Departament de Llenguatges i Sistemes Informatics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Master in Computing

Master of Science Thesis

INCLUDING FUNCTIONAL AND NON-TECHNICAL REQUIREMENTS IN A SOFTWARE REQUIREMENT PATTERNS CATALOGUE

Cristina Palomares Bonache

Advisor/s: Xavier Franch Gutiérrez
Carme Quer Bosor

23/06/2011

Meta-Document section

Table of contents

Table of figures

Table of tables

Master thesis background

This master thesis is the result of my work on the research group GESSI (*Group of Software Engineering for Information Systems)* [1] during the last months. This group, that belongs to the department ESSI (*Enginyeria de Serveis i Sistemes de la Infomació)* [2] of the UPC (*Universitat Politècnica de Catalunya* BarcelonaTech) [3], is working on several research lines, such as Model-Driven Engineering (MDE) [4], Software Quality (SQ) [5], i* modeling [6] and Service Monitoring [7]. Among them, I have been working on the Requirements Engineering [8] line, which is already a consolidated research line in the group.

I am principally working in Requirements Engineering area, focused on the development of a framework called PABRE (PAtterns Based Requirements Elicitation) [9]. The idea behind this framework is to provide requirements engineers or system analysts with different artifacts, which trough the use of requirement patterns, make easier the requirements elicitation stage and improve the traceability of the requirements and its corresponding documentation. For more information of my previous work in this direction see [10, 11, 43].

Preface

Due to the increasing pressure to achieve software with a high quality in the shortest time, from many years ago different reuse techniques have been introduced in the software development process [12]. These techniques facilitate the design and development of components in order to be reused in other applications, reducing the development time, improving the product quality and being more competitive on costs.

Ideally, the reuse consists on using knowledge in its most abstract form [13]. The requirements represent the most abstract level of knowledge in software projects, in a way that by reusing requirements the level of abstraction of reusable items increases.

Because of this, the concept of reuse in the requirements engineering stage is accepted as a desirable goal from years ago [14-19]. There have been several techniques proposed to reuse knowledge during Requirements Engineering, but it seems that no concrete proposal has achieved a good acceptance. Particularly, what appears to be missing in the literature is the form that requirements should have so that reuse can be achieved in requirements elicitation as part of regular projects. In the field of software engineering, patterns were created firstly to solve problems identified during the design of software systems, and from many years ago, many research and development efforts in software engineering have focused on the identification and use of such patterns [20].

On the other hand, according to Glinz [21], requirements set the boundaries of an important dimension of the software products, and because of this their final quality depends on their Software Requirement Specifications (SRS) (also known as requirements books). However, there are some problems that often exist in SRSs: usually requirements are stated in an ambiguous, incomplete and inconsistent manner, and generally they are expressed in an unsystematic way [22].

Then, with the motivation of requirements reuse and with the objective of solving the above existent problems, among all the proposed techniques to achieve reuse, the GESSI research group has created its own approach of Software Requirement Patterns (SRPs) [23-25,11] and an entire framework around it [9]. This framework proposes the use of SRPs for reusing knowledge obtained during requirements engineering and improving the quality and validity of SRSs.

Next it is described the assets of the PABRE framework and the motivation of the contributions of the thesis in each one:

- A proposal of a process for reusing knowledge during requirements elicitation stage using a SRPs catalogue. There not yet exists a systematic review [26] looking for the existent approaches and related work on the use of patterns during requirements engineering for reusing knowledge.
- A set of non-functional SRPs. Non-functional requirements are only one of the types of requirements that can be found in SRSs, so in order to achieve all the benefits that requirement patterns can provide, it is necessary that requirement patterns embrace also functional and non-technical requirements. Taking into account that the use of requirement patterns during Requirements Engineering stage can make easier this stage and improve its results, it seems of great relevance that requirement patterns can be applied to all type of requirements.
- A metamodel of SRPs. This metamodel has been constructed taking into account non-functional requirements present in the SRSs corresponding to 6 real projects. However, it is necessary to check if it is suitable for functional and non-technical requirement patterns.
- Two tools [9, 43] that support the construction, use and evolution of the SRPs catalogue.

Objectives

Taking into account the drawbacks presented above for each asset in the PABRE framework, the objectives of this thesis are:

1. Do a systematic review of the existent published works on reuse in Requirements Engineering stage, particularly on the use of patterns to achieve the reuse of requirements during Requirements Engineering.
2. Construction of a complete set of non-technical SRP that can be obtained from the Software Requirement Specifications (SRSs) corresponding to 6 real projects.
3. Study of the Content Management System domain and construction of some examples of functional SRP for this domain from the same 6 SRSs.
4. Check the validity of the current SRP metamodel for its suitability for non-technical and functional SRPs.
5. Validate the structure of SRPs (as it is the base of this thesis) and construct a survey which will be used to know what requirements engineers think about the usability of SRP catalogues in real projects in their different enterprises or organizations and if it will be applicable or not.

Master thesis structure

This Master Thesis is structured in four parts:

- The first part is a systematic review of how patterns are applied in Requirements Engineering, and concretely of how patterns are applied for reusing knowledge acquired during this stage, and specifically during requirements elicitation. It is divided into three sections: first, the planning of the review; second, how the review process was conducted; and third, the analysis of the relevant works found.
- The second part is the main part of the document and is divided into four sections: first, an introduction to the PABRE framework; second, the description of the construction of a first set of non-technical Software Requirement Patterns (SRP); third, the presentation of the construction of some functional SRPs for an specific software domain (specifically, the Content Management System domain); and fourth, the revision of the current Non-Functional (NF) SRP metamodel to check if it is convenient to the Non-Technical (NT) and Functional (F) SRPs, and the possible changes that may be necessary to adapt this metamodel to the 3 types (NF, NT and F).
- The third part is related to the validation. Specifically, it is related with two different lines of validation: first, the validity of the structure of SRPs (as it is the base used for this thesis); and second, the construction of a survey for requirement engineers from different companies and organizations with the aim to analyze how useful an SRP catalogue and its structure would be for their work.
- Finally, the fourth part contains the final remarks of this thesis and is divided into two sections: first, the conclusions; and second, the ongoing and future work.

Part I

Systematic Review:

Patterns to Reuse

in

Requirement Engineering

1. Research Method for the Systematic Review

1.1. Motivation

In order to prove the usefulness of the work done by GESSI on the use of software requirement patterns (SRP) for reusing knowledge in requirements engineering and improving the quality and validity of Software Requirement Specifications (SRSs), we present in this part of the Master Thesis a study of research works that can be found in the most important databases (IEEE Xplore,....). Instead of using an arbitrary methodology for finding these documents, we decided to do a systematic review. In a systematic review the used methodology drives to an accurate and objective selection of the documents to study and ensures a more accurate and reliable State of the Art. The importance of the used methodology is critical, since unless a literature review is thorough and fair, it is of little scientific value.

This systematic review is based on B. Kitchenham's methodology [26]. She proposes some specific systematic review guidelines for software engineering researchers. Her proposal defines a systematic review as follows:

"A systematic review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Systematic reviews aim to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology."

The proposal of Kitchenham is based on similar guidelines for medical researchers, with the aim of reusing the knowledge and experience of a well-known and consolidated research area. Her main intention with the guidelines is to introduce the concept of rigorous reviews of current empirical evidence to the software engineering community.

Some features that differentiate a systematic review from a conventional literature review are:

- *Definition of a review protocol*. A review protocol specifies different points that will be needed while conducting the review. The most important ones are the research questions being addressed and the methods that will be used to perform the review.
- *Definition of a search strategy*. A search strategy aims at detecting as much of the relevant literature as possible and can be focused on concrete sources (the most important conferences and journals, for instance) or can be broader (taking into account general sources of knowledge or databases: GoogleScholar [27], IEEE Xplore [28], etc.).
- *Documented searches*, so that readers can assess its rigor and completeness. One of the important characteristics of systematic reviews is that they could be replicated, so it is very important documenting all the process.
- *Explicit inclusion and exclusion criteria* to assess each potential primary study (a systematic review is a secondary study based on primary studies).

- Systematic reviews specify the information to be obtained from each primary study, including quality criteria by which to evaluate each primary study.

1.2. The review process

As mentioned before, one of the key features of systematic reviews is the definition of a review protocol that specifies the methods that will be used to perform the review. As stated in [26], *"A review protocol specifies the methods that will be used to undertake a specific systematic review. A pre-defined protocol is necessary to reduce the possibility researcher bias."* Following Kitchenham's guideline, we should undertake this State of the Art accordingly to the following steps:

1. **Planning the review (Section 2)**
   1.1. Identification of the need for a review
   1.2. Development of a review protocol
2. **Conducting the review (Section 3)**
   2.1. Conduct the search
   2.2. Selection of primary studies
   2.3. Quality assessment study
   2.4. Data extraction
   2.5. Data synthesis
3. **Reporting the review (Section 4 & 5)**

Although these stages seem to follow a sequential mode, it's important to clarify that the development of these stages may involve iteration.

2. Planning the Review

The planning of the review consists of 2 basic steps, which are described in the following subsections:

1. Identification of the need for a review.
2. Development of a review protocol.

2.1. Identification of the need for a review

As specified in the guideline, the first step before undertaking a systematic review is to search for existing systematic reviews of the subject that we want to study. Then, prior to undertaking a systematic review, we should ensure that a systematic review is necessary (i.e. a systematic review that covers our subject of study does not already exist).

There is no procedure defined in [26] in order to search for these systematic reviews in an accurate manner. Nevertheless, we present here the procedure that was followed for retrieving existing reviews. To increase the number of results, we did not focus just on systematic reviews, but on reviews and states of the art, regardless the methodology followed for developing them.

The following databases were searched for reviews:

- Google Scholar [27]
- IEEE Xplore [28]
- ACM [29]
- Springer [30]
- ISI Web of Knowledge [31]
- Science Direct [32]

We were interested in searching reviews that contained the use of patterns to reuse knowledge in Requirements Engineering. Given that this is a subject too specific, we searched for reviews of reuse during Requirements Engineering in general (taking into account that patterns are a particular way of reuse). The following keywords were used to search in these sources. In order to reduce the noise of the results, the keywords were applied just to the title of the papers. We consider that because of the nature of this kind of papers, they should have a relevant name in the title itself.

1. reus* **AND** requirement* **AND** review
2. reus* **AND** requirement* **AND** “state of the art”

The queries didn’t produce any result in any of the above sources. Then we concluded that no review or State of the Art has been reported to the significant listed databases. To fill this gap, we needed a more extended research to cover densely the current State of the Art of reuse of requirements, and specifically in the use of patterns during this reuse. Therefore, the development of a systematic review was required.

2.2. Development of a review protocol

A review protocol specifies the methods that are to be used to undertake a specific systematic review. A predefined protocol is necessary to reduce the possible researcher bias.

A review protocol is composed of the following elements:

- The research question/s that the review is intended to answer.
- The strategy that will be used to search for primary studies.
- Study selection criteria and procedures.
- Data extraction strategy.
- Synthesis of the extracted data.
- Project timetable.

2.2.1. The research questions

The first step of the review is the development of the research questions. It is intended by this systematic review to answer questions that identify and/or scope future research activities. During the development of this work, we will focus on those approaches related to Requirements Engineering that deal in some way with patterns, putting special attention on those proposals that use patterns to reuse requirements.

The research questions of this systematic review are then:

1. How patterns are used during requirements engineering?
2. Are there specific proposals that use patterns to reuse requirements?
3. Among the previous proposals, do any of them propose a well-established set of patterns (i.e. a catalogue)?

2.2.2. The search strategy

In this systematic review, in order to cover a broader set of sources, the search was done directly over well-known databases that embrace a lot of different conferences, journals, etc.

After evaluating different options, we decided to discard ISI Web of Knowledge [31] and Google Scholar [27]: the first one because the sources that it embraces also appear in other databases and the second one because contains a lot of different topics and produces a lot of noise.

The selected databases to search in to perform this systematic review were:

IEEE Xplore [28] 
Description: **IEEE Xplore is a database produced by IEEE which includes the full publications of IEEE (Institute of Electric and Electronic Engineers) and IET (Institution of Engineering and Technology). It contains journals and proceedings from both institutions since 1988.**

ACM [29] 
Description: **It has complete access to the publications of ACM (Association for Computing Machinery), for both journals and proceedings.**

Springer [30] 
Description: **Database of journals and books published by Springer-Verlag and other editors, such as Kluwer. It includes 500 multidisciplinary journals, and also 1800 monographs of the**

collection Lecture Notes in Computer Science (LNCS), all of them specialized in computer science.

Science Direct [32]

Description:

It has complete access to the publications of ElSevier editorial, which includes a lot of journals and proceedings, with the particularity that its content is open (not is necessary having a subscription to access to the full text of the published works).

To find out how patterns are involved in Requirements Engineering, we decided to search proposals that in some way are related with requirements or Requirements Engineering and reuse (patterns are always associated to the reuse concept, so we preferred to look for reuse instead of patterns or templates because the term pattern or template is not always used when talking about these artifacts). Then, the keywords used were:

- Reuse or Reusability or Reusing (i.e. reus*)
- Requirement, Requirements, Requirements Engineering (i.e. requirement*)

We decided to search these keywords in Title and Abstract fields of documents, as they usually contain the most representative information of a document.

Then, our string search was:

reus* in (Title **or** Abstract) **AND** requirement* in (Title **or** Abstract)

The search and data extraction strategy used to find the primary studies was:

1. *Search in the selected databases*. The string search is used into the selected databases and the obtained results are exported to RefWorks [33], a reference manager.
2. *Delete duplicates and non-relevant results*. RefWorks is used to remove results that are duplicated and also those ones that are not relevant (for instance those results that are the introduction of proceedings).
3. *Selection of works based on their titles*. To try to reduce the number of articles, we rule out the ones that evidently are out of the scope of the systematic review.
4. *Selection of works based on their abstracts.* Many times the titles are confusing or not representative enough. Reading the abstract helps to refine the selection done in step 3.
5. *Selection of works based on an overview of the entire work*. Each work is skimmed to be sure before the previous stage that it is relevant for the systematic review.
6. *Selection of works based on their full texts*. Only the selected works after step 5 are read in depth. The deep reading of the article is made marking the relevant parts and annotating comments to ensure that future readings will take less time. During this step a template related with keywords and other important issues is filled in order to classify and facilitates the future data extraction and synthesis for those works that are selected in this step.

7. *Mark work relevance*. As final step, we annotate a personal view of the article, pointing out the benefits, and the detected disadvantages.
8. *Addition of further work.* During the process of the systematic review, other works might be included. This process is performed through obtaining relevant citations of the papers and through further work of the researchers. For all article in this further work, steps 3 to 7 must be carried out.

2.2.3. Study selection criteria and procedures

Study selection criteria are intended to identify those primary studies that provide direct evidence about the research question. In order to reduce the likelihood of bias, selection criteria should be decided during the protocol definition. Specifically, taking into account our strategy defined in the previous section, these study selection criteria must be applied from step 3 through 7 of our strategy.

We define the selection criteria in the following strategy steps:

- *Selection of works based on their titles (step 3).* The objective of this first filter is to identify and remove the noise of the results. After this selection, documents whose scope is not related with Requirements Engineering were removed.
- *Selection of works based on their abstract (step 4).* At this stage, we discarded all works that although being related with Requirements Engineering are definitively out of the scope. This can be due to the fact that Requirements Engineering is not the primary contribution of the paper.
- *Selection of works based on an overview of the entire work (step 5).* At this point we remove by skimming the article all works which their contributions are not relevant or present poor results. In this step we focus exactly on removing those works that are not related with patterns.
- *Selection of articles based on their full text (step 6).* The selection criteria are the same as those ones applied in step 5.

2.2.4. Data extraction strategy & Synthesis of the extracted data

In order to fulfill these two parts of the protocol, we created a template that is fulfilled for ach relevant work of the systemic review (those ones that are selected after step 6). For further details of this template see Table 1.

Topic	Description
Domain of the proposal	Is a proposal that can be used in all domain or it is for a specific domain? Which domain?
Type of requirements to reuse	Functional requirements, Non-functional requirements, some of them, etc.
Object to reuse	Sentences in natural language, Conceptual Models, Use Cases, etc.
Notation used to define the object to reuse in the pattern	Natural Language, Modeling Language (for instance UML, i* or a own one), Formal Language (like Description Logics and Formal Temporal Logic), etc.

Structure in which the object to reuse is contained	If the object to reuse is contained in a pattern (which have further information to facilitate its reuse) or only contain the object to reuse (template). If it is a pattern, whether it contains information about when to use the object (context) or about further stages in the development process.
Object information	What is the information contained in the object to reuse?
Relationships	There is some type of relationships considered among the objects to reuse? What are they?
Reus Object Metamodel	There exists a metamodel to describe the objects to reuse?
Reuse Methodology	There exists a methodology to reuse the objects?
Reuse object construction	There is a methodology to construct the objects to be reused?
Arrangement	If the objects are arranged in a taxonomy or directly in a repository.
Catalogue	There is an established set of reusable objects? It is general or specific for a project or test? Is it a finished set or in evolution?
Catalogue evolution	It is explained in the work how to evolve the catalogue?
Classification	There exists a classification of the objects to reuse?
Classification Metamodel	There exists a metamodel to describe the classification?
Related tools	Does the proposal have related tools? What are their functionalities?
Scope of the proposal	Specification, Documentation, Elicitation, etc.
Proves	Has the proposal been tested in real cases?
Type of study	Empirical, Non-Empirical, Experience Report, Literature Review, etc.

Table 1: Template used to data extraction and synthesis

2.2.5. Project timetable

There is no timetable for this work apart from the deadlines imposed by the Master Thesis. Then, this part of the protocol is omitted.

3. Conducting the Review

This section explains how the review was undertook and the problems encountered in that period.

The conduction of this systematic review was a part of a broader systematic review that some GESSI members did. This broader systematic review was intended to have a global vision of reuse in Requirements Engineering, while this thesis systematic review was only interested in those approaches that use patterns to achieve the reuse. Because of this, the first five steps that the review protocol marks (see section 2.2.2) were conducted by me and three other members of the group in equal parts. The other steps (from 6 to 9) were conducted by me independently from the general systematic review.

