RSL-IL Excel Template: A System Requirement Specification based on the RSL-IL Language

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

Requirements specifications (SRSs) describe multiple technical concerns of a system and are used throughout different stages of the project life-cycle to help sharing the system vision among the main stakeholders, as well as to facilitate communication and the overall project management and system development processes. For achieving an effective communication, everyone should be able to communicate by means of a common language, and natural language provides the foundations for such language. However, although natural language is the most common and preferred form of requirements representation, it also exhibits some intrinsic characteristics that often present themselves as the root cause of many requirements quality problems, such as incorrectness, inconsistency, incompleteness and ambiguousness. The RSL-IL is a requirements specification language that includes several constructs logically arranged into viewpoints according to the specific RE concerns they address. This article introduces in a simple way the "RSL-IL Excel Template", which is a SRS template based on the RSL-IL Language and can be customized or just used for your own purpose.

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

1. INTRODUCTION

Requirements Engineering (RE) intends to provide a shared vision and understanding of the system to be developed between business and technical stakeholders. The adverse consequences of disregarding the importance of the early activities covered by RE are well-known. System requirements specification, software requirements specification or just requirements specifications (SRS) is a document that describes multiple technical concerns of a software system. An SRS is used throughout different stages of the project life-cycle to help sharing the system vision among the main stakeholders, as well as to facilitate communication and the overall project management and system development processes.

For achieving an effective communication, everyone should be able to communicate by means of a common language, and natural language provides the foundations for such language. Natural language is flexible, universal, and humans are proficient at using it to communicate with each other. Natural language has minimal adoption resistance as a requirements documentation technique. However, although natural language is the most common and preferred form of requirements representation, it also exhibits some intrinsic characteristics that often present themselves as the root cause of many requirements quality problems, such as incorrectness, inconsistency, incompleteness and ambiguousness. In our recent research we consider the RSLingo approach [1-4] as a starting point for this challenge. RSLingo is an approach for the formal specification of software requirements that uses lightweight Natural Language Processing (NLP) techniques to translate informal requirements – originally stated in unconstrained natural language by business stakeholders – into a formal representation provided by a language specifically designed for RE.

This work is supported by the Instituto Superior Técnico.
Author's address: A. Rodrigues da Silva, email: alberto.silva@tecnico.ulisboa.pt; D. Serrão, email: daniel.serrao@tecnico.ulisboa.pt; T. Catarino, email: tiagomcatarino@tecnico.ulisboa.pt

Publication date: July 2015

A good SRS provides several benefits, namely: 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 previous work we discussed the result of our experiences in looking for combinatorial effects (CE) at different levels in the RE-process [5]. 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 were 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, recently 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 [6].

This document introduces and discusses the proposed "RSL-IL Excel Template" that is a SRS template based on the multi-view architecture defined in the RSL-IL language [2] and also aware of the problems and practical recommendations discussed [6]. The proposed template includes the following files:

- RSLIL-ExcelTemplate.xsl (the companion Excel template to be configured and used in a project basis);
- RSLIL-Example-BillingSystem.xsl (a companion Excel file with a simple example based on the specification of the "BillingSystem").

The main purpose of this document is to help and guide to better customize and use this SRS template. This document briefly introduces the views (Excel sheets) included in this template, namely: glossary, stakeholders, goals, actors, structural, use cases and requirements views. Each view is represented as an Excel sheet, and all sheets are arranged in the common tabular format. In addition, using the information defined in this template, it is possible to produce (manually or automatically) complementary diagrams (e.g., simple UML diagrams) to help visualizing the respective system.

2. RSL-IL LANGUAGE

RSLingo is an approach for the formal specification of software requirements that uses lightweight Natural Language Processing (NLP) techniques to (partially) translate informal requirements – originally stated by business stakeholders in unconstrained natural language – into a formal representation provided by a language specifically designed for RE [1]. The name RSLingo stems from the paronomasia on "RSL" and "Lingo". On one hand, "RSL" (Requirements Specification Language) emphasizes the purpose of formally specifying requirements. The language that serves this purpose is RSL-IL, in which "IL" stands for Intermediate Language [2]. On the other hand, "Lingo" expresses that its design has roots in natural language, which are encoded in linguistic patterns used during by the information extraction process that automates the linguistic analysis of SRSs written in natural

