. should be placed in rslil.config sheet. After this step it is necessary to use the option "Define Name" (click with right button of the mouse), in order to limit the error in filling the other sheets. Next must be select the cells where they will appear and using the option "Validation of Data" it’s possible to made the connection between what is defined in rslil.config sheet with the sheet where we want to use the terms. More information to help to implement this can be found here: <https://support.office.com/en-ca/article/Define-and-use-names-in-formulas-4d0f13ac-53b7-422e-afd2-abd7ff379c64>.

Regarding the sheet home, it contains content that is dynamic, so we can add more sheets and the index is updated automatically. More information to help to manage the index can be found here: <http://www.ozgrid.com/VBA/sheet-index.htm>.

# REFERENCES

[1] Alberto Silva, Jan Verelst, Herwig Mannaert, David Ferreira, Philip Huysmans, Towards a System Requirements Specification Template that Minimize Combinatorial Effects

[2] David Ferreira, Alberto Silva, INESC-ID, RSLingo: An Information Extraction Approach toward Formal Requirements Specifications

[3] David Ferreira, Alberto Silva, INESC-ID, RSL-IL: An Interlingua for Formally Documenting Requirements

[4] David Ferreira (PhD Thesis), RSLingo: A Formal Requirements Specification Approach based on Linguistic Patterns

[5] Alberto Silva, Carlos Videira, UML, Metodologias e Ferramentas CASE

[6] Klaus Pohl, Requirements Engineering, Fundamentals, Principles and Techniques

1. This work is supported by the Instituto Superior Técnico.

   Author's address: A. Silva, email: alberto.silva@tecnico.ulisboa.pt; D. Serrão, email: daniel.serrao@tecnico.ulisboa.pt; T. Catarino, email: tiagomcatarino@tecnico.ulisboa.pt

   © 2015 Instituto Superior Técnico, Universidade de Lisboa [↑](#footnote-ref-1)