3.1. Conducting the search

This stage corresponds to steps 1 and 2 of the review protocol (*Search in Selected Databases* and *Delete duplicates and non-relevant results,* respectively) and was done in March 2011.

The major problem of this stage was that the different databases have different ways of doing queries (the way in which *and's* and *or's* are inserted, if they have or not implemented the * or if it is implicit, etc.). Because of this, we had to look the different manuals of the databases and solve all this doubts. In table 2 we can find the number of results found in the different databases.

After doing all the searches and importing them to Refworks, we proceed to delete duplicates and non-relevant results. Refworks has options that can help to do this. To remove duplicates, we used the option "Remove duplicates" (which show the exact duplicates in the list) and "Remove duplicates almost exact" (which show the references that are very similar, but have little differences: for instance, case sensitive differences, order in the authors list, etc.). Table 3 contains the number of works that were deleted with these two options.

Source	Number of works
IEEE Xplore	1711
Science Direct	250
Springer Link	534
ACM	72
TOTAL	**2567**

Table 2: Database searches results

Non-relevant Cause	Number of works
Duplicates	21
Almost duplicates	26
"Proceedings" in title	18
"Conference" in title	17
"Title" in title	3
"IEEE" in title	3
TOTAL	**88**

Table 3: Number of non-relevant results

Finally, in this stage we also removed the works that are non-relevant, such as introduction to conference proceedings, other introductions and so on. To do this, we searched in the title the terms "proceedings", "conference", "title" and "IEEE" and deleted those ones that were clearly not-relevant. Table 3 also contains these figures.

At the end of this stage we had 2567 – 88 = 2479 works.

3.2. Selection of primary studies & Quality assessment study

This stage corresponds to the steps 3 from 5 (*Selection of articles based on their titles*, *Selection of articles based on their abstracts* and *Selection of articles based on an overview of the entire work*) and also part of the step 6 (*Selection of articles based on their full text*), the part of rejecting those works that are not relevant to this systematic review.

At the beginning of this stage we had 2479 works to start step 3. After reading their titles, we rejected in this step 2081 articles, so we had 398 articles to revise the abstract. We read the abstract of these works and discard 187 articles, and at the end of step 4 we had 211 articles.

During step 5, while having an overview of the entire work, we classified the articles regarding the object that they use to reuse (patterns, ontologies, etc.) and what was the purpose of the reuse (specification, elicitation, etc.). In this step the systematic review of this thesis took a different course from GESSI systematic review. GESSI passed to step 6 all those works that where related with reuse in the elicitation of requirements, while this thesis systematic review passed to step 6 all those articles that were related to patterns.

As we have mentioned in the introduction of this section, these previous five steps of the review were conducted by me and three other members of the group in equal parts. The following steps (from 6 to 8) were conducted by me independently from the general systematic review.

Finally, in step 6 we started with 51 works that seemed to be related to patterns (we discarded 160 articles in step 5). After reading the entire works in detail we discarded 17 works:

- 16 of them because they didn't talk really about patterns in Requirements Engineering (the term pattern appear through the text but it is not relevant or they talk about patterns but not in Requirements Engineering) [SLR1 to SLR16].
- The other one because it was written in Portuguese [SLR17].

Then, we found 34 works that were relevant to this systematic review and we revised them carefully in the following stage. Figure 1 resumes all the results in each step.

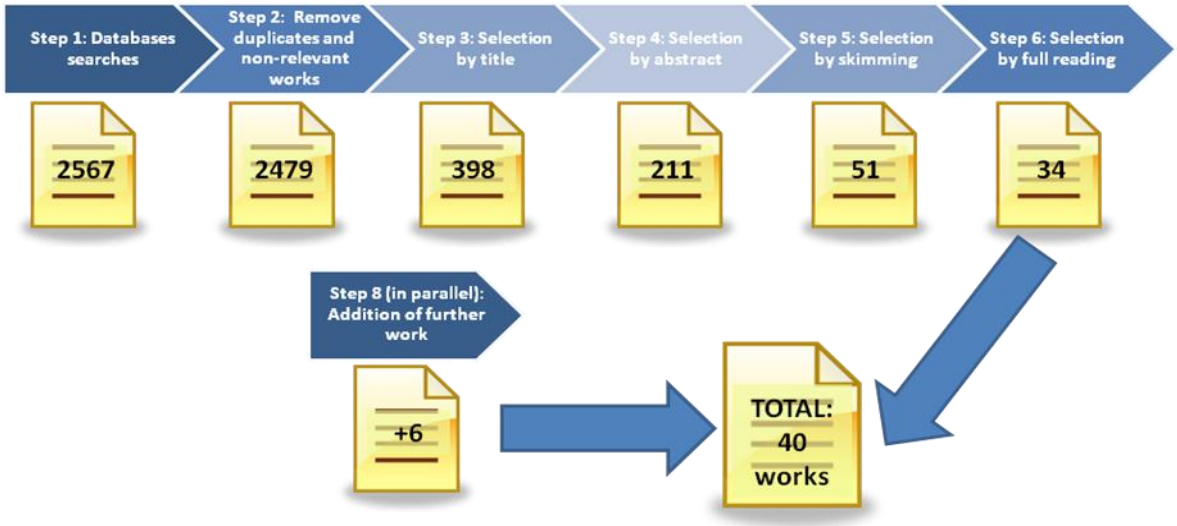


Figure 1: Summary of the works in every step of the review

Also step 8 was done in parallel to all the previous steps. During reading the articles, checking some of the references and so on, 5 works [SLR18 to SLR22] were added because after its revision they were considered relevant to this systematic review. Also, the well-known Withall's book about Software Requirement Patterns was included [SLR23].

3.3. Data extraction & Data synthesis

This stage corresponds to a part of step 6 *Selection of articles based on their full text* (specifically, the part related to fill up the template proposed in Table 1) and step 7 *Mark work relevance*.

This part was conducted with no problems and the analysis of the examined documents is detailed in the following section.

4. Systematic Review Results Analysis

Given the criteria and protocols defined in the previous sections, the synthesis of the 40 retrieved documents is presented. Here we will describe the State of the Art of the different approaches that deal with patterns (sometimes also called templates) in Requirements Engineering, focusing specially on those ones that use patterns to elicit requirements.

As shown in the following figure, we present a 'roadmap' identifying the different groups detected while doing these review. The results of the same authors that talk about the same approach have been considered as an only proposal. The proposals found have been classified:

- First taking into account if they are used to elicit requirements or not,
- And inside these two groups, taking into account the information about the object to be reused. Next, there is one subsection for each of these three analyses.

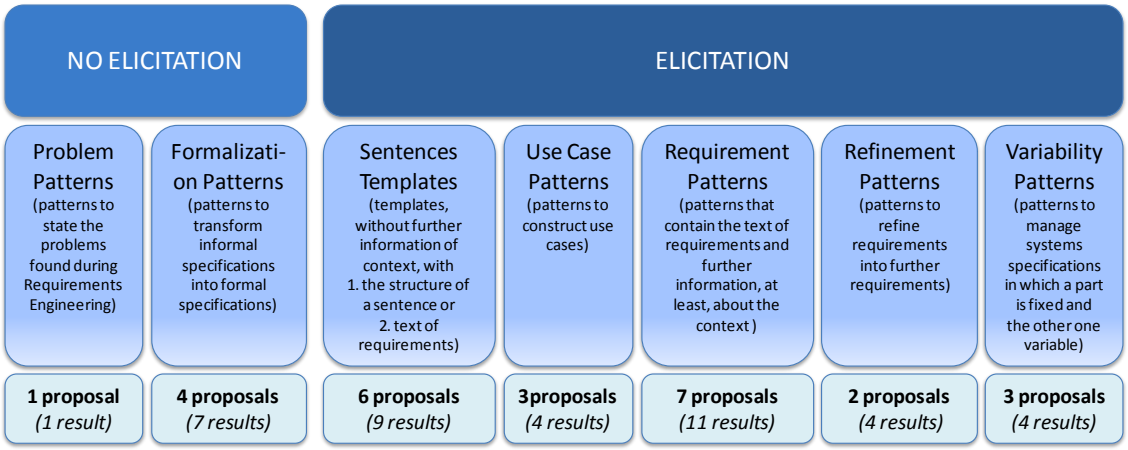


Figure 2: Classification of the systematic review results

The next sections shown how each one of these works has been classified taken into account the reuse object they propose, their catalogues and their reuse process. Finally, the last section contains the analysis of the proposals found.

4.1. Requirements Engineering Reuse Object

Table 4 contains the number of relevant proposals of this systematic review classified according to the characteristics of the object that they propose to reuse.

- The first characteristic states whether the object to be reused may be reused in general independently of the domain of the software addressed or for a specific domain.
- The second and third ones are useful to know what is the object to be reused (sentences in natural language, their structure, conceptual models, uses cases, formalization of requirements) and for what type of requirements the reuse of this object produces (functional, non-functional or both of them).
- The fourth characteristic states what is the notation to define the object to be reused, such as natural language, modeling language

(e.g. UML, i*) or formal language (e.g. formal logic, temporal description logic).

- The fifth and sixth ones are related to the structure in which the object to be reused is presented (a template if it only has the object to be reused or a pattern if it contains the reusable object and further information about when to use it, relationship with other patterns, etc.) and to the existence of a metamodel to define it.
- The seventh and eight characteristics are useful to know if the structure in which the object to be reused is presented also contains information about when to use this object or useful information for further stages (for instance, tests to take into account).
- Finally, the ninth one states if it has been considered the relationships among the objects to be reused.

As general conclusions, we can say that most of the proposals have been created with general purpose, although there are some specific proposals for security requirements due to its relevance when developing a product, and they try to reuse sentences in natural language in most of the cases because it is the most common way to elicit requirements. It is also worth noting that 17 works (almost 65%) don't propose a metamodel to define the object to be reused. Regarding the information that is given with the object to be reused, just over 50% of the proposals detail information about when to use the object reused and relationships among them, but only 19% (5 proposals) state what the implications in further stages when applying this object in a project are.

Characteristics of the object to be reused		No Elicitation	Elicitation
1. Domain	General Purpose	3	13
	Security Requirements	0	4
	Others	2	4
2. Object to be reused	Sentences in Natural Language (SNL)	1	14
	SNL Structure	0	2
	Formalization of Requirements	4	0
	Conceptual Models	0	1
	Use Cases	0	6
	Other diagrams (state diagrams, sequence diagrams, i* models, etc.)	0	2
3. Requirement Type	Functional	2	6
	Non-Functional	0	4
	Functional + Non-Functional	1	11
	Not Stated	2	0
4. Notation of the object to be reused	Natural Language	1	14
	Modeling Language (e.g. UML and i*)	3	2
	Formal Language (e.g. Temporal Logic)	1	1
	Natural and Modeling Languages	0	4
5. Where the object to be reused is contained	Template (normally only the object to be reused)	1	11
	Pattern (object to be reused and further information about the context, relations with other patterns, etc.)	4	10
6. Reusable object metamodel	Yes	1	8
	No / Not Stated	4	13

7. Context information to use the object	Yes	4	10
	No / Not Stated	1	11
8. Information for further stages	Yes	0	5
	No / Not Stated	5	16
9. Relations among reusable objects	Yes	1	13
	No / Not Stated	4	8

Table 4: Systematic review works according the characteristics of the object to be reused

4.2. Requirements Engineering Reuse Objects Catalogue

Another point of view is considering the relevant proposals from the perspective of the catalogue (the set of all reusable objects defined). Table 5 contains the number of relevant proposals of this systematic review classified according to the characteristics of the catalogue of objects to be reused.

- The first characteristic is related to how the set of objects is stored to access them later: in a taxonomy (i.e. some classification is used to their access) or in a repository.
- The second and third ones are useful to know if some classification is considered in the approach (although it may be the case that this classification is not used for accessing the objects) and if there is a metamodel to describe the structure of this classification.
- The fourth and fifth characteristics consider the real existence of a catalogue (either a general one or one created for a specific project or test) and if this catalogue is a finished one or it is in evolution.
- Finally, the sixth characteristic states if the evolution of the catalogue (there exists it or not) is considered as part of the approach or not, and specifically if a method to do this evolution exists.

Characteristics of the catalogue of objects to be reused		No Elicitation	Elicitation
1. Arrangement	Taxonomy (access using a classification)	1	2
	Repository	2	12
	Not Stated	2	7
2. Classification for reusable objects	Yes	1	5
	No / Not Stated	4	16
3. Classification metamodel	Yes	0	2
	No / Not Stated	5	19
4. Existence of a catalogue	General	1	5
	For a specific project or test	0	13
	No / Not Stated	4	3
5. State of the catalogue	Finished	0	1
	In evolution	0	5
	Not Stated / Non-relevant	5	15
6. Consideration of the evolution of the catalogue	Yes (method)	0	2
	Yes (no method)	2	4
	No / Not Stated	3	15

Table 5: Systematic review works according the characteristics of the catalogue of objects to be reused

The first remark to do after this data synthesis is that when taking into account things beyond the reusable object the proposals start to be less specific (in almost all the characteristics, almost 50% or more of the works

don't state anything about the subject). Regarding the arrangement of the objects, 54% of the proposals store the objects in a repository, whereas only the 12% organize them using a taxonomy (which is an easier way to access the objects). Related to the existence of a catalogue, almost 70% of the proposals defines one, but only 23% of them propose a general one (that are those ones that can be reused in real projects). What is more, in only 1 proposal the catalogue is finished, so it seems that the other ones have still work to do. Finally, it is worth noting that despite the fact that 31% of the proposals consider the evolution of their catalogues to have them up-to-date, only 8% have a method to carry out this evolution.

4.3. Requirements Engineering Reuse Process

Finally, in table 6 we can find the proposals organized according to the characteristics for the use of the objects to be reused.

- The first and second characteristics are related to the existence of a methodology to construct the objects to be reused and to use these objects, respectively.
- The third one states if there are specific tools in the proposal to reuse the objects, making easier with these tools the possibility to use the approaches in real projects.
- Finally, the fourth characteristic states if the proposal has been tested in real cases

Characteristics for the use of objects to be reused		No Elicitation	Elicitation
1. Reusable object construction methodology	Yes (method)	0	6
	Yes (informal process)	1	6
	No / Not Stated	4	9
2. Reusable object utilization methodology	Yes (method)	1	9
	Yes (informal process)	2	6
	No / Not Stated	2	6
3. Reuse tools	Yes	2	8
	No / Not Stated	3	13
4. Proposal tested in real cases	Yes	1	8
	No / Not Stated	4	13

Table 6: Systematic review works according the characteristics for the use of objects to be reused

Related to the methodology to construct and use reusable objects, it is worth noting that only the 23% and 38% of the proposals have a formal methodology, respectively. It is also important to highlight that almost 40% of them have a tool to facilitate the reuse in real situations, but only 34% of the proposals have been tested in real cases.

4.4. Analysis of each identified group

The next sections explain for each group presented in figure 2 a general overview and the most relevant aspects.

4.4.1. Proposals not focused on the elicitation of requirements

During developing this review, we found some works that, despite not being related to the elicitation of requirements, could be interesting for knowing how patterns are used in other aspects of Requirements Engineering and also how these patterns are structured. The next sections try to focus a little

bit more in these works, putting special attention in the structure of the patterns used.

4.4.1.1. Problem patterns

The approach in [SLR24] proposes a use of patterns that is very different from other proposals, where the most important point is their structure. As a brief summary, the author try to state what are the recurrent problems in Requirements Engineering and their solution. To do this, they have two different types of assets:

- Inside structure forms (figure 3), which corresponds to the problems that they found to appear recurrently in Requirements Engineering, which corresponds to the notion of pattern.
- Outside structure forms, which expresses the position of a specific project in the problem space.

Name:
Undefined requirements for incomplete domain knowledge.
Context:
In the requirements elicitation process, analysts listen to the user's requirements. There is time remaining for the requirement definition
Problem:
Users cannot define the official requirements. The cause of the problem is that the users have an incomplete understanding of the needs.
Goal-a:
Project manager improves the knowledge level of the project in order to define the requirements within the planned time.
Solution-a1:
The project involves domain experts in the project.
Reason-a1:
Domain experts have enough knowledge to define the requirements.
Solution-a2:
Users learn domain knowledge from the domain experts.
Reason-a2:
If users can obtain related domain knowledge, they will be able to define the requirements.
Result-1:
(Better) Requirements are defined by the users themselves or the domain experts.
Result-2:
(Worse) Requirements can not be defined.

Figure 3: Inside Structure Form example

The relevant asset to this systematic review is the inside structure form. Each one of them contains the following fields: name, context, problem, goal, solution, reason and result. It is important to highlight the attribute goal, which is present in almost all the patterns that we will review, whatever the focus of the pattern is. An important point of this proposal is the metamodel to represent this asset (see figure 4), which despite of being simple, contains all the relevant points of a pattern.

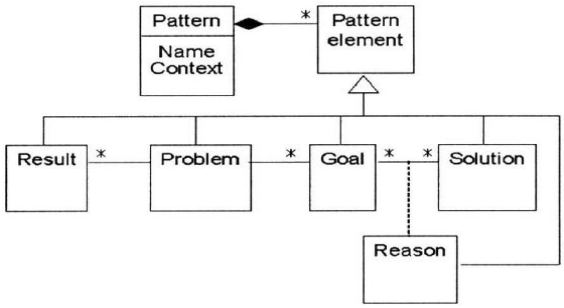


Figure 4: Inside Structure Form metamodel

4.4.1.2. Formalization patterns

The specification of requirements is among the first tasks of any system development. The requirements document is part of the contract between the customer and the system developer, and will be the basis for the acceptance of the final implementation. To avoid later disagreements, it is important that the requirements be stated completely and precisely. In practice, requirements are often stated imprecisely, due to the use of natural language, and this may lead to further problems during the implementation stage. Because of this, some works have proposed the

transformation of informal requirements specifications (normally written in natural language or in other some informal way) to a formal one (specified using some logic language or type of diagram that is understood by a machine).