language. The language designed for encoding these RE-specific linguistic patterns is RSL-PL, in which "PL" stands for Pattern Language [11]. These linguistic patterns are used by lightweight NLP techniques and, when combined with general-purpose linguistic resources (e.g., VerbNet (http://verbs.colorado.edu/~mpalmer/projects/verbnet.html) and WordNet (http://wordnet.princeton.edu)), enable the extraction of relevant information from the textual representations of requirements. Finally, the extracted information with these lightweight NLP techniques is formally specified in RSL-IL notation through predefined transformations from RSL-PL into RSL-IL. Upon a match of a requirement's textual representation with one of the RSL-PL linguistic patterns, a transformation should become active. This transformation takes into consideration the semantic roles of each word within the linguistic pattern, and drives the mapping between RSL-PL and RSL-IL.

RSL-IL provides several **constructs** that are logically arranged into **viewpoints** according to the specific RE concerns they address [2]. These viewpoints are organized according to two abstraction levels: business and system levels (Figures 1 and 2).

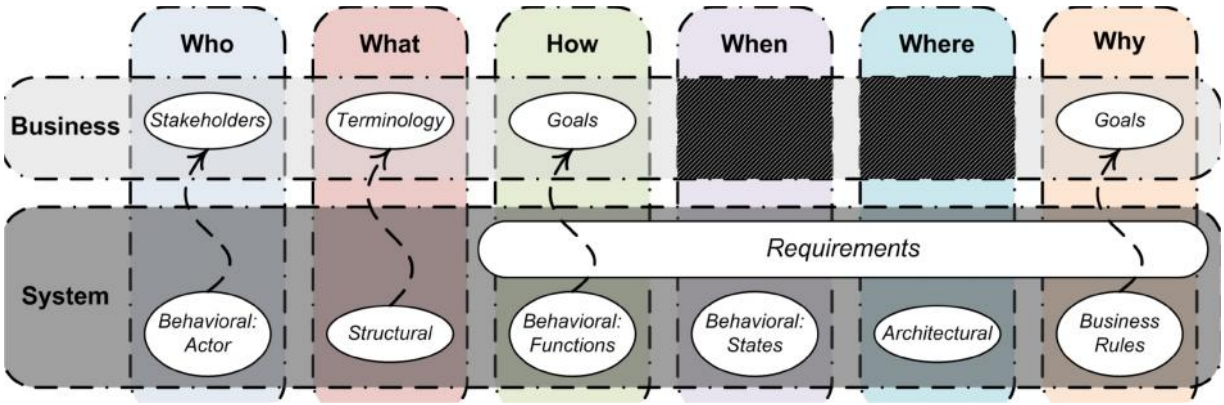

Figure 1. Classification of RSL-IL viewpoints: levels versus interrogatives [2].

To properly understand and document the business context of the system, the **business level** of the RSL-IL supports the following business-related concerns, namely: (1) the concepts that belong to the business jargon; (2) the people and organizations that can influence or will be affected by the system; and (3) the objectives of business stakeholders regarding the value that the system will bring. Considering these concerns, business level requirements comprise respectively the following viewpoints: Terminology, Stakeholders, and Goals.

On the other hand, at the **system level**, the RSL-IL supports the specification of both static and dynamic concerns regarding the system, namely: (1) the logical decomposition of a complex system into several system elements, each with their own capabilities and characteristics, thus providing a suitable approach to organize and allocate requirements; (2) the requirements that express the desired features of the system, and also the constraints and quality attributes; (3) the data structures aligned with the business jargon, their relations, and a logical description of their attributes; and (4) the actors, functions, event-based state transitions, and use cases that further detail the aforementioned requirements. Considering these concerns, the System Level comprises the following viewpoints: Architectural; Requirements; Structural; and Behavioral, respectively.