Table 7 contains the proposals founds and their main general characteristics. We can distinguish two different paradigms in order to formalize requirements: the first one transforms requirements into some model (W.C. Chu et al., V. de F. Sodré et al. and G. Grosz proposals) and the second one transforms them into logic (C. Peper et al. proposal).

		W.C. Chu et al. [SLR19], [SLR20], [SLR25]	**C. Peper et al.** [SLR26], [SLR27]	**V. de F. Sodré et al.** [SLR28]	**G. Grosz** [SLR29]
Object to be reused characteristics	1. Domain	General Purpose	General Purpose	Geographic IS	General Purpose
	2. Object to be reused	Formalization of Requirements	Formalization of Requirements	Formalization of Requirements	Formalization of Requirements
	3. Requirement type	F, NF	------	F	F
	4. Notation of the object to be reused	Modeling Language	Formal Language (Temporal Formal Logic)	Modeling Language (UML Geo Frame)	Modeling Language
	5. Where the object to be reused is contained	Pattern	Pattern	Pattern	Template
	6. Reusable object metamodel	------	------	------	------
	7. Context information to use the object	Yes	Yes	Yes	------
	8. Information for further stages	------	------	------	------
	9. Relations among reusable objects	------	------	Yes	------
Catalogue characteristics	1. Arrangement	Repository	Repository	Taxonomy	------
	2. Classification for reusable objects	------	------	Yes	------
	3. Classification metamodel	------	------	------	------
	4. Existence of a catalogue	------	------	------	------
	5. State of the catalogue	------	------	------	------
	6. Consideration of the evolution of the catalogue	------	Yes (no method)	Yes (no method)	------
Use characteristics	1. Reusable object construction methodology	Yes (informal process)	------	------	------
	2. Reusable object utilization methodology	Yes (informal process)	------	Yes (informal process)	Yes (method)
	3. Reuse tools	Yes (formalize req. books previously written using patterns)	------	Yes (use patterns creating UML-Geo Frames models previously specified)	------
	4. Proposal tested in real cases	Yes	------	------	------

Table 7: Summary of the systematic review works found related to the formalization of requirements

Both W.C. Chu et al. [SLR19] [SLR20] [SLR25] and G. Grosz [SLR29] proposals are useful to formalize requirements in any domain, although the last one only takes into account functional requirements in this formalization:

- W.C. Chu et al. proposes the MORE (Model-based Object-oriented Requirements Engineering) framework (figure 5) to model informal requirements documents using patterns and, after constructing this model, transform it into a XML-based Unified Model (XUM). Due to the nature of Requirement Object Models (which correspond to transformation patterns), they also embraces non-functional requirements. The principal problem of this proposal is that although it seems interesting and has been proven in real cases, the structure of the pattern is not clear.

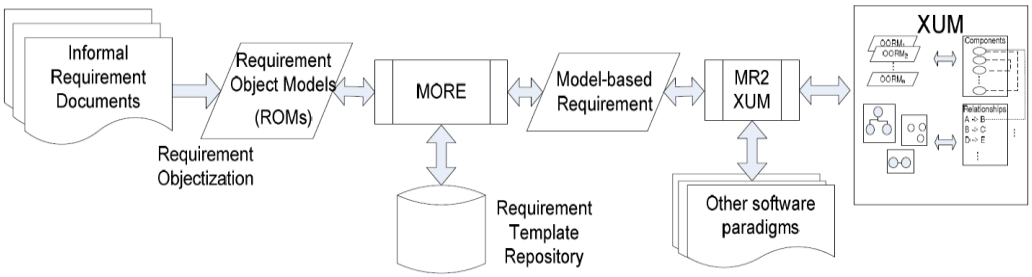


Figure 5: Model-based Object-oriented Requirement Engineering (MORE) framework

- G. Grosz presents the matching of functional requirements using the triplet <situation, decision, action>. After this matching, some templates (see figure 6 for an example) are used to transform this triplet, specially the action element, into its own descriptive schemas, which are the result of applying a template to a corresponding triplet (figure 7).

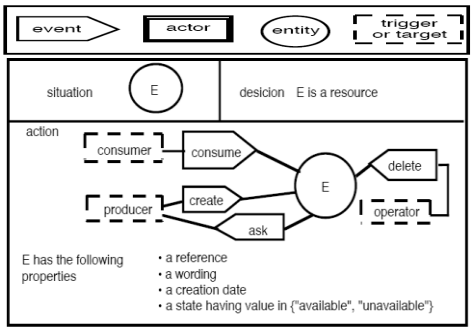


Figure 6: G. Grosz template example

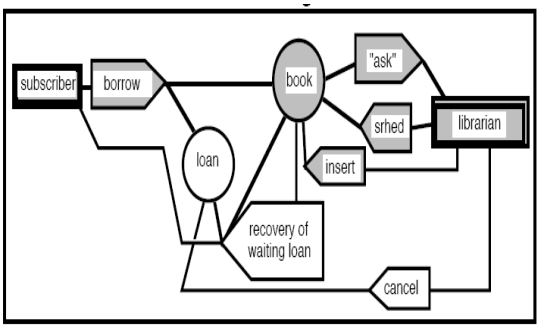


Figure 7: G. Grosz application template example

Also from the point of view of transforming requirements into models, we found the V. de F. Sodré et al. proposal [SLR28]. It is focused on geographic information systems, and basically it has a set of patterns (catalogue) that the requirements engineer can consult after requirements elicitation in order to construct UML-GeoFrame models, based on stereotyped UML diagrams (figure 8) that represent these requirements. The interesting point of this proposal is the structure of a pattern (see figure 9 for an example), highlighting the fields problem, context (to know when to use the pattern) and solution, following the well-known context-problem-solution paradigm of patterns, and also the related patterns field, which will appear in further

proposals of patterns, and states those patterns that may be relevant when using this one.

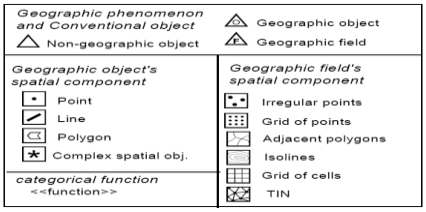
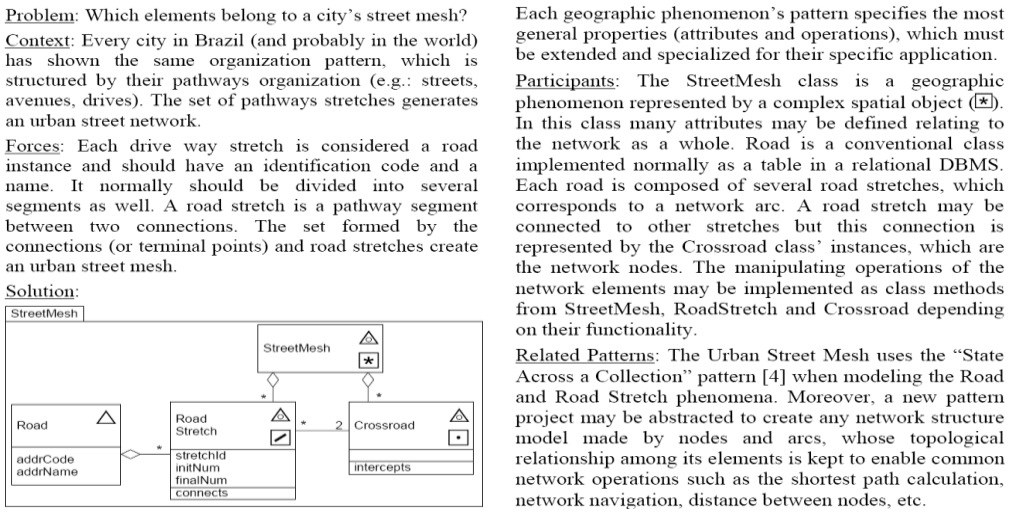


Figure 8: UML-GeoFrame model stereotypes



Problem: Which elements belong to a city's street mesh?

Context: Every city in Brazil (and probably in the world) has shown the same organization pattern, which is structured by their pathways organization (e.g.: streets, avenues, drives). The set of pathways stretches generates an urban street network.

Forces: Each drive way stretch is considered a road instance and should have an identification code and a name. It normally should be divided into several segments as well. A road stretch is a pathway segment between two connections. The set formed by the connections (or terminal points) and road stretches create an urban street mesh.

Solution:

Each geographic phenomenon's pattern specifies the most general properties (attributes and operations), which must be extended and specialized for their specific application.

Participants: The StreetMesh class is a geographic phenomenon represented by a complex spatial object (*). In this class many attributes may be defined relating to the network as a whole. Road is a conventional class implemented normally as a table in a relational DBMS. Each road is composed of several road stretches, which corresponds to a network arc. A road stretch may be connected to other stretches but this connection is represented by the Crossroad class' instances, which are the network nodes. The manipulating operations of the network elements may be implemented as class methods from StreetMesh, RoadStretch and Crossroad depending on their functionality.

Related Patterns: The Urban Street Mesh uses the "State Across a Collection" pattern [4] when modeling the Road and Road Stretch phenomena. Moreover, a new pattern project may be abstracted to create any network structure model made by nodes and arcs, whose topological relationship among its elements is kept to enable common network operations such as the shortest path calculation, network navigation, distance between nodes, etc.

Figure 9: UML-GeoFrame pattern example

In the group of proposals that use patterns to formalize requirements to formal logic, we found the Peper et al. proposal [SLR27]. Their object is to transform natural language requirements for a real-time system to real-time temporal logic using patterns similar to the one in figure 10. The remarkable parts of this proposal compared to the previous ones are: the notion of parameter (*p, q, T_1, T_2* and *T_3* in the example) is present; the semantic properties field includes some special cases that could be relevant when using this pattern. As in other patterns, the goal field (intention in the example) is present.

Name	LazyReaction (p, q, T_1, T_2, T_3)
Intention	Applicable if a reaction is not immediate, but contains some delay.
Example	$roomUsed \leftrightarrow_{Df} (\blacksquare_{\leq T_1} \neg roomEmpty \vee \langle *[\blacksquare_{\leq T_1+T_2} \neg roomEmpty] \Rightarrow \rangle \blacksquare \blacklozenge_{\leq T_3} roomEmpty)$
	A room is currently in use (*roomUsed*) iff one of the following holds: – during the past T_1 time units, a person has continuously been in the room; – there has already been a time span of length $T_1 + T_2$ such that a person has continuously been in the room, and since this has been the case for the last time, periods where the room was empty were not longer than or equal to T_3.
Definition	$p \leftrightarrow_{Df} \blacksquare_{\leq T_1} q \vee \langle *[\blacksquare_{\leq T_1+T_2} q] \Rightarrow \rangle \blacksquare \blacklozenge_{\leq T_3} q$
	The predicate p holds iff one of the following statements holds: – during the past T_1 time units, q has continuously been true; – there has already been a time span of length $T_1 + T_2$ such that q has continuously been in the room, and since this has been the case for the last time, periods where q was false were not longer than or equal to T_3.
Semantic properties	$T_1 = T_2 = T_3 = 0 \rightarrow p \leftrightarrow q$; $\quad T_2 = T_3 = 0 \rightarrow p \leftrightarrow \blacksquare_{\leq T_1} q$ $T_2 = 0 \rightarrow p \leftrightarrow \langle *[\blacksquare_{\leq T_1} q] \Rightarrow \rangle \blacksquare \blacklozenge_{\leq T_3} q$; $\quad T_1 = T_2 = 0 \rightarrow p \leftrightarrow \blacklozenge_{\leq T_3} q$

Figure 10: C. Peper et al. temporal logic formalization pattern

4.4.2. Proposals focused on the elicitation of requirements

In the last years, elicitation of requirements during Requirements Engineering and also its final result, the requirements specification documents, has been a focus to deal with. Due to the importance that requirements specifications have in further development stages, it has great importance that these specifications are defined precisely and also that state all the needed requirements for the new system. Among the different ways to achieve these goals, the reuse of requirements, specifically with patterns, holds a prominent position. The next sections focus deeply in the different patterns that have been used to reuse requirements, putting special attention in the structure of the patterns used and also if there exists or not a formal method to reuse them, a well-defined set of patterns and in its viability of being used in real projects.

4.4.2.1. Sentences Templates

The first approach to reuse requirements in their simplest form is to reuse directly the text of these requirements, which is stated using sentences in natural language (SNL). The basic idea of this form of reuse is that the requirements engineers have a set of templates of SNLs, without further information about when to reuse them, and they have to look for those templates that could be useful for a particular problem. We can distinguish two different ways of reusing these SNLs: the first one is related to reuse the structure of these texts, having templates stating what have to be the structure of requirements, similar to language patterns but a little more simpler (K. Watahiki et al. and L. Wei et al. proposals); the second one reuse directly texts of requirements, similar to a copy and paste approach but more sophisticated (B. Estes et al., J. Jensen et al., A. Monzon and W. Lam et al. proposals). Table 8 and 9 contains the proposals founds and their main general characteristics.

		K. Watahiki et al. [SLR30], [SLR31]	L. Wei et al. [SLR32]	B. Estes et al. [SLR33]
Object to be reused characteristics	1. Domain	General Purpose	Network Software	Security Requirements
	2. Object to be reused	SNL Structure	SNL Structure	SNL
	3. Requirement type	F	F, NF	F, NF
	4. Notation of the object to be reused	Natural Language	Natural Language	Natural Language
	5. Where the object to be reused is contained	Template	Template	Template
	6. Reusable object metamodel	------	Yes	Yes
	7. Context information to use the object	------	------	------
	8. Information for further stages	------	------	------
	9. Relations among reusable objects	Yes	------	------
Catalogue characteristics	1. Arrangement	Repository	------	Repository
	2. Classification for reusable objects	------	------	------
	3. Classification metamodel	------	------	Yes
	4. Existence of a catalogue	Project / Test	------	Project / Test
	5. State of the catalogue	------	------	------
	6. Consideration of catalogue evolution	------	------	Yes (no method)

Use characteristics	1. Reusable object construction methodology	Yes (informal process)	------	Yes (informal process)
	2. Reusable object utilization methodology	Yes (informal process)	------	Yes (informal process)
	3. Reuse tools	------	Yes (use defined patterns to state requirements in natural language)	Yes (to consult the DB of requirements and introduce new ones)
	4. Proposal tested in real cases	Yes	------	------

Table 8: Summary of the systematic review works found related to the reuse of requirements using templates in natural language (part I)

		J. Jensen et al. [SLR34]	A. Monzon [SLR18]	W. Lam et al. [SLR35], [SLR36], [SLR37]
Object to be reused characteristics	1. Domain	Security Requirements	General Purpose	General Purpose
	2. Object to be reused	SNL	SNL	Sets of SNL
	3. Requirement type	NF	F, NF	F
	4. Notation of the object to be reused	Natural Language	Natural Language	Natural Language
	5. Where the object to be reused is contained	Template	Template	Template
	6. Reusable object metamodel	------	------	------
	7. Context information to use the object	------	------	------
	8. Information for further stages	------	------	------
	9. Relations among reusable objects	------	------	------
Catalogue characteristics	1. Arrangement	Taxonomy	------	Repository
	2. Classification for reusable objects	Yes	------	------
	3. Classification metamodel	------	------	------
	4. Existence of a catalogue	Project / Test	Project / Test	Project / Test
	5. State of the catalogue	In Evolution	------	------
	6. Consideration of catalogue evolution	------	------	------
Use characteristics	1. Reusable object construction methodology	------	Yes (informal process)	Yes (method)
	2. Reusable object utilization methodology	------	Yes (informal process)	Yes (informal process)
	3. Reuse tools	------	Yes (to mark in requirements specifications what are the requirements that are reusable and reuse them, and also create new requirements)	Yes (a tool to create new tools (or formularies) to reuse the defined patterns)
	4. Proposal tested in real cases	------	Yes	------

Table 9: Summary of the systematic review works found related to the reuse of requirements using templates in natural language (part II)

The main difference between K. Watahiki et al. [SLR30, SLR31] and L. Wei et al. [SLR32] proposals, which are focused on the reuse of text structure, is the way in which they define the structure of these requirement texts. The first one is based in case grammar, one of the techniques to manipulate the semantic structure of sentences and to formalize the rules of the combination of the verbs and their attributes (also called cases), such as "actor", "object" and "instrument". They define a case frame, the entity that contains the sentence structure to reuse, and the case structure that a verb can take (see figure 11 and 12 to have a general overview of the entire approach). This approach has the goal to construct scenarios for any domain using this case frames, so it is focused on defining only the functional requirements of a system. The second proposal, however, define a new language, called SORL, to describe requirements with functional and non-functional properties for networked software. Basically, they define natural language patterns (see figure 13 for an example), which are combined with domain ontologies to help to fill pattern's gaps. Figure 14 give an overview of how natural language patterns and ontologies are related.

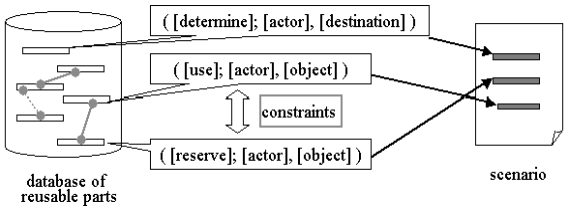


Figure 11: Case frame overview

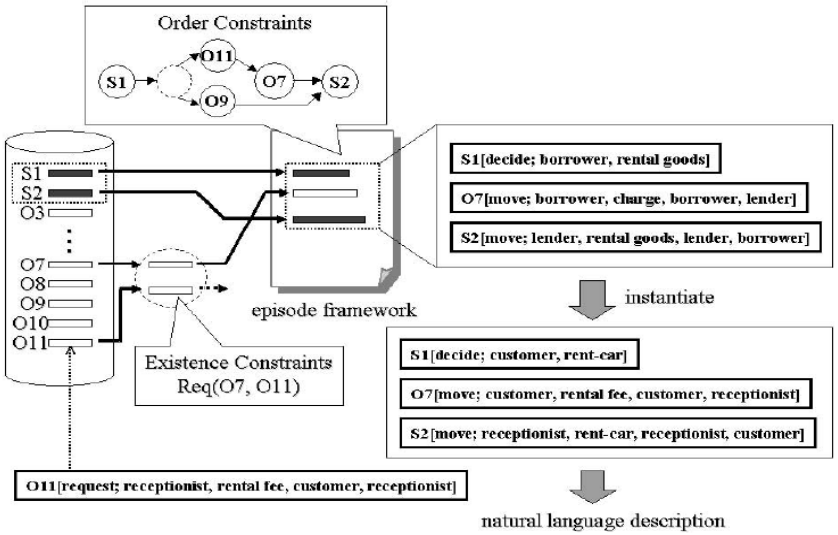


Figure 12: Scenario construction process based on reusing case frames

Element: Behavior	
Pattern name	Pattern syntax
BFRP	<Role> {<DAP> | <NDAP>} [<MP>] [<TP>| <SP>| <SP> <TP>]
DAP	<verb phrase> <entity>
NDAP	<verb phrase>
MP	*prep.* <manner >
TP	{*at*| *in*| *on*}<time>| *from*<time>*to*<time> | *between*<time>*and*<time> | *before*<time >| *after*<time >
SP	{*at*| *on*| *in*}<space>| *from*<space>*to*<space >

Figure 13: SORL natural language pattern example

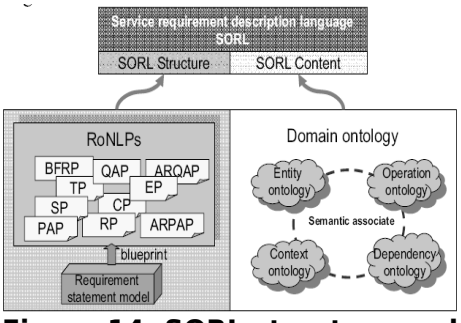


Figure 14: SORL structure and content

As explained before, the second group of approaches in this category deals directly with the reuse of texts of requirements. The basic principle of these approaches is to have a repository or taxonomy which stores all reusable requirements. Then the reuse method entails that requirement engineers have to look for those requirements that are useful for a new system making queries, which sometimes may be even harder that define the requirements from scratch.

Both B. Estes et al. [SLR33] and A. Monzon [SLR18] works propose to reuse the entire content of requirements specifications (i.e. functional and non-functional requirements) to create new specifications systems. The main differences between both proposals are:

1. B. Estes et al. work is focused in system requirements, whereas A. Monzon proposal has no specific domain to its work.
2. B. Estes et al. proposal define a metamodel that describes the structure of requirements, and also their attributes and classification (figure 15).
3. A. Monzon proposes a particular reuse methodology (MIA) that permits three different types of requirements (figure 16): strong reused requirements (or cloned objects), that are those ones considered to evolve synchronously in the future (i.e. if a requirement is changed the cloned requirements also change); weak reused requirements (or derived objects), that are those ones that are copied at the beginning of a project from a project source but evolve separately from the source; and specific requirements (or new objects) for a particular project.

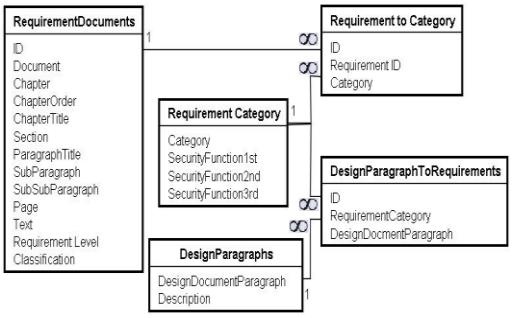


Figure 15: B. Estes et al. requirements metamodel

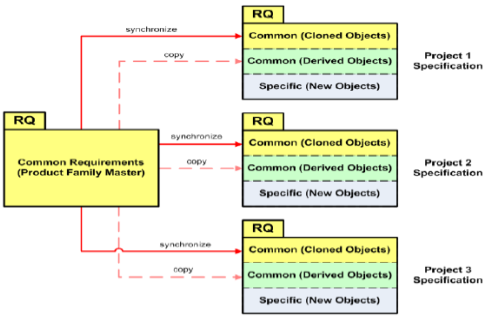


Figure 16: MIA add-in reuse schema

From another point of view, J. Jesen et al. [SLR34] map legislation-based legal requirements for sensitive personal information to a set of reusable technical information security requirements, creating a security requirements catalogue for a concrete legislation. The objective is that if requirements engineers know what the legislation that a new system has to fulfill is, they can reuse directly the requirements of those aspects that are needed for the new system, avoiding that requirements engineers have to make the effort to read and understand complex legislations. Figure 17 contains a subset of the catalogue of reusable security requirements applicable to healthcare.

The last approach found in this category is that one of W. Lam et al. [SLR35, SLR36, SLR37]. They explain from an enterprise point of view the approach taken to achieve reusing the set of requirements in natural language for a specific component, defining as the artifact to reuse the Reusable

Requirements Component (RRC), which encapsulates closely related generic requirements of a component. The main contribution of the proposal, apart from grouping requirements, is the process of abstraction done to generalize requirements as much as possible (making them, for instance, independent of the technology used) and the introduction of parameters to achieve this generalization (figure 18).

Requirement type	Description
Identification and authentication requirements	Services should identify and verify the identity of all of its human users before allowing them access to their resources. [17] §21,[20] §14,[18] §3-6, [16] §2-11 and §2-12
	Services should identify and verify the identity of corresponding services before they are allowed to communicate. [22] §5-4, [20] §14
Authorisation requirements	Services should verify the authorisation level of users before access to sensitive data can be given. [17] §21, §22 [18] §3-6 [16] §2-11,§2-12
Integrity requirements	The platform should support integrity protection of sensitive personal data while it is stored. [22] §5-5, [19] §16, [14] §13, [16] §2-13
	The platform should be able to detect unauthorised manipulation of data that is being transmitted. [22] §5-5, [19] §16, [14] §13, [16] §2-13

Figure 17: Reusable security requirements extracted from healthcare legislations by J. Jensen et al.

Element in Abstraction Process	Example
concrete requirement from system A	When the engine is not in the process of being started, cranked or run, if the fuel switch is in the OFF position and the master crank switch is in the ON position, and the engine start switch is then turned to the ON position then a dry crank will be initiated.
equivalent concrete requirement from system B	When the engine is not in the process of being started, cranked or run, if the fuel switch is in the OFF position and then the engine start switch is turned to the CRANK position, then a dry crank will be initiated.
constant requirement part	When the engine is not in the process of being started, cranked or run, if (X) and then (Y), a dry crank will be initiated.
variable requirement part	X and Y are cockpit-specific signals.
abstraction reasoning	Cockpits are specific to a particular system, and not all systems will have the same cockpit layout. Hence, this aspect is a variable requirement part and must be factored out of the generic requirement.
generic requirement	When the engine is not in the process of being started, cranked or run, if (cockpit signal 1) and then (cockpit signal 2), a dry crank will be initiated.

Figure 18: Example of the abstraction process to achieve generic requirements by W. Lam et al.

4.4.2.2. Use Case Patterns

Use case elicitation and modeling is one of those techniques that have been used extensively in a variety of software development models to bridge the gap between requirements engineers and stakeholders, and capture requirements of software systems. Then, another way of reusing requirements is reuse use cases, and with this goal use case patterns were created. Table 8 summarizes the general characteristics of the relevant proposals found for this systematic review related to use case patterns.

		L. Chung et al. [SLR38]	A. A. Issa et al. [SLR39]	M. Saeki [SLR40], [SLR41]
Object to be reused characteristics	1. Domain	General Purpose	General Purpose	General Purpose
	2. Object to be reused	Use Cases	Use Cases	Use Cases
	3. Requirement type	F, NF	F, NF	F
	4. Notation of the object to be reused	Modeling Language (UML)	Natural Language, Modeling Language	Natural Language, Modeling Language (UML)
	5. Where the object to be reused is contained	Pattern	Pattern	Template
	6. Reusable object metamodel	Yes	Yes (partial)	Yes
	7. Context information to use the object	Yes	Yes	------
	8. Information for further stages	Yes	------	------
	9. Relations among reusable objects	Yes	Yes	Yes

Catalogue characteristics	1. Arrangement	------	Repository	Repository
	2. Classification for reusable objects	------	------	------
	3. Classification metamodel	------	------	------
	4. Existence of a catalogue	Project / Test	General	------
	5. State of the catalogue	------	In Evolution	------
	6. Consideration of the evolution of the catalogue	------	------	------
Use characteristics	1. Reusable object construction methodology	------	Yes (informal process)	Yes (method)
	2. Reusable object utilization methodology	------	Yes (method)	------
	3. Reuse tools	------	------	------
	4. Proposal tested in real cases	------	Yes	------

Table 10: Summary of the systematic review works found related to use case patterns

Both L. Chung et al. [SLR38] and A. A. Issa et al. [SLR39] proposals define use case patterns as diagrams that can be reused in new software projects. The main difference among them is that in the first one, use case diagrams (defined with UML) are accompanied by Non-Functional Requirement (NFR) diagrams of goals and subgoals (defined using the NFR framework [34]) (figure 19) and in the second one use cases in patterns (see figure 20 for an example) are also defined using natural language and some other key attributes, as the goal of the use case and other related use cases.

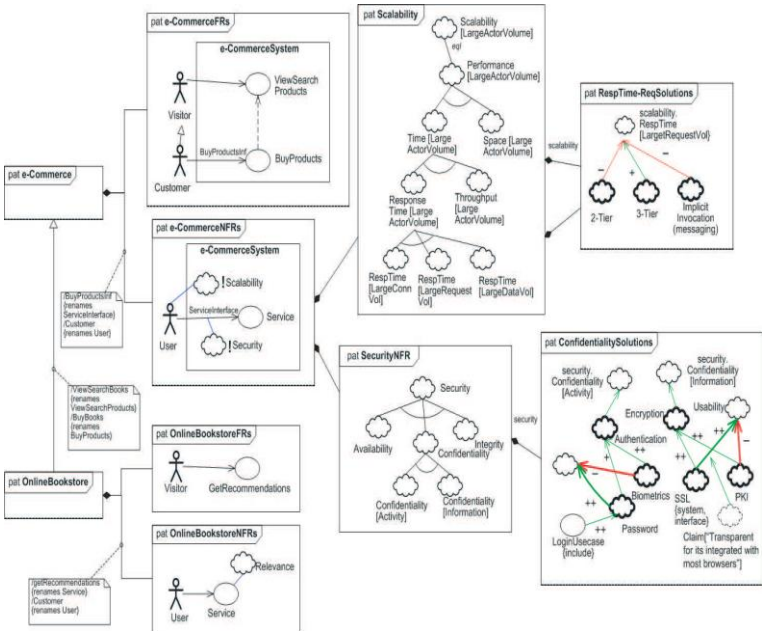


Figure 19: L. Chung et al. use case patterns used in on-line bookstore example

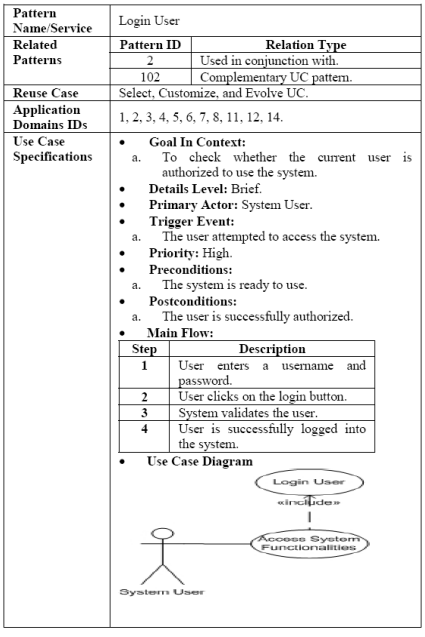


Figure 20: A.A. Issa et al. use case pattern example

The approach of M. Saeki [SLR 40, SLR41] defines use cases in a similar way of the previous approaches (see figure 21 for its metamodel). The particularity of this work is that use case patterns consist of use cases as classes, generalization relationships “extends” and “uses”, and dependency

relationships such as control dependency, data dependency and aggregation. An example of these patterns is shown in figure 21, the *decorator pattern* or *wrapper pattern*, which is used to structure the use cases, so getting flexible and reusable structures.

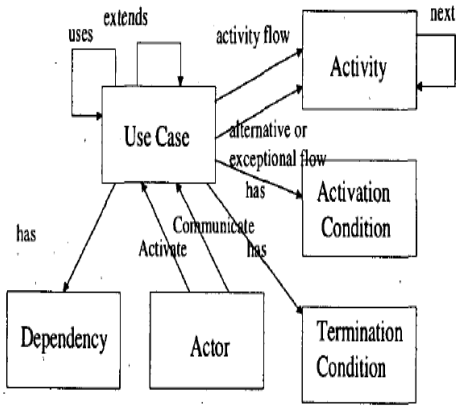


Figure 21: M. Saeki use case pattern metamodel

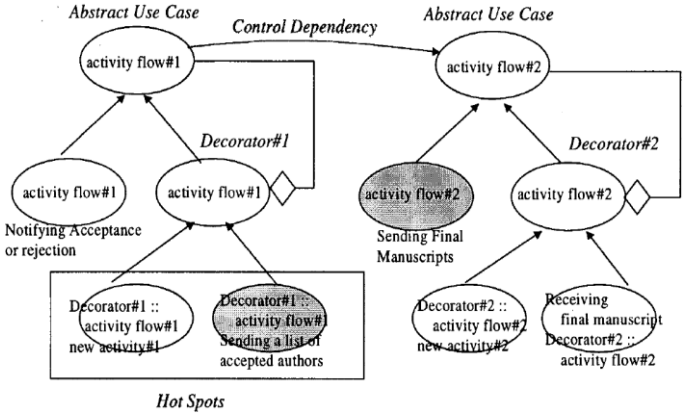


Figure 22: M. Saeki decorator pattern example

4.4.2.3. Requirement Patterns

A different approach to achieve reuse during Requirements Engineering is to create patterns that contain the text of requirements, often parameterized texts, and also some extra information about when each pattern has to be used. Requirements patterns are a sophisticated way of reusing requirements, not just reusing the text of other projects like in most of the sentences templates approaches (section 4.2.1), but embracing also a process of abstraction in order to make requirement patterns as general as possible but without losing the essence of the requirements contained in the pattern. The principal difference among requirement patterns and use case patterns is that the last ones reuse a set of functional requirements to describe a complete functionality, but a requirement pattern contains just one or more requirements that are related with some criteria (for instance, availability or interface language) but they don't contain all the requirements for a specific functionality. We have found seven different approaches that state requirement patterns to reuse requirements during requirements elicitation (table 11 and 12).

A. Toval et al. [SLR42] define the SIREN method (SImple Reuse of software requiremENts) to requirements reuse, where a requirement pattern is defined as a sentence in natural language, sometimes parameterized, that contains also the following attributes: identification, priority, critically, viability, risk, source and other ones that only appear in some requirement patterns. Their repository (figure 23) contains requirement patterns from specific domains and profiles, where each domain or profile is a view of the global repository (i.e. classifications). Domains consist of patterns belonging to a specific application field, such as accounting or finance, whereas profiles consist of homogeneous set of patterns that can be applied to a variety of domains, such as information systems security or personal data privacy law. Their requirement patterns can also include any kind of objects as complementary information, like tables or schemas of any type.