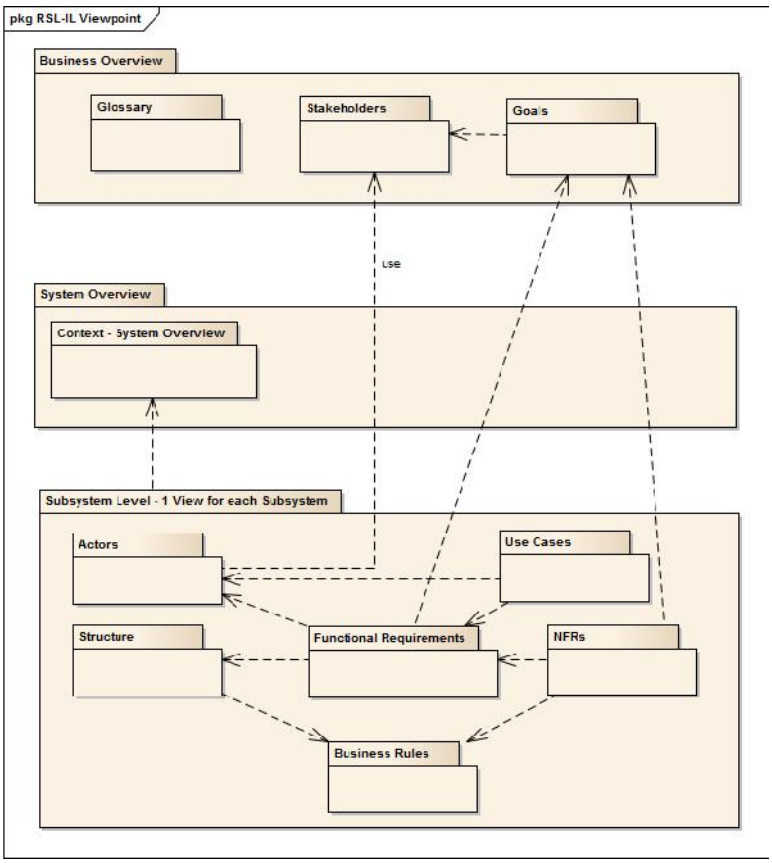

Figure 2: Viewpoint dependencies and respective abstraction levels.

2.1 Business-level Views

Glossary view. Glossary view should include the domain and system terms used throughout the 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 used by 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 defining and building the system.

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

Stakeholders View. Stakeholders view defines the most important source of requirements, identified as stakeholders. Without stakeholders identification you 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 has only one dependency: the most relevant stakeholder names should be present in the glossary; the views Actors and Goals depend of this view.

Goals view. Goals view intention is to answer to two fundamental questions: (1) Why is this software system need? and (2) How can coarse-grained, high-level business objectives be decomposed into more concrete requirements? Also, it allows establishing a bridge between the system-of-interest's capabilities and its business context.

This view depends on the Stakeholders view: each goal needs to define its stakeholder(s) (that should be previously defined at the Stakeholder view). On the other hand, views like "Functional Requirements", "Non-Functional Requirements" might show dependencies to the Goals view.

2.2 System-level Views

Actors view. Actors view defines the actors of the system; these actors represent the roles played by the end-users (a kind of stakeholder) and other third-party information systems.

This view dependents on the Stakeholders view because some actors need to be associated with an identified stakeholder in the Stakeholder view. On the other hand, views like "Use Cases" and "Functional Requirements" might show dependencies to the Actors view.

Structural view. Structural view defines the "informational entities" (aka "data entities" or just "entities") and respective relationships. It shows the data perspective of the requirements and what information the system needs to deal with. It is also possible to define constraints of the system-of-interest, such as defining a value range limit into an attribute.

This view depends on the Glossary view: the entity names should be present in the glossary. On the other hand, views like "Use Cases" and "Functional Requirements" might show dependencies to the Structural view.

Use Cases view. Use Cases view supports the popular technique to specify complex functional requirements. Despite its simplicity, use cases specification technique provides a practical way to describe and analyze part of the behavior of the system-of-interest based on the concept of usages scenarios.

This view depends on the Actors and Structural views because each use case needs to refer at least an actor (defined in the Actors view) and an entity (defined in the Structural view).

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 more traditional documentation approaches. Some advantages of this view are to provide means to: (1) classify individual requirements according to their type (e.g., 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.

2.3 Other Views

You may consider extending this template by including other views, namely Business Rules view and Non-Functional Requirements View. **Business Rules view** defines 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 attribute value, you can define 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** may makes sense in cases that the system is complex or in those situations that a detailed level of specification is required, by defining quality properties or quality attributes, in what concerns such as security, privacy, performance, portability.