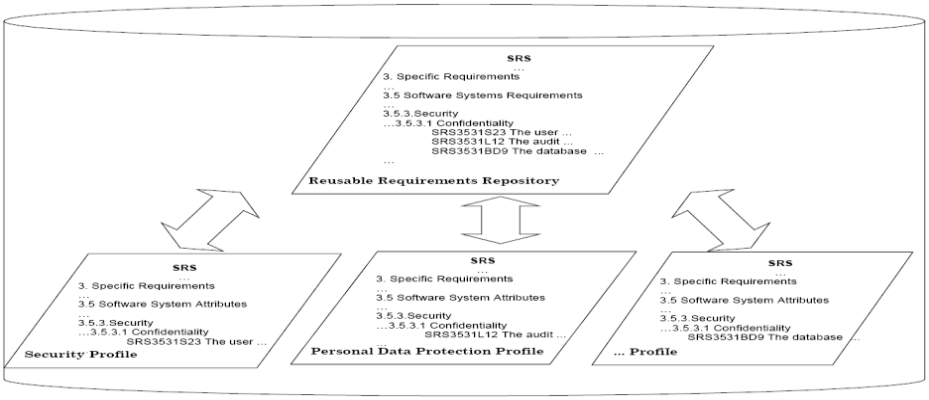


Figure 23: SIREN reusable requirements repository

		A. Toval et al. [SLR42]	R.S. Wahono et al. [SLR43]	M. Mannion et al. [SLR44], [SLR45], [SLR46]	A. Durán Toro et al. [SLR22]
Object to be reused characteristics	1. Domain	Security Requirements	Web Systems	General Purpose	General Purpose
	2. Object to be reused	SNL	SNL, Use Cases	SNL	SNL, Use Cases
	3. Requirement type	F, NF	F, NF	F	F, NF
	4. Notation of the object to be reused	Natural Language (NL)	NL, Modeling Language	Natural Language	Natural Language
	5. Where the object to be reused is contained	Pattern	Pattern	Pattern	Pattern
	6. Reusable object metamodel	------	------	------	------
	7. Context information to use the object	Yes	Yes	Yes	Yes
	8. Information for further stages	Yes	Yes	------	------
	9. Relations among reusable objects	Yes	Yes	Yes	------
Catalogue characteristics	1. Arrangement	Repository	Repository	Repository	Repository
	2. Classification for reusable objects	Yes	Yes	------	------
	3. Classification metamodel	------	------	------	------
	4. Existence of a catalogue	Project / Test	Project / Test	Project / Test	Project / Test
	5. State of the catalogue	In Evolution	------	In Evolution	------
	6. Consideration of the evolution of the catalogue	Yes (no method)	------	------	------
Use characteristics	1. Reusable object construction methodology	------	------	Yes (informal process)	------
	2. Reusable object utilization methodology	Yes (method)	Yes (method)	Yes (method)	Yes (informal process)
	3. Reuse tools	------	------	Yes (to search in the repository and create new patterns)	------
	4. Proposal tested in real cases	Yes	------	------	Yes

Table 11: Summary of the systematic review works found related to requirement patterns (part I)

		S. Konrad et al. [SLR47]	S. Withall [SLR23]	X. Franch et al. [SLR48], [SLR49], [SLR50]
Object to be reused characteristics	1. Domain	Embedded Systems	General Purpose	General Purpose
	2. Object to be reused	SNL, Use Cases, Conceptual Models	SNL	SNL
	3. Requirement type	F, NF	NF	NF
	4. Notation of the object to be reused	Natural Language, Modeling Language (similar to UML)	Natural Language	Natural Language
	5. Where the object to be reused is contained	Pattern	Pattern	Pattern
	6. Reusable object metamodel	------	------	Yes
	7. Context information to use the object	Yes	Yes	Yes
	8. Information for further stages	Yes	Yes	------
	9. Relations among reusable objects	Yes	Yes	Yes
Catalogue characteristics	1. Arrangement	Repository	------	Taxonomy
	2. Classification for reusable objects	------	Yes	Yes
	3. Classification metamodel	------	------	Yes
	4. Existence of a catalogue	General	General	General
	5. State of the catalogue	------	------	Finished
	6. Consideration of the evolution of the catalogue	------	Yes (no method)	Yes (method)
Use characteristics	1. Reusable object construction methodology	------	Yes (method)	Yes (method)
	2. Reusable object utilization methodology	------	Yes (method)	Yes (method)
	3. Reuse tools	------	------	Yes (one to manage requirement patterns and evolve the catalogue; another one to create requirements specifications using patterns)
	4. Proposal tested in real cases	------	------	Yes

Table 12: Summary of the systematic review works found related to requirement patterns (part II)

R.S. Wahono et al. [SLR43] define the concept of extensible requirement pattern for web applications. They define an extensible requirement pattern as a reusable and extensible framework for requirements that support a system analyst to validate the customer feedback (and needs), by reusing and reconstructing requirement patterns. An extensible requirement pattern has some features related to the extensibility characteristic. Not like other existing patterns, an extensible requirement pattern has two kinds of pattern elements: constraint elements (pattern name, which also includes the classification, problem, contexts, examples, forces, related patterns) and extensible elements (solution, user behaviour, page behaviour,

prototype, data behaviour, architecture, security), which is the main difference between this proposal and the other requirement patterns studied. Figure 24 contains the template of an extensible requirement pattern filled with the example *Products Catalogues* pattern. Another interesting point of this proposal is the method defined to use these patterns (figure 25).

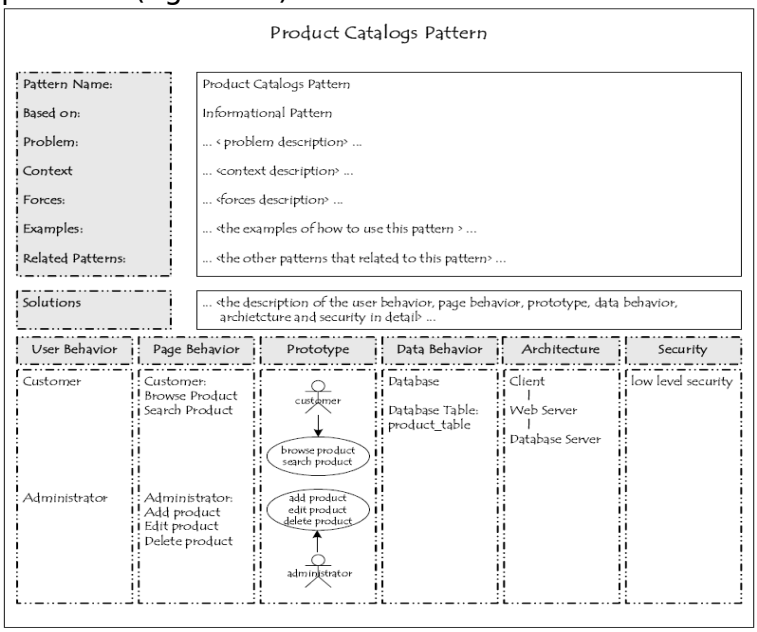


Figure 24: Extensible requirement pattern template filled with the *Products Catalogues* pattern

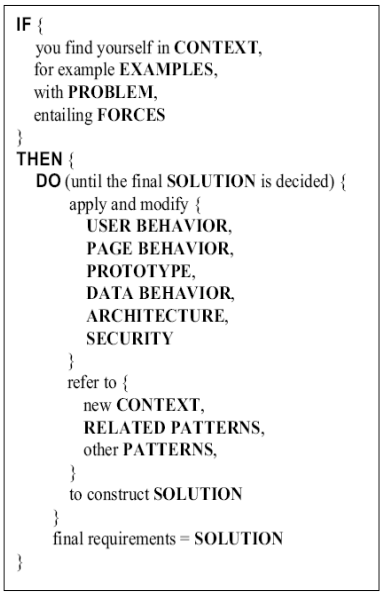


Figure 25: Methodology to use extensible requirement patterns

Other way to understand requirement patterns is thus of M. Mannion et .al [SLR44, SLR45, SLR46]. They define a Viewpoint-Oriented Domain Requirements Definition method (VODRD) to analyze user requirements from an individual stakeholder's perspective. Because of this, their concept of requirement pattern is, as usual, a parameterized sentence in natural language, but grouping these sentences using viewpoints, in a way that each requirement pattern corresponds to a different viewpoint. Figure 26 contains the example *Flight Dynamic Systems* of a viewpoint requirement pattern.

Viewpoint	Flight Dynamics System
Rationale	External system
Associations	None
Requirement	ERS1-FDS-01
Definition	The MPS shall take Flight Prediction files from the flight dynamics system and modify the plan to allow the operations [OP_a, OP_b, OP_c] to occur
Rationale	To ensure accurate plans
Associations	ERS1-SEC-01
Requirement	CLUSTER-FDS-01
Definition	The MPS will use the Flight Dynamics Prediction files to define windows in the plan for the operations [OP_a, OP_b, OP_c, OP_d]
Rationale	To produce accurate plans
Associations	CLUSTER-SEC-01

Note: [OP_a, OP_b, OP_c, OP_d] are variables whose definition and value appear in the domain dictionary.

Figure 26: Viewpoint requirement pattern example

Finally, the proposals of A. Durán Toro et al. [SLR22], S. Konrad et al. [SLR47], S. Withall [SLR23] and X. Franch et al. [SLR48, SLR49, SLR50] define requirement patterns in a similar way. They understand requirement patterns as one or more sentences in natural language, where each one corresponds to a requirement, and some metadata fields (that are the same for all the patterns defined). The main differences between their requirement patterns are (figures 27, 28, 29 and 30 have a template or an example of requirement pattern for each one of the different proposals):

- Some proposals consider more metadata than other ones. For instance, A. Durán Toro et al. requirement patterns state the *importance* and *urgency* of the requirements contained in the pattern.
- In S. Withall and X. Franch et al. proposals, they distinct alternative ways to write the same requirement (for instance, the same requirement can have more or less detailed information, or it could be the case that for different contexts the requirement is expressed in different ways).
- S. Konrad et al. and S. Withall works also incorporate information for further development stages. For instance, the first one incorporates the field design patterns and the second one includes some considerations related with the requirements contained in the pattern for further development and testing.
- S. Konrad et al. proposal also incorporates some fields (*behavior, participants, collaborations*) to state models that are related to the requirements considered in the pattern.

It is important to highlight which is exactly the field that contains in each one of the proposals the final requirement that will be incorporated into the requirements specification: in A. Durán Toro et al. is the *description* field; in S. Konrad et al. is the *constraints* field; in Withall is the *template(s)* field; in X. Franch et al. is the *form text* field.

From another point of view, these proposals also differ in the assets that surround requirement patterns, such as whether there is a methodology to construct or to use patterns, classifications for them and metamodels to define both patterns and their classification. In this area, S. Withall and X. Franch et al. works are worth noting. Withall also define in his work how classify requirement patterns to facilitate their access, a methodology to construct and use requirement patterns and, although in an informal manner, how to evolve the defined catalogue. However, X. Franch et al. goes one step further and, apart from a classification for patterns (using this classification as a taxonomy to access patterns) and a methodology to construct and use them, also defines the metamodel for both requirement patterns and their classification and a methodology to evolve the catalogue. Another interesting point of this work is that they have constructed a catalogue for non-functional requirements which is already finished and can be used in real projects, and they have developed two different tools in order to facilitate the use of their patterns in real projects: one to create and manage requirement patterns, their classifications and to support the evolution of the catalogue, and another one to use these patterns in real software projects, following the methodology defined to their use, and finally to create the corresponding requirements specifications for each project. For more information about some of Franch et al. assets see section 6.

RI–<*id*>	<*descriptive name*>
Version	<*current version number*> (<*current version date*>)
Author	<*current version author*> (<*author's organization*>)
Source	<*current version source*> (<*source's organization*>)
Purpose	<*purpose of requirement*>
Description	The system shall store the information corresponding to <*relevant concept*>. More precisely:
Specific data	• <*specific data about the relevant concept*> • …
Time interval	{ past and present, only present }
Importance	<*importance of requirement*>
Urgency	<*urgency of requirement*>
Comments	<*additional comments about the requirement*>

L-pattern for information storage requirements

RF–<*id*>	<*descriptive name*>	
Version	<*current version number*> (<*current version date*>)	
Author	<*current version author*> (<*author's organization*>)	
Source	<*current version source*> (<*source's organization*>)	
Purpose	<*purpose of requirement*>	
Description	The system shall behave as described in the following sequence of interactions when <*triggering event*>	
Precondition	<*precondition of use case*>	
Ordinary sequence	**Step**	**Action**
	…	…
	n	{The {<*actor*>, system} <*action performed by actor/system*>, Steps described in <*use case (RF–x)*> are performed}
	n.1	If <*condition*>, {the {<*actor*>, system} <*action performed by actor/system*>, steps described in <*use case (RF–x)*> are performed}
	…	…
	…	…
Postcondition	<*postcondition of use case*>	
Exceptions	**Step**	**Action**
	p	If <*exception condition*>, {the {<*actor*>, system} <*action performed by actor/system*>, steps described in <*use case (RF–x)*> are performed}, then the sequence is {resumed, aborted}
	…	…
Performance	**Step**	**Maximum time**
	q	*m* seconds
	…	…
Frequency	This use case is expected to be performed <*number of times*> times/<*time unit*>	
Importance	<*importance of requirement*>	
Urgency	<*urgency of requirement*>	
Comments	<*additional comments about the requirement*>	

L-pattern for functional requirements

Figure 27: A.Durán Toro et al. L-patterns for information systems requirements and functional requirements

Pattern Name and Classification: The pattern name consists of a description of the pattern; the classification provides the purpose of the pattern.

Intent: A brief description of the problems that the pattern addresses.

Motivation: A description of sample goals and objectives of a system that motivate the use of the pattern Problem frame diagrams are used to provide context for the problem specified by the requirement of the pattern. Furthermore, use-cases and use-case-diagrams describe the goals of the pattern application. (These diagrams are not included in the examples due to space constraints.)

Constraints: Restrictions that are applied to the system. For example, constraints can elucidate the system's special hardware constraints and timing restrictions. Any environmental, domain, or implementation-imposed constraints should be included here; these may be functional or non-functional by nature. Safety should also be viewed in this section due to its importance in embedded systems.

Applicability: Provides a description concerning the conditions in which the pattern may be applied.

Structure: A representation of the classes and their relationships depicted in terms of UML class diagrams.

Behavior: Provides an illustrative representation of scenarios for class and object interaction. Also gives a description of the behavior of the pattern by using UML state and sequence diagrams.

Participants: Itemizes the classes/objects that are included in the requirements pattern and their responsibilities.

Collaborations: Describes how objects and classes interact, as well as their roles for carrying out various responsibilities.

Consequences: How are the objectives supported by a given pattern? Using the pattern, what are the trade-offs and outcomes?

Design Patterns: Applicable design patterns that can be used to refine the requirements patterns.

Also Known As: Lists alternative names for the Requirements Pattern.

Related Requirements Patterns: What requirements patterns are related to this one? What are the advantages and shortcomings of this pattern compared to different ones and which pattern should be used?

Figure 28: S. Konrad et al. requirement patterns template for embedded systems

1. **Basic details** The pattern manifestation, owning domain, related patterns (if any), anticipated frequency of use, pattern classifications, and pattern author.
2. **Applicability** In what situations can the pattern be applied? And when can it not be applied?
3. **Discussion** How do we write a requirement of this type? What does a requirement of this type need to consider?
4. **Content** What must a requirement of this type say? What extra things *might* it say? This is the main substance of the pattern.
5. **Template(s)** A starting point for writing a requirement of this type—or more than one if there are distinct alternative ways.
6. **Example(s)** One or more representative requirement written using this pattern.
7. **Extra requirements** What sorts of requirements often follow on from a requirement of this type? And what pervasive systemwide requirements might define something for all requirements of this type?
8. **Considerations for development** Hints for software designers and engineers on how to implement a requirement of this type.
9. **Considerations for testing** What do we need to bear in mind when deciding how to test this type of requirement?

Figure 29: Withall's requirement pattern template

Requirement Pattern *Failure Alerts*	**Description**	*This pattern expresses the need of a software solution for having the capability to inform its users about failures*	
	Comments	*The alert is supposed to be issued at the moment the failure occurs.*	
	Pattern goal	*Alert the users about failures*	
	Author	*Oscar Mendez-Bonilla*	
	Sources (0..*)	• *Requirement books from CITI.* • *Specialized literature.*	
	Keywords (0..*)	*Alert, Failure, Crash*	
	Dependencies (0..*)	*IMPLIES: Failure Reports*	
Requirement Form *Failure Alerts Provided*	**Description**	*This form does not establish any relationship among the type of alert and the type of failure.*	
	Comments	*Each extension may be applied just once*	
	Version	*Wed, 26/11/2008 - 2:25am*	
	Author	*Oscar Mendez-Bonilla*	
	Sources (0..*)	• *Requirement books from CITI.* • *Specialized literature.*	
	Fixed Part	**Form Text**	*The solution shall give an alert in case of failure*
	Extended Part *Alert Types*	**Form Text**	*Alerts provided by the solution shall be: AL*
		Parameter	**Metric**
		AL: is a non-empty set of alert types	*AL: Set(AlertType)* *AlertType: {E-mail, SMS, Page, Fax, Skype, IM, ...}*
	Extended Part *Failure Types*	**Form Text**	*Failures to be alerted of shall be: FL*
		Parameter	**Metric**
		FL: is a non-empty set of failure types	*FL: Set(FailureType)* *FailureType: {Server Crash, Network Crash, ...}*
Requirement Form *Alert Types Dependent on Failure Types*	**Description**	*This form establishes a dependency among the type of alert and the type of failure that occurs.*	
	Comments	*The extensions may be applied more than once*	
	Version	*Wed, 26/11/2008 - 2:45am*	
	Author	*Carme Quer*	
	Sources (0..*)	• *Requirement books from CITI.* • *Specialized literature.*	
	Fixed Part	**Form Text**	*The solution shall give alerts of a certain type depending on the type of failure*
	Extended Part *Specific Dependence*	**Form Text**	*An alert of one of the types AL shall be provided for a failure of some one of the types FL*
		Parameter	**Metric**
		AL: is a non-empty set of alert types	*AL: Set(AlertType)* *AlertType: {E-mail, SMS, Page, Fax, Skype, IM, ...}*
		FL: is a non-empty set of failure types	*FL: Set(FailureType)* *FailureType: {Server Crash, Network Crash, ...}*

Figure 30: X. Franch et al. requirement patterns example

4.4.2.4. Refinements Patterns

A different way from the previous proposals to reuse requirements consists on the refinement of requirements into more detailed ones, which normally add more restrictions to the new system. During this systematic review we found two proposals that deal with the refinement of requirements in different ways (see table 13 for further details).

		S. Supakkul et al. [SLR51], [SLR52], [SLR53]	**R. Darimont et al. [SLR54]**
Object to be reused characteristics	1. Domain	General Purpose	General Purpose
	2. Object to be reused	Refinement of i* models	Refinement of KAOS diagrams
	3. Requirement type	NF	F
	4. Notation of the object to be reused	Modeling Language (i*)	Formal Language (Temporal Formal Logic)
	5. Where the object to be reused is contained	Pattern	Template
	6. Reusable object metamodel	------	------
	7. Context information to use the object	Yes	------
	8. Information for further stages	------	------
	9. Relations among reusable objects	Yes	Yes
Catalogue characteristics	1. Arrangement	------	Repository
	2. Classification for reusable objects	------	------
	3. Classification metamodel	------	------
	4. Existence of a catalogue	------	General
	5. State of the catalogue	------	In Evolution
	6. Consideration of the evolution of the catalogue	------	Yes (no method)
Use characteristics	1. Reusable object construction methodology	------	Yes (informal process)
	2. Reusable object utilization methodology	Yes (method)	Yes (informal process)
	3. Reuse tools	Yes (one to model NF Requirements, another one to capture and visualize them)	------
	4. Proposal tested in real cases	------	------

Table 13: Summary of the systematic review works found related to refinement patterns

In S. Supakkul et al. [SLR51, SLR52, SLR53] a Non-Functional Requirement (NFR) framework is presented for capturing NFR knowledge using goal, problem, casual attribution, solution/means and requirement patterns that can be used to refine each one of these i* elements. To capture NFR knowledge, the framework defines five kinds of NFR patterns (or refinement patterns), including goal pattern for clarifying NFRs, problem pattern for identifying goal obstacles, causal attribution pattern for identifying problem

causes, solution/means pattern for capturing solutions to solve problems or means to achieve goals, and requirements pattern for how a solution/means may be specified as requirements. Figure 31 illustrates how different NFR patterns may be used during requirements engineering to help to develop early-phase goal models and late-phase requirements models. In addition to the refinement rule (defined using i*), each NFR pattern also consists of a unique name, and supplemental applicability (e.g., when and where) and credential (e.g. author and sources) information. An NFR pattern and its application are shown in figure 32.

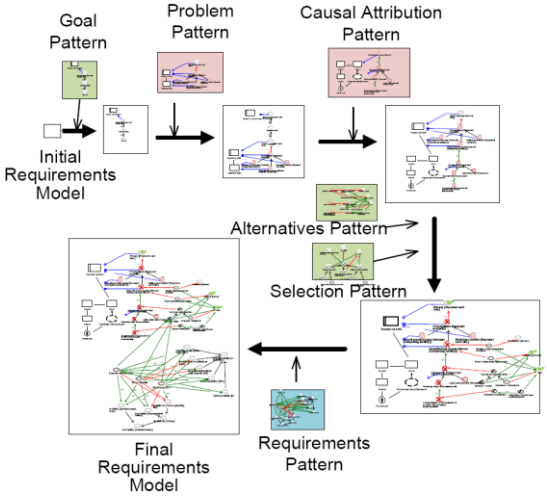


Figure 31: NFR refinement pattern application example

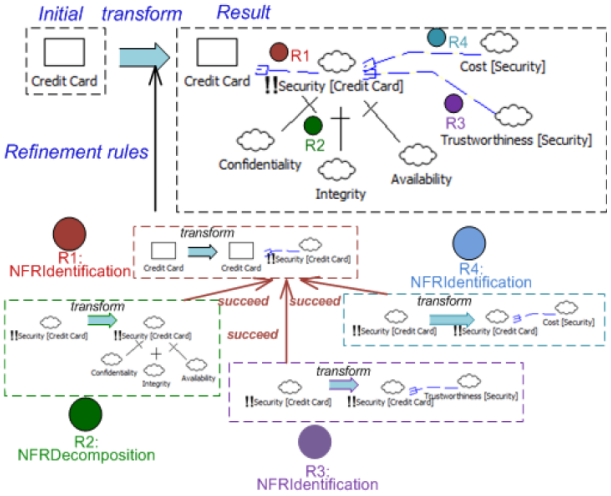


Figure 32: NFR refinement pattern example and its application

Using KAOS language, R. Darimont et al. [SLR54] proposes a similar approach to the previous one. They propose a method to goal refinement and operationalization using refinement templates, which can be used for guiding the refinement process or for pointing out missing elements in a refinement. In this proposal, a refinement template is a one-level AND-tree of abstract goal assertions such that the set of leaf assertions is a complete refinement of the root assertion. Figure 33 contains a refinement template example to decompose *Achieve* goals into three subgoals. As an example of application of the previous template (figure 34), we can consider a train control system. One functional goal is to ensure that trains move through consecutive blocks (figure 34.a). A particular case is when block *b+1's* signal is set to *go*. One may thus instantiate the meta-variable *R* in the first subgoal of the template to the predicate *Go[b+1]* formalizing this situation; hence the previous functional requirement is refined into the three subgoals shown in figure 34.b.

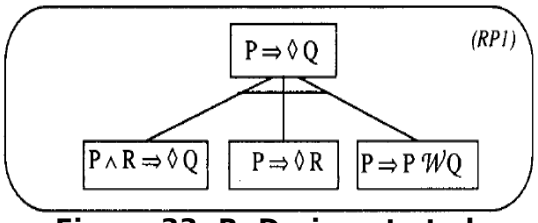


Figure 33: R. Darimont et al. refinement pattern to decompose *Achieve* goals

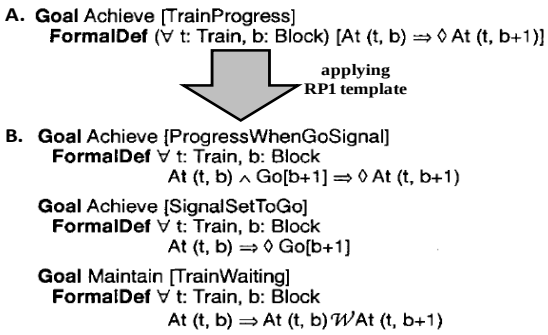


Figure 34: R. Darimont et al. refinement pattern application example

4.4.2.5. Variability Requirement Patterns

A completely different way of using patterns for achieving reuse is when Domain Engineering approach is adapted to Requirements Engineering, having in this way patterns that contain a set of requirements but where variability is integrated. That is, among the set of requirements that a pattern contains, is not necessary to use all these requirements when a pattern is applied. Instead, these requirements follow some usage rules (for instance, exclusion between some requirements) that have to be fulfilled when applying this pattern. This type of patterns is what we call variability requirement patterns. Table 14 summarizes the general characteristics of the relevant proposals found for this systematic review related to this type of patterns.

		N. Heumesser et al. [SLR55]	M. Mannion et al. [SLR21]	J. Dehlinger et al. [SLR56], [SLR57]
Object to be reused characteristics	1. Domain	Security Requirements	General Purpose	Agent-based Systems
	2. Object to be reused	SNL	SNL	SNL
	3. Requirement type	F	F, NF	F, NF
	4. Notation of the object to be reused	Natural Language	Natural Language	Natural Language
	5. Where the object to be reused is contained	Template	Template	Template
	6. Reusable object metamodel	------	Yes	Yes
	7. Context information to use the object	------	------	------
	8. Information for further stages	------	------	------
	9. Relations among reusable objects	------	Yes	------
Catalogue characteristics	1. Arrangement	Repository	------	------
	2. Classification for reusable objects	------	------	------
	3. Classification metamodel	------	------	------
	4. Existence of a catalogue	Project / Test	Project / Test	Project / Test
	5. State of the catalogue	------	------	------
	6. Consideration of the evolution of the catalogue	------	------	Yes (method)
Use characteristics	1. Reusable object construction methodology	------	Yes (method)	Yes (method)
	2. Reusable object utilization methodology	------	Yes (method)	Yes (method)
	3. Reuse tools	Yes (to browse and manage templates)	------	------
	4. Proposal tested in real cases	Yes	Yes	------

Table 14: Summary of the systematic review works found related to variability requirement patterns

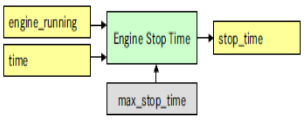
X. Engine Stop Time

X.1 Modification history

<omitted>

X.2 Overview

X.2.1 Relationship to other functions



X.2.2 Core functionality

The function calculates the time since the engine has been turned off in minutes (up to a fixed value). This value is needed by the engine to optimize engine restart.

X.3 Functionality description

At first activation, stop_time is max_stop_time. When the engine has been started, stop_time is decremented by 10 every minute until it has reached 0. As long as the engine is running, stop_time keeps the value 0. When the engine stops running, stop_time is incremented every minute by 1 until it reaches max_stop_time.
If there is no valid information about the engine state available, it is assumed that the engine is not running.
Setting the system into hibernation mode and waking it afterwards should not affect stop_time in the sense that after waking the system, stop_time provides the same value as it would have done without interruption by hibernation mode. During hibernation mode, however, no information is provided.

X.4 Interfaces and parameters

X.4.1 Inputs

Name	Description	Values
engine_running	Is engine running?	true/false/error
time	System time	hh:mm:ss

X.4.2 Outputs

Name	Description	Values
stop_time	Duration engine isn't running	0..max_stop_time

X.4.3 Parameters

Name	Description	Values
max_stop_time	Max. perceived stop time	Integer (minutes)

X.5 Appendix

<omitted>

X.6 Meta-Requirements

– none –

X.7 Specific requirements for car model XYZ

X.7.1 Modification history

<omitted>

X.7.2 Overview of model-specific requirements

Compliant with model-independent description.

X.7.3 Model-specific requirements

– no modifications –

X.7.4 Instantiation of interfaces and parameters

X.7.4.1 Inputs

Name	Source
engine_running	[internal] functionality engine run detection
Time	[CAN] by clock module

X.7.4.2 Outputs

Name	Sink
stop_time	[CAN] e_stp_tm

X.7.4.3 Parameters

Name	Source	Defaultvalue
max_stop_time	[hard coded]	1200 (minutes)

X.7.5 Appendix

<omitted>

Figure 35: N. Heumesser et al. variability requirement template example

No.	Requirement Text
R1	There shall be a telephone number address book facility.
R1.1	There shall be the facility to add a telephone number.
R1.2	There shall be the facility to search for a telephone number.
R1.3	There shall be the facility to delete a telephone number.
R2	The mobile phone shall be able to store @DIRECTORY_SIZE telephone numbers.
R3	The mobile phone shall have a memory capacity of @MEMORY_SIZE.
R3.1	The built-in memory shall be extendible using additional memory within @MEMORY_RANGE.
R3.2	If the built-in memory is equal to @MEMORY_SIZE Gb then the maximum number of telephone numbers stored shall be @DIRECTORY_SIZE = (@MEMORY_SIZE/64) * 100.
R4	The status of the mobile phone shall at any given time be @PHONE_STATUS.
R4.1	The mobile phone shall respond to @COMMAND_NUM commands simultaneously within $COMMAND_RESPONSE seconds.
R5	The mobile phone shall have a display.
R5.1	The mobile phone shall have a black and white display.
R5.2	The mobile phone shall have a colour display.
R6	There shall be the facility to make a telephone call.
R6.1	The mobile phone shall allow making a telephone call by pressing the numeric digits that form a telephone number.
R6.2	The mobile phone shall allow making a telephone call by storing the telephone number in built-in memory and then pressing a memory recall button.
R6.3	The mobile phone shall allow making a telephone call by pressing a ring-back facility to dial the number of the last incoming call.
R6.4	The mobile phone shall allow making a telephone call by using speech recognition technology to say the telephone numbers aloud.
R6.4.1	The speech recognition facility shall be able to recognise single digit numbers.
R7	The mobile phone shall have an email facility.
R8	There shall be the facility to use one or more email protocols.
R8.1	There shall be the facility to use the Post Office Protocol.
R8.2	There shall be the facility to use the Internet Message Access Protocol.
R8.3	There shall be the facility to use the Simple Mail Transfer Protocol.
R9	There shall be a Mobile TV service.

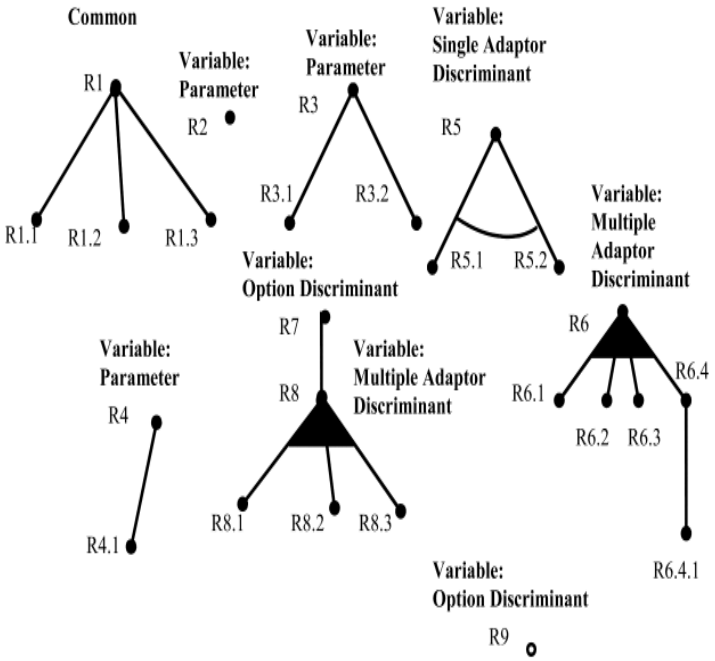


Figure 36: M. Mannion et al. variability requirement template example

N. Heumesser et al. [SLR55] and M. Mannion et al. [SLR21] propose this idea of variability requirement patterns, but in their case templates (as they don't propose any information apart from requirements to be reused), where requirements are expressed using sentences in natural language, in the case of Mannion proposal parameterized. The main difference among both works is the way in which they represent variability. The first one separates model-dependent (i.e. core requirements) from model-independent requirements (e.g. a specific communication protocol used, the implementation of a specific visual display or threshold values) on the same level of abstraction, giving as a result the variability requirement template shown in figure 35. The variability requirement template of the second one, however, is composed by a set of requirements and some variability diagrams that express how a complex requirement is decomposed into simpler ones and how the requirements have to be used when the template is applied (for instance, in the example in figure 36, single adaptor discriminant expresses that their associated requirements are mutually exclusive and multiple adaptor discriminant expresses that from their associated requirements at least one has to be chosen).

J. Dehlinger at al. [SLR56, SLR57] also define variability requirement templates, but from a different perspective of the previous ones. Their idea is defining templates for agent's roles in a way that can be reused. Each agent's role template has the following characteristics: protocols (the way an agent can interact with other agents), activities (computations associated with the role that can be executed without interacting with other agents), permissions (information resources rights that a role has to read) and responsibilities (functionalities of a role). The variation is introduced in this template using variation points, which give a classification of the different levels of intelligence that a role can adopt during its lifetime. All this information is encapsulated in what they call *Role Schema* and *Role Variation Point Schema* (see figure 37 and 38 for an example, respectively). In this way, future systems will employ roles comprising some of the variation points previously defined as well as new capabilities not found in any template.

Role Schema: Cluster Allocation Planning Agent	**Schema ID:** F32
Description: Assigns a new cluster configuration by assigning new microsatellite positions within the cluster. This is done to equalize fuel use across the cluster.	
Variation Points: I4: receive/execute commands [F32-I4] I3: local planning and receive/execute commands [F32-I3] I2: local planning, interaction, partial cluster-knowledge and receive/execute commands [F32-I2] I1: cluster-level planning, interaction, full cluster-knowledge and receive/execute commands [F32-I1]	
Binding Times: All binding time for the variation points are at run-time.	

Role Schema for cluster allocation planning agent

Figure 37: Role Schema example

Role Schema: Cluster Allocation Planning Agent **Schema ID:** F32-I1

Variation Point: I1

Description:

Assigns a new cluster configuration by assigning new microsatellite positions within the cluster. This is done to equalize fuel use across the cluster. This may occur when a new microsatellite is added or in the case of a failure of a microsatellite.

Protocols and Activities:

CalculateDeltaV, UpdateClusterInformation, MoveNewPos, DeOrbit, AssignCluster, AcceptDeltaVBids, RequestDeltaVBids, SendMoveNewPosMsg, SendDeOrbitMsg

Permissions:

Reads -

position	// current position of the microsatellite
velocityIncrement	// current velocity increment of the microsatellite
supplied *microsatelliteID*	// identification number of a microsatellite
supplied *velocityIncrment*	// velocity increment of a microsatellite

Changes -

position	// current position of the microsatellite
velocityIncrement	// current velocity increment of the microsatellite

Generates -

newPositionList	// new position list to assign to the microsatellites within the cluster

Responsibilities:

Liveness -

Optimize the fuel use across the cluster.

Safety -

Prevent microsatellite collisions during a new cluster configuration.

Role Variation Point Schema for the I1 variation point of the cluster allocation planning agent

Figure 38: Role Variation Point Schema example

5. Conclusions

In this systematic review the following goals have been accomplished:

- Definition of a protocol for the systematic review.
- Definition of research questions.
- Historical exploration of the research done around the studied research area.
- Overview of the current and common knowledge of the studied research area.
- Validation of the systematic review.

The definition of the protocol is based on the indications made by B. Kitchenham in [26]. We have established a rigorous protocol to reduce a possible bias in the results. The protocol contains all the strategies used to search and to extract data, including the definition of the research questions.

During the exploration of the research area, we selected the proposals that, after applying the review protocol and the selection criteria, were considered relevant for this study. For each of them we did a little introduction during the previous section, putting special attention in the structure of the patterns and the assets around them.

Finally, the validation of this systematic review has been done taking into account the amount of reviewed work for each topic and the importance of each studied work.

5.1. Answer to the research questions

In the beginning of the systematic review we stated the following questions:

1. *How patterns are used during requirements engineering?*
2. *Are there any specific proposals that use patterns to reuse requirements?*
3. *Among the previous proposals, do any of them propose a well-established set of patterns (i.e. a catalogue)?*

As said at the beginning of the systematic review results analysis, most of the found proposals (21 proposals over 26) that use patterns during Requirements Engineering are focused on the elicitation of requirements, which give the answer to question 1. However, we also found some other works that are not used during this stage, basically those ones that are used to transform informal requirements specifications to formal ones.

During this review, we found exactly 21 proposals (corresponding to 32 results) that use patterns or templates to reuse requirements during the elicitation stage. However, as said during the analysis of the results, some of the proposals are too fuzzy or, on the contrary, they define precisely the pattern or template approach but although it seems that it can be applied during requirements elicitation, they don’t explain how it will be used or the proposed method is not clear explained, which is a drawback for using them in a real situation (see table 15 for a classification). Then, as answer to question 2, we can say that there are proposals that use patterns to reuse requirements, but not in all of them the reuse process is clear.