3. RSL-IL EXCEL TEMPLATE

3.1 Configuration and Index Sheets

3.1.1 home Sheet

The home sheet defines the index of the template, where each index line has one link to a specific sheet. Using the home sheet is easier to see and to access directly to the sheets that exist in the template.

3.1.2 config Sheet

The config sheet lists the configuration names that are used to classify the constructs used in the SRS template. These configuration names are referred in any other sheet. Below, during the explanation of the business level sheets and system level sheets, we explain most of the classes and configuration names that are identified in this sheet.

3.1.3 rslil home Sheet

The rslil-home sheet defines the system-under-study with all its sub-systems. This sheet allows to define the decomposition of the system in sub-systems as well as to define relationships between the system-under-study and other third-party systems.

3.2 Business Level Sheets

3.2.1 Glossary View (rslil.glossary sheet)

This sheet includes the terms defined in the SRS; each term is defined by the following information:

Property Name	Description	Type/Values
Id	Unique identifier of the term.	String
Name	Name of the term.	String
Type	Type of the term.	stakeholder, actor, entity, architectural, stakeholder.actor, stakeholder.entity, stakeholder.actor.entity
Description	Brief description of the term.	String
POS	Part-of-speech; define if the term is a noun, adverb or verb.	adjective, adverb, noun, verb
Synset	Set of synonyms; can be used together with an external database such as Wordnet (that may define relationships with other words).	String
DependsOn	Id of other term(s) used in the glossary to establish a relation with the current term.	<Term Ids>
DependsOn Type	Type of the relation of the chosen term above.	antonym, hypernym, synonym.

3.2.2 Stakeholders View (rslil.stakeholders sheet)

This sheet includes the stakeholders defined in the SRS; each stakeholder is defined by the following information:

Property Name	Description	Type/Values
Id	Unique Identifier of the Stakeholder.	String
Name	Name of the stakeholder.	String
Type	Type of the stakeholder.	group.organization, group.business_unit, group.team, individual.person, individual.external_system
Category	Category of the stakeholder.	business, business.customer, business.customer.sponsor, business.user, business.user.direct, business.user.indirect, business.advisor, business.advisor.expert, business.advisor.trainer, business.advisor.regulatory business.system, technical
Description	The purpose and/or what the stakeholder do with the system.	String
DependsOn	The Id of other stakeholder that aggregates the stakeholder of the current line.	<Ids>
DependsOn Type	Type of the relation of the chosen stakeholder above.	is-part-of

3.2.3 Goals View (rslil.structural sheet)

This sheet includes the goals defined in the SRS; each goal is defined by the following information:

Property Name	Description	Type/Values
Id	Unique identifier of the goal	String
Description	Describe the goal	String
Source (Stakeholder)	The Stakeholder who defined the goal.	<Stakeholder Ids>
Priority	The level of importance and priority of the goal.	very-low, low, medium, high, very-high
DependsOn	Here you may define an existent goal in this sheet. The current line goal dependents on the identified goal(s).	<Goal Ids>
DependsOn Type	Type of dependency of the current line goal with other goal(s).	requires, supports, obstructs, conflicts, identical
ComposedBy	Here we can write one or more existent goals in this sheet. The current line goal is decomposed by all the chosen goals.	<Goal Ids>
ComposedBy Type	The ComposedBy relationship may have two types: The type “and” which means that all the goals in the “ComposedBy” 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

3.3 System Level Sheets

3.3.1 Actors View (rslil.actors sheet)

This sheet includes the actors defined in the SRS; each actor is defined by the following information:

Property Name	Description	Type/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.	String
Name	At the blue line we define the name of the system and at the others lines we define the name of the actors.	String
Type	Type of the actor.	User, external_system, timer
Description	Only used for the white lines related to the actors and define the responsibilities of the actor.	String
Source (Stakeholder)	The stakeholder from which this actor derives.	<Stakeholder Id>
DependsOn	The current line actor depends on the identified actor (s).	<Actor Ids>
DependsOn Type	Type of dependency of the current line actor with other actor(s).	is-part-of, is-specialized-by

3.3.2 Structural View (rslil.structural sheet)

This sheet includes the (data-) entities defined in the SRS; each entity defines a set of attributes with the following information:

Property Name	Description	Type/Values
Id	Unique identifier of the entity.	String
Name	At the blue line you should write the name of the entities and at the others lines you should write the name of the attributes.	String
Description	At the blue line you should write the description of the entity and at the other lines you should write the description of the attributes.	String
Type	This column is only used at the white lines (for attributes), and it defines the type of the attributes	boolean, integer, decimal, currency, date, time, date time, enumeration, text, regex, ref, image, video, map
Default value	This column is only used at the white lines (for attributes) and it defines the default value of the attribute.	String
References to	Id of other entity(ies) if needed. The chosen entity(ies) will have some type of association with the current line entity.	<Entity Ids>
Multiplicity Type	If relevant You need to select the multiplicity of the association defined at “References to”.	0, 1, 0..1, *, etc.