Elicitation proposals too fuzzy or without a clear reuse method	Elicitation proposals with a clear defined pattern and a clear reuse method
1. L. Wei et al. [SLR32]	1. K. Watahiki et al. [SLR30, SLR31]
2. B. Estes et al. [SLR33]	2. W. Lam et al. [SLR35, SLR36, SLR37]
3. J. Jensen et al. [SLR34]	3. M. Saeki [SLR40, SLR41]
4. A. Monzon [SLR38]	4. M. Mannion et al. [SLR44, SLR45, SLR46]
5. L. Chung et al. [SLR38]	5. S. Withall [SLR23]
6. A. A. Issa et al. [SLR39]	6. X. Franch et al. [SLR48, SLR49, SLR50]
7. A. Toval et al. [SLR42]	7. S. Supakkul et al. [SLR51, SLR52, SLR53]
8. R.S. Wahono et al. [SLR43]	
9. A. Durán Toro et al. [SLR22]	
10. S. Konrad et al. [SLR47]	
11. R. Darimont et al. [SLR54]	
12. N. Heumesser et al. [SLR55]	
13. M. Mannion et al. [SLR21]	
14. J. Dehlinger at al. [SLR56, SLR57]	

Table 15: Classification of the elicitation proposals of the systematic review according their details explained

Answering to question 3, among all the proposals reviewed, only two of them have a well-established catalogue of patterns (table 16), which are those ones of S. Withall and X. Franch et al. Both of them define a requirement patterns catalogue for non-functional requirements that can be applied in any software project of any domain. They are probably the proposals that define better what a requirement pattern is, and that also define a methodology to construct patterns and to use them, even considering how to maintain this catalogue up-to-date. X. Franch et al. work goes one step further and, among other different assets, they have also developed tools to facilitate the reuse in real projects. Not only for their requirement pattern structure, but also for the framework that surround these patterns and the work published about them, we can say that probably this two proposals are the most-consolidated ones, and because of this they have catalogues defined. As a final remark, it's important to say that these two proposals only have worked with non-functional requirements, but it is important to know how they will work with other types of requirements (non-technical or functional ones), so further work is needed in both approaches to deal with this issue.

Detailed elicitation proposals without a well-established catalogue	Detailed elicitation proposals with a well-established catalogue
1. K. Watahiki et al. [SLR30, SLR31]	1. S. Withall [SLR23]
2. W. Lam et al. [SLR35, SLR36, SLR37]	2. X. Franch et al. [SLR48, SLR49, SLR50]
3. M. Saeki [SLR40, SLR41]	
4. M. Mannion et al. [SLR44, SLR45, SLR46]	
5. S. Supakkul et al. [SLR51, SLR52, SLR53]	

Table 16: Classification of the detailed elicitation proposals of the systematic review according their type of catalogue

Part II

Developing a Software Requirement Patterns Catalogue:

The Non-Technical and Functional Parts

6. Introduction to PABRE

This chapter is not a contribution of the Master Thesis, but a description of PABRE that facilitates the understanding of the next chapters and parts of the document.

Requirement elicitation is the process of acquiring the system requirements from the system stakeholders. This process is critical in all software projects: if not all the requirements are elicited, or if some elicited requirements do not describe real stakeholder needs, or if the quality of the requirements is poor (e.g., they suffer from ambiguities), the chance of project failure increases.

Techniques supporting requirements elicitation (interviews, meetings, storyboards...) are mostly oriented to obtain requirements from scratch and they may hardly take advantage of a fundamental observation [35]: "*When specifying a system, it is quite usual that a significant proportion of requirements is recurrent and belongs to a relatively small number of categories, especially in the case of non-functional requirements.*"

The PABRE (PAtterns Based Requirements Elicitation) framework [9] was created by GESSI research group with a clear goal: the reuse of software requirements to help requirement engineers to elicit, validate and document software requirements and, as a consequence, obtain Software Requirement Specifications (SRS), also known as requirements books, of better quality both in contents and syntax.

Among the many proposed approaches to reuse in software engineering, patterns hold a prominent position, and because of this, GESSI selected patterns as the way to achieve reuse. "*Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*" [36].

The PABRE framework is focused on the use of patterns for the requirements elicitation stage, namely Software Requirement Patterns (SRP). The patterns applicability to this context is clear, since requirements that appear over and over in requirements specifications could be identified as the solution to particular problems in a given context (the classical context-problem-solution scenario of patterns).

The current state of the PABRE framework consists of:

- A metamodel that describes the structure of non-functional SRP and also the structure of the catalogue (see section 6.3).
- A non-functional part for the SRP catalogue with 29 patterns, which can be found in [9] (see section 6.4).
- A method for the process of use of the catalogue in the requirements engineering stage. This method includes how to use the set of SRPs to create new requirements specifications, but also how to evolve the catalogue in order to have it up-to-date. More information about this method could be found at [24, 25].
- Two subsystems for helping in the use, management and evolution of the catalogue (PABRE-Man and PABRE-Proj in figure 39). As a brief summary, PABRE-Man facilitates the definition of SRPs, as well as the

maintenance and evolution of an SRP catalogue; PABRE-Proj provides the use of the SRP catalogue during the elicitation stage and the generation of requirement specifications and call-for-tenders documents. For more information about the tools see [9].

The research method used to build requirements patterns catalogue and its underlying metamodel was based on the study of requirement specifications from several call-for-tender real projects conducted by the *Service Science and Innovation (SSI)* department [37] from the *Centre de Recherche Publique Henri Tudor (CRPHT)* [38]; experts' knowledge, being these experts: IT consultants, facilitators and researchers; and background on requirements engineering literature and especially on requirement patterns.

An overview of a typical scenario of use of the PABRE framework can be found in figure 39. The general idea is that the requirement analyst has a meeting with the client in order to elicit the requirements. During this meeting, the requirement analyst use PABRE-Proj subsystem to elicit the requirements using the PABRE catalogue (the set of all the SRPs defined). With this subsystem, once the meeting is finished, the requirement analyst can generate automatically the requirements specification document. All the information about the projects and the use of the PABRE catalogue in these projects is imported from PABRE-Proj to PABRE-Man subsystem. This information is used in PABRE-Man by the requirement patterns expert to evolve the catalogue and keeping it up-to-date (creating new patterns, removing old ones, etc.).

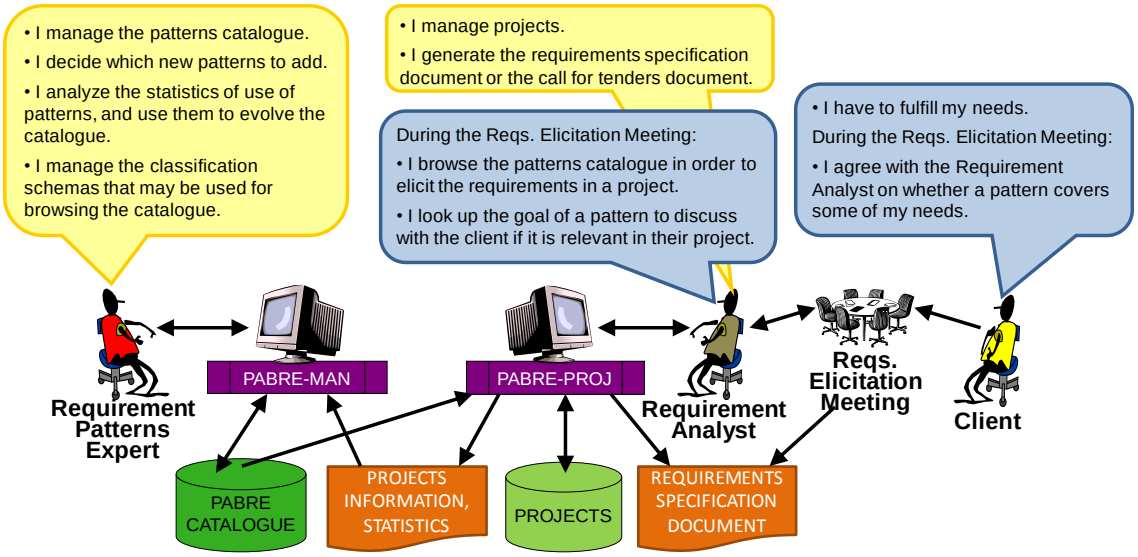


Figure 39: PABRE framework scenario

6.1. What a Software Requirement Pattern (SRP) is

A Software Requirement Pattern (SRP), following the typical context-problem-solution scenario of patterns, basically consists of a template that may generate one or more requirements by tailoring the template (solution) to a certain project, and some information (context-problem) to identify its need to be applied in that particular project.

An SRP represents a cluster of requirements, i.e. a group of interrelated requirements. More concretely, an SRP groups different requirements that are related with the same issue, with the goal of this SRP.

The structure of GESSI Software Requirement Patterns is presented in [24, 25]. Table 17 contains what is the template of a SRP filled with the example Failure Alerts. Summarizing, this template contains:

- **Pattern metadata**. The first set of attributes defines the metadata about the pattern itself: the name of the pattern, its description, its author, comments included by its author and users, its goal, the sources from where it was obtained (e.g., the requirement specifications and projects from which it was identified and included in the repository), and some keywords to facilitate searches in the repository. It is important to highlight the role that the goal attribute plays: a pattern is added in a requirements specification if the customer needs to achieve its goal.
- **Requirement forms**. A requirement pattern, when used in different projects to achieve the same goal, may be written differently, thus the template allows declaring several forms in a pattern. A requirement form captures a particular context that may be the most appropriate for a software project. When we decide to apply this pattern, we have to decide also which the most suitable form for the project is. The *Failure Alerts* pattern has two different forms that differ in the granularity of needed information: if the client needs a specific type of alert when a specific type of error occurs (*Heterogeneous Failure Alerts*) or if the client doesn't need it (*Homogeneous Failure Alerts*). Each form has some metadata similar to the one of the pattern, and exactly one fixed part and may have extended parts.
- **Fixed Part**. Fixed parts of forms are usually quite abstract: the inclusion in a requirements specification of a requirement obtained from the application of a fixed part states that the system has to achieve the goal of the requirement pattern, but it does not state how this goal is achieved. In case of *Homogeneous Failure Alerts* form, the fixed part states that failures will be informed by means of alerts, but it doesn't give further details.
- **Extended Parts**. Since the fixed part of a form is abstract, it is usual to need some extra information or constraints about how to achieve the goal of the requirement pattern. An extended part contains other ways of expressing a requirement (often with parameters), either by rewriting the fixed part or by restricting it. In case of the *Heterogeneous Failure Alerts* form, the extended part states that the system shall trigger *AL* alerts in case of *FL* failures, where *AL* and *FL* may take the set of alerts and failures, respectively.
- **Form text**. The requirements represented by every fixed and extended part of a pattern are specified by a form text. This text is expressed as a short sentence written in natural language that may include one or more parameters that indicate those parts that may vary in different projects. When a pattern is selected and a form applied, the parameters that appear in the text will be substituted by values. In order to define the valid values that a parameter may take, each parameter will be bound to a metric and optionally will also have a correctness condition. In the example, the parameter AL in the extended part of the *Heterogeneous Failure Alerts* form will take as values the (non-empty) set of types of alerts that the customer wishes the solution to provide (SMS, Mail, Fax, Sound, etc.).

Requirement Pattern *Failure Alerts*	**Description**	*This pattern expresses the need of a software solution for having the capability to inform its users about failures.*	
	Comments	*The alert is supposed to be issued at the moment the failure occurs.*	
	Pattern goal	*Alert the users about failures*	
	Author	*Oscar Mendez-Bonilla*	
	Sources (0..*)	• *Requirements specifications from SSI.* • *Specialized literature.*	
	Keywords (0..*)	*Alert, Failure, Crash*	
	Dependencies (0..*)	*IMPLIES: Failure Alerts*	
Requirement Form *Homogeneous Failure Alerts*	**Description**	*This form does not establish any relationship among the type of alert and the type of failure.*	
	Comments	*Each extension may be applied just once.*	
	Version	*Wednesday, 26/11/2008 - 2:25am*	
	Author	*Oscar Mendez-Bonilla*	
	Sources (0..*)	• *Requirements specifications from SSI.* • *Specialized literature.*	
	Fixed Part	**Question Text**	*Does the solution provide alerts in case of failures?*
		Form Text	*The solution shall give an alert in case of failure.*
	Extended Part *Alert Type*	**Question Text**	*Does the solution allow notifying users of errors through specific mechanisms? What are these mechanisms?*
		Form Text	*Alerts provided by the solution shall be: AL.*
		Parameter	**Metric**
		AL: is a non-empty set of alert types	*AL: Set(AlertType)* *AlertType: {E-mail, SMS, Page, Fax, Skype, IM, ...}*
	Extended Part *Failure Type*	**Question Text**	*Does the solution allow warning only specific types of errors? What are these types?*
		Form Text	*Failures to be alerted of shall be: FL.*
		Parameter	**Metric**
		FL: is a non-empty set of failure types	*FL: Set(FailureType)* *FailureType: {Server Crash, Network Crash, ...}*
Requirement Form *Heterogeneous Failure Alerts*	**Description**	*This form establishes a dependency among the type of alert and the type of failure that occurs.*	
	Comments	*The extensions may be applied more than once.*	
	Version	*Wednesday, 26/11/2008 - 2:45am*	
	Author	*Carme Quer*	
	Sources (0..*)	• *Requirements specifications from SSI.* • *Specialized literature.*	
	Fixed Part	**Question Text**	*Does the solution provide alerts depending on the types of failures?*
		Form Text	*The solution shall give alerts of a certain type depending on the type of failure.*
	Extended Part *Alerts for Failure Types*	**Question Text**	*Does the solution allow specific alert mechanisms depending on the type of error occurred? What are the types of alerts depending on the type of error provided by the solution?*
		Form Text	*An alert of one of the types AL shall be provided for a failure of some one of the types FL.*
		Parameter	**Metric**
		AL: is a non-empty set of alert types	*AL: Set(AlertType)* *AlertType: {E-mail, SMS, Page, Fax, Skype, IM, ...}*
		FL: is a non-empty set of failure types	*FL: Set(FailureType)* *FailureType: {Server Crash, Network Crash, ...}*

Table 17: SRP template filled with Failure Alerts example

- **Question text**. Each fixed and extended part of a pattern has a question text associated. This question is expressed as a short sentence written in natural language and corresponds to what would

be asked when you want to know whether or not a product meets the requirement set by the fixed or extended part to which the question text belongs.

- **Dependencies**. Requirements patterns do not live isolated; they may be interrelated in the catalogue. Dependencies among requirement patterns generalize the well-known idea of having dependencies among requirements [39, 40]. These dependencies may be used during the elicitation process (e.g., to help determining the application order) and also they may be propagated to the requirements specification to improve traceability, e.g., if the requirements pattern catalogue has reported that the achievement of a requirement influences on the achievement of another one.

6.2. Classification Schemas

The SRPs in the catalogue need to be indexed following some hierarchical classification schema to facilitate their comprehension and reuse during the elicitation process. A way to understand a classification schema is as a folder tree, such as the ones that we found in operating systems (see figure 41). Currently the repository has two of these hierarchies introduced (figure 40), which are one classification schema based on the ISO 9126-1 characteristics and subcharacteristics [41] and a classification schema, based on the Volere approach [42] and on empirical experiments of SSI-CPPHT.

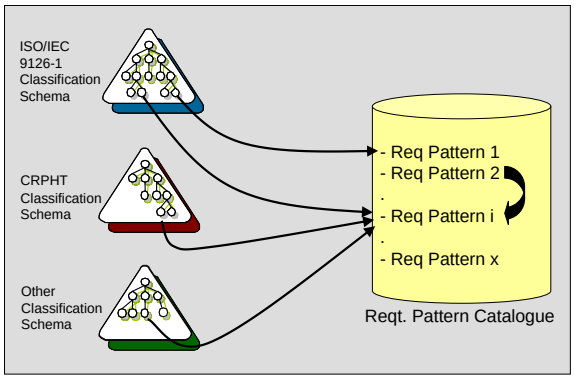


Figure 40: Organization of the SRP catalogue: Patterns, Dependencies and Classification Schemas

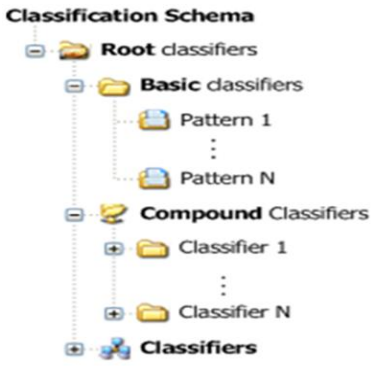


Figure 41: Classification schema as a folder tree

The reason of having several classification schemas is for improving both catalogue's usability and portability: usability, because the same catalogue may be used with different classification schemas by the same requirements engineer; portability, because different requirements engineers, used to other standards or even their own, can customize classification schemas and may view the requirements patterns catalogue with their own perspective.

6.3. Current SRP Metamodel

Figure 42 contains the metamodel to support SRP structure and SRP classification schemas [11]. In this metamodel we can distinguish four independent, but interrelated, parts.

The first part corresponds to the core of a SRP presented in section 6.1, the structure of a requirement pattern. It contains the exact structure and

details of a SRP. Briefly, this structure is: a requirement pattern has several requirement forms, which are made of one fixed part, several extended parts and several constraints.

The second part is the Application part. This part allows adapting a SRP in every project, using parameters and metrics associated to these parameters. In other words, the application part is the only part that varies in every project when you apply a pattern.

The next part is the relationships part. It arises from the observation that patterns' elements are not isolated units of knowledge. This part is useful to create complete requirements specifications without incoherencies. One example of part relationships can be the extended part *Alerts for Failure Types* of the *Failure Alerts* pattern with the extended part *Data Integrity by Failure* of *Recovery Procedures* pattern. The extended part *Alerts for Failure Types* states that the system shall trigger *AL* alerts in case of *FL* failures, and the extended part *Data Integrity by Failure* states that data shall be protected in case of *FL* failures, were both *AL* and *FL* are parameters, being *FL* the set of possible failures. In this two extended parts, the type of failure may be different but, of course, there is some relationship, and because of this it is necessary to establish that the extended parts are related.

Finally, we have the part to support how to classify our SRPs, the Classification part. It gives support to the classification schemas explained in the previous section.