3.3.3 Use Cases View (rslil.usecases sheet)

This sheet includes the use cases defined in the SRS; each use case is defined by the following information (as well as a set of scenarios):

Property Name	Description	Type/Values
Id	Use case unique identifier.	String
Name	Name of the use case.	String
Type	Type of the use case.	manage-entity, create-entity, search, browse, report, etc. (TBD)
Description	Description of the use case.	String
Priority	The level of importance and priority of the goal.	very-low, low, medium, high, very-high
Accomplished goals	Goals accomplished by the use case.	<Goal Ids>
Functional Requirements	Ids of the functional requirements that the use case fulfill and that exist at the Requirements View (rslil.requirements)	<Requirement Ids>
Actor initiates	Ids of the actors that can start the use case and that exist at the Actors View (rslil.actors).	<Actor Id>
Actors participates	Here we need to write the ids of the actors that can participate in the use case.	<Actor Ids>
Pre-conditions	Conditions needed to start the use case.	String
Pos-conditions	Conditions needed to fulfill the use case successfully.	String
Include	Ids of the use cases that are included in the current line use case.	<Use Case Ids>
Extended By	Ids of the use cases that extend the current line use case and the related extension points.	<Use Case Ids>

3.3.4 Scenarios View (rslil.usecases.Sn sheet)

This view is part of the Use Cases View. Each Scenario is attached to a use cases which may have several Scenarios. Each use case' scenario describes a set of steps, with the following information:

Property Name	Description	Type/Values
Id	Id of the scenario.	String
Name	Name of the scenario.	String
Type	Type of the scenario.	MainScenario, AlternativeScenario, ExceptionScenario
Sequential	The steps of the scenario can be sequential or not.	TRUE, FALSE
Step ID	Number of the current step.	String
Step Type	It defines the type of the step.	ActorPrepareData, ActorCallSystem, SystemExecutes, SystemReturnResult, ActorPrepareData.ActorCallSystem, SystemExecutes.SystemReturnResult
Description	Description of what happen at the current step.	String
Actor	Actor that executes the current step.	<Actor Id>
Pre-conditions	Conditions that need to be fulfilled to start the step.	String
Pos-conditions	Conditions that need to be fulfilled to finish the step successfully.	String
Next Step	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.)	String

3.3.5 Requirements View (rslil.requirements sheet)

This sheet includes the other types of requirements defined in the SRS; each requirement is defined by the following information:

Property Name	Description	Type/Values
Id	Id of the requirement.	String
Name	Name of the requirement.	String
Description	Description of the requirement.	String
Modality Type	Modality type that the requirement must fulfill.	obligation, permission, prohibition
Action Type	Action Verb used by the requirement.	manage-entity, create-entity, search, browse, report, etc. (TBD)
Proposer	Stakeholder who created or want the current functional requirement.	<Stakeholder Id>
Priority	The level of importance and priority of the requirement.	very-low, low, medium, high, very-high

4. EXAMPLE – THE BILLING SYSTEM

The companion annex of this document includes a very simple (and not complete) application example of this RSL-IL Excel template. This example is based on a fictions information system, named "the Billing system". It should manage the tracking of billable products and services delivered to customers, and so, it should support the management of customers, products, invoices, payments, etc.

This annex specifies the goals, stakeholders, actors, entities, use cases, requirements and scenarios of the Billing System, providing practical and simple examples. Figure 3 depicts how the Billing System is structured and its main actors and use cases.

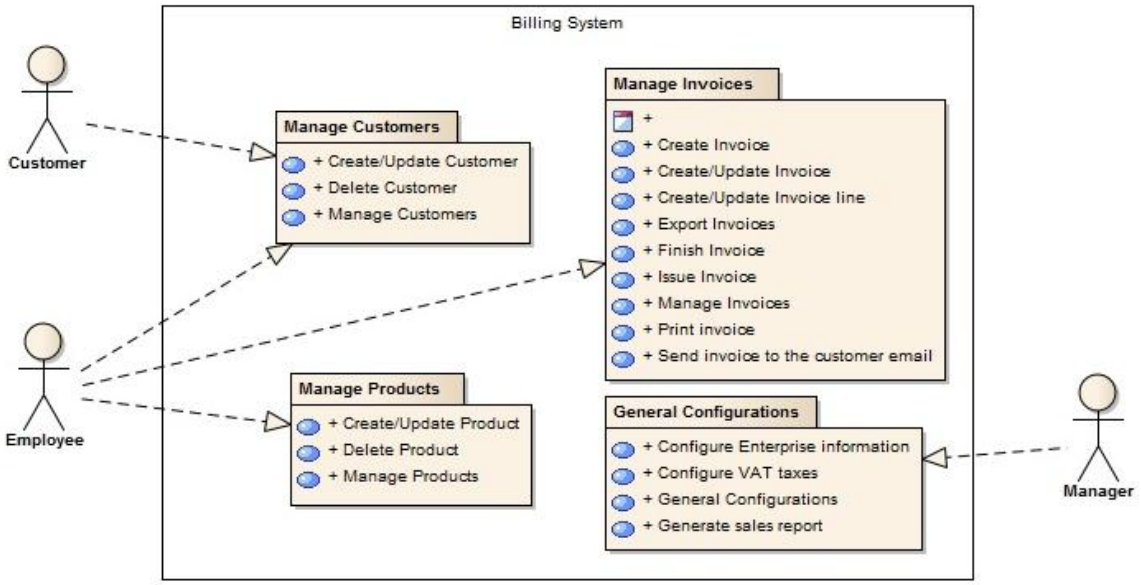

Figure 3: Viewpoint dependencies and respective abstraction levels.

5. CONFIGURATION GUIDELINES

The RSL-IL Excel template can be used as-is by any organization. However, it can be adapted and configured to attend the specific needs of each organization. In these specific situations you may customize this Excel template and in particular its config sheet.

The classifiers used to define the types of glossary, stakeholders, goals, etc. are defined/configured in rslil.config sheet. If you need to update the values of these classifiers change there those values. After that 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, select the set of these cells and, using the option "Validation of Data", it is possible to make the connection between what was defined in rslil.config sheet with the other sheets that use those classifiers. More information to help to implement this can be found for example at https://support.office.com/en-ca/article/Define-and-use-names-in-formulas-4d0f13ac-53b7-422e-afd2-abd7ff379c64.

Regarding the home sheet, it contains contents that are dynamic, so you can add additional sheets and the index is updated automatically. More information to help to manage this index feature can be found at http://www.ozgrid.com/VBA/sheet-index.htm.

REFERENCES

1. Ferreira, D., Silva, A. R., 2012. RSLingo: An Information Extraction Approach toward Formal Requirements Specifications. In: Proc. of the 2nd Int. Workshop on Model-Driven Requirements Engineering (MoDRE 2012), IEEE CS.
2. Ferreira, D., Silva, A. R., 2013. RSL-IL: An Interlingua for Formally Documenting Requirements. In: Proc. of the of Third IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE 2013), IEEE CS.
3. Ferreira, D., Silva, A. R., 2013. RSL-PL: A Linguistic Pattern Language for Documenting Software Requirements. In: Proc. of the of Third International Workshop on Requirements Patterns (RePa 2013), IEEE CS.
4. Silva, A. R., 2014. Quality of Requirements Specifications: A Framework for Automatic Validation of Requirements, in Proceedings of ICEIS'2014 Conference, 2014, SCITEPRESS.
5. Verelst, J., et al., 2013. Identifying Combinatorial Effects in Requirements Engineering, Proceedings of Third Enterprise Engineering Working Conference (EEWC 2013), Advances in Enterprise Engineering, LNBIP, Springer.
6. Silva, A. R., et al., 2014. Towards a System Requirements Specification Template that Minimizes Combinatorial Effects, Proceedings of QUATIC'2014 Conference, IEEE CS.

Publication date: October 2015