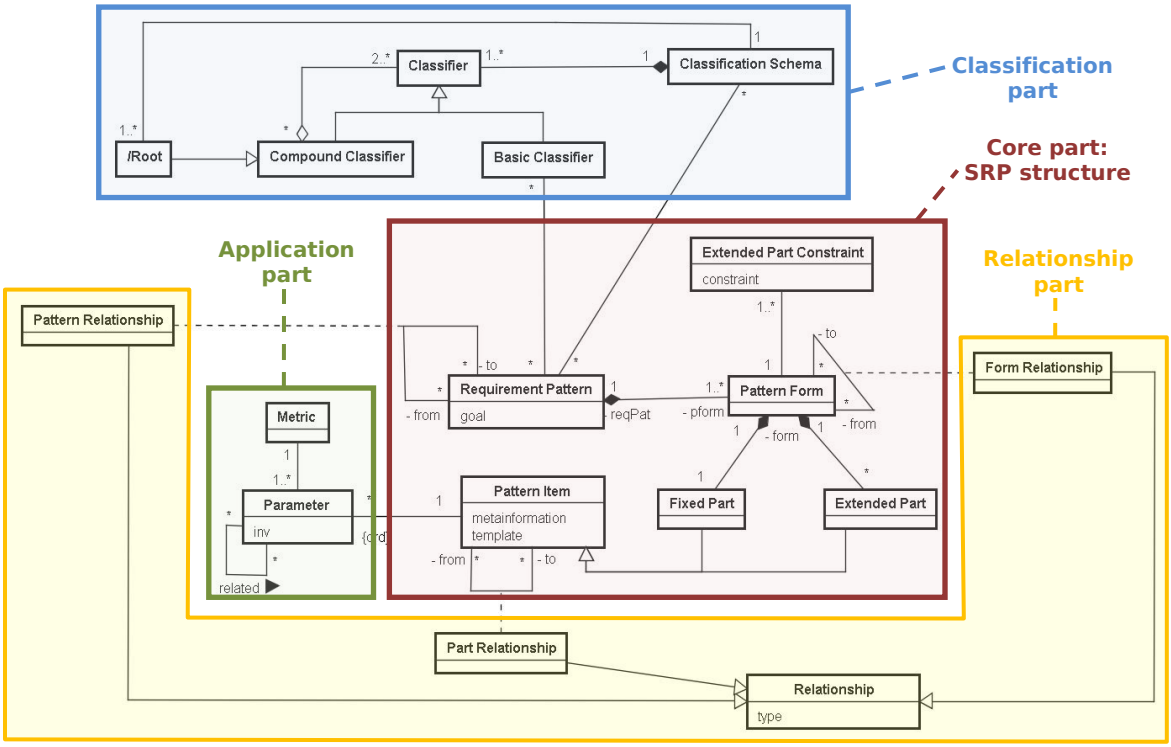


Figure 42: Metamodel for software requirement patterns (SRP) and their classification

The previous metamodel is extended giving further details about the class *Metric*. The metamodel for the class *Metric* and its hierarchy is contained in figure 43. Firstly, a *Metric* can be either a *Simple Metric* (when it is applied it corresponds only to one value) or a *Set Metric* (when it is applied it corresponds to a set of values of the associated *Simple Metric*). *Simple*

Metrics may be enumerated values (*Domain Metrics* in the metamodel) (e.g., names of middleware platforms), *Integer* Numbers (e.g., for stating number of connections supported), *Float* Numbers (e.g., for measuring response time), *Boolean* values (e.g., for knowing if some protocol is supported) and *Time Point* (e.g., the date in which the installation has to be done).

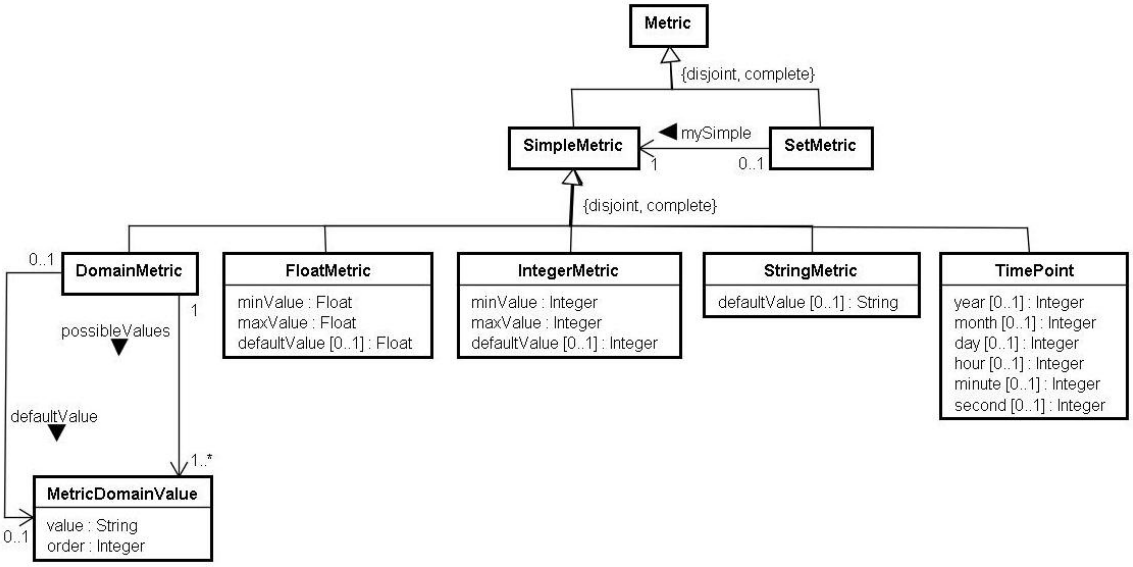


Figure 43: Metamodel for SRP Metrics

6.4. Non-Functional Part of the SRP Catalogue

A catalogue is the set of all defined Software Requirement Patterns (SRPS). The current version of this catalogue has 29 SRPs, which can be found at [9].

Functionality	
Suitability	-
Accuracy	Data precision
Interoperability	Data exchange, Interoperability with external systems
Security	Authentication, Authorization, Automatic logoff, Data transmission protection, Stored data protection
F. Compliance	-
Usability	
Understandability	Interface Language, Interface type
Learnability	Online help, Interface Learnability, Documentation
Operability	Failure alerts, Recovery procedures, Installation procedures, Update procedures
Attractiveness	-
U. Compliance	-
Efficiency	
Time Behaviour	Interface load time, Concurrent users capacity
Resource Utilisation	Data capacity, Users capacity
E. Compliance	Backups, Log

Reliability	
Maturity	Failure alerts
Fault Tolerance	Alternative data storage, Availability, Downtime, Uptime
Recoverability	Backups, Log
R. Compliance	-
Maintainablity	
Analyzability	-
Changeability	-
Stability	-
Testability	-
M. Compliance	-
Portability	
Adaptability	Development language
Installability	Platform
Coexistence	-
Replaceability	-
P. Compliance	-

Table 18: Non-Functional part of the SRP catalogue classified according to ISO/IEC 9126-1

It is the first catalogue that GESSI built and all patterns are about non-functional requirements since this type of requirements are the less

sensitive to changes in the problem domain. In table 18 you can find the non-functional part of the SRP Catalogue classified according to the ISO/IEC 9126-1 quality standard [41].

To construct the current version of the SRP catalogue, GESSI analyzed the non-functional part of 6 requirement specifications that SSI-CPPHT left them. After this phase, they had a catalogue with 48 non-functional requirement patterns. To continue with, they did the refinement of this catalogue. This refinement was done using three different artifacts: firstly, with case studies (this is, a post-portem analysis of one requirement specification); secondly, with expert reviews (done with iterations between SSI-CPPHT and GESSI); thirdly and finally, with the literature review done by GESSI. In this way, they achieved the current version of the SRP catalogue.

7. Obtaining Software Requirement Patterns (SRP)

This section is not a contribution of the Master Thesis, but a description of the method already used to develop the non-functional part of the SRP catalogue and a first approach of a functional part for ERP (Enterprise Resource Planning) systems [58]. This explanation will ease the understanding of the next chapters and parts of the document. Here we define the steps that have to be followed for the creation of SRPs and that we will apply to develop the non-technical and functional parts of the requirement patterns catalogue presented in this Master Thesis.

This method follows a bottom-up process. That is, it takes as input the requirements to build patterns. There are other proposals, as the one described in [44], to do the process with a top-down approach (from general to particular), so we do not discard that a further analysis of this type could be applied similarly in the future, enriching in this way the method, or done in order to validate the resulting patterns. The use of a bottom-up approach has been possible due to the fact that the SSI-CRPHT department [37] lend us some real Software Requirement Specifications (SRSs) as a result of the collaboration that has with GESSI [1].

Before beginning the process of extracting patterns, we must analyze the type of requirement for which we want to create patterns, in our case non-technical and functional ones. For functional ones, the analysis should be specific for the domain of these requirements, which for us is Content Management Systems (CMS).

In this chapter we are going to present the process of extracting patterns. The analysis of each type of requirement will be explained in section 8.1.1 and 9.2.1, respectively.

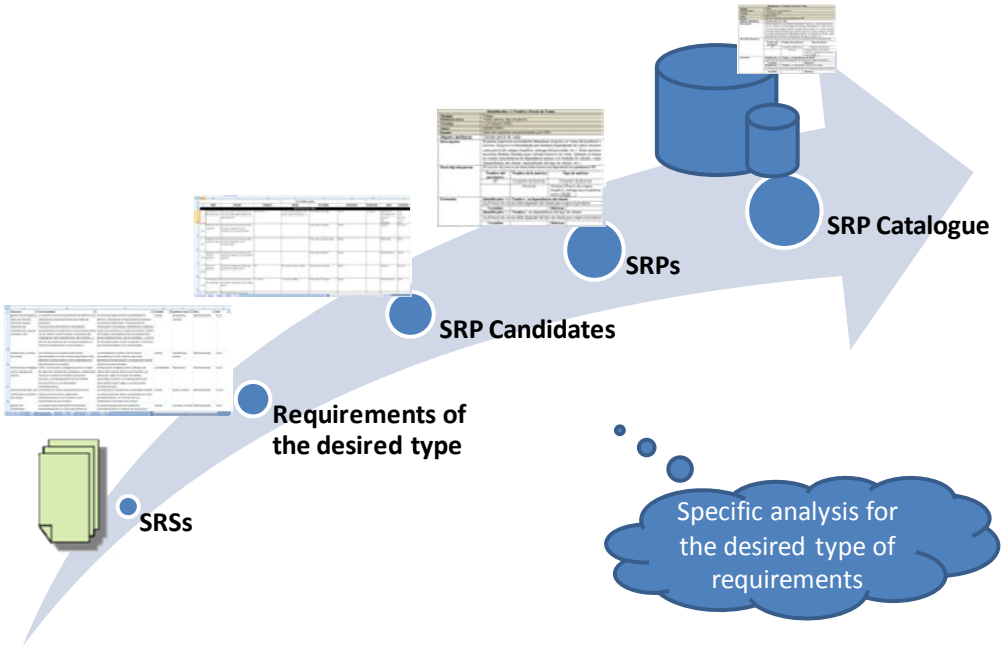


Figure 44: Method to construct parts of the SRP catalogue

As an overview, the method (figure 44) to build patterns for a specific type of requirement follows the steps outlined below:

1. Analyze the specific type of requirement.

2. Extract the requirements of this specific type of SRSs.
3. Perform the semantic analysis and refine the requirements.
4. Insert in the catalogue of candidate patterns those requirements that fit as candidates to patterns, whether as a new candidate or an associated one, depending on the outcome of the search of coincidences in the candidates' catalogue.
5. Create and / or refine the pattern and insert / update it in the patterns catalogue.

The following section describes in detail the method of figure 44, leaving aside the particular type of analysis required, which will be explained in further sections.

7.1. Applied method

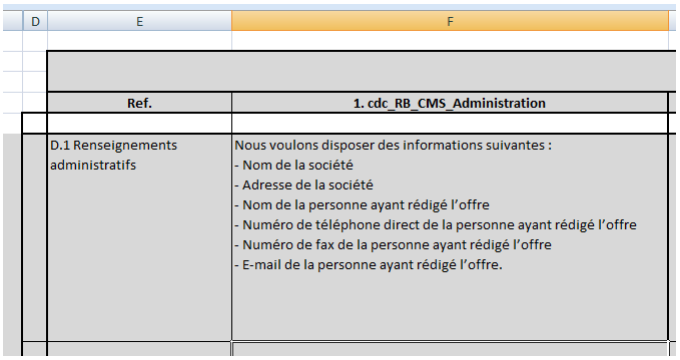
The steps that comprise the method are:

1. Extract requirements of the desired type.
2. Semantic analysis and refinement.
3. Inclusion in the candidates for pattern catalogue.
   3.1. Search of coincidences in the candidates' catalogue.
   3.2. Validation of coincidences and updating or insertion in the candidates' catalogue.
4. Insertion in the catalogue of patterns.
   4.1. Refinement of the pattern.
   4.2. Formalization and storage in the pattern catalogue.

Figure 46 shows the order in which these steps have to be developed and the information flows circulating between them.

7.1.1. Extraction of the requirements of the desired type

The requirements that appear in the SRSs are of three types (functional, non-functional and non-technical) and it is necessary to extract only those ones that are of the desired type, creating a set of target requirements that will evolve until obtain the patters. In our case this step is easy because SSI-CRPHT SRSs have different sections for each type of requirement. This set of target requirements contains requirements as they are found in the SRSs, as well as its location in the document. This allows us to have a reference to the origin of the requirement and to make future queries (see figure 45).



D	E	F
	Ref.	1. cdc_RB_CMS_Administration
	D.1 Renseignements administratifs	Nous voulons disposer des informations suivantes : - Nom de la société - Adresse de la société - Nom de la personne ayant rédigé l'offre - Numéro de téléphone direct de la personne ayant rédigé l'offre - Numéro de fax de la personne ayant rédigé l'offre - E-mail de la personne ayant rédigé l'offre.

Figure 45: Example of requirements extracted

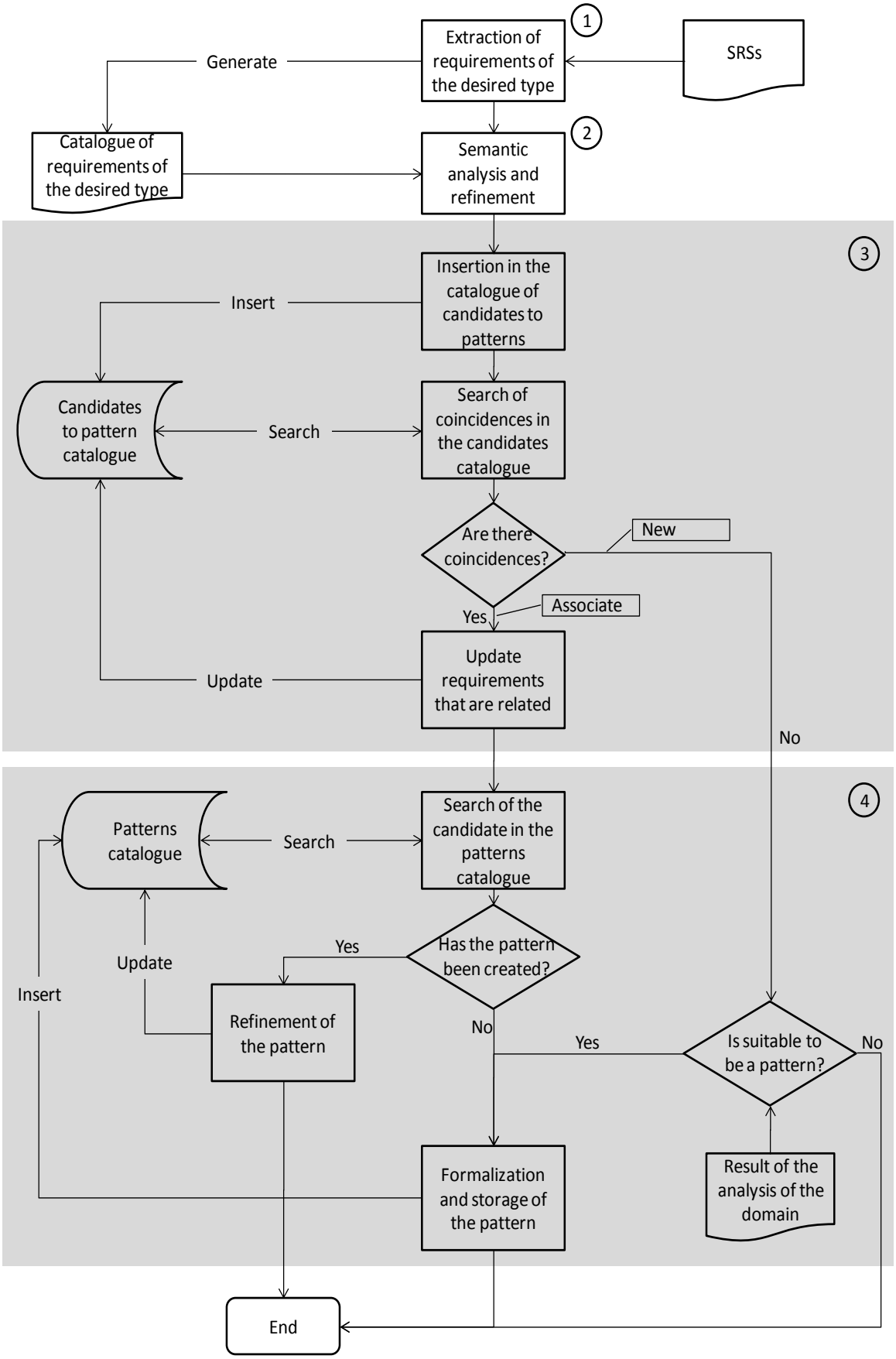


Figure 46: Method to build parts of a SRP catalogue

7.1.2. Semantic analysis and refinement

Once we have the first subset of target requirements, it is necessary to do a semantic analysis and refinement of them. These target requirements are expressed in natural language (see figure 45). The first step is to translate these requirements into a known language, in our case English, using different tools (such as translators and dictionaries).

The same requirement may be expressed differently in each of the SRSs (table 19) or may contain more than one requirement expressed in it (figure 47). Because of this, it is necessary to study each requirement with a semantic analysis. This semantic analysis (currently manual) allows us to identify and understand the concepts presented in each requirement, and also separate in the requirement the various restrictions found in it.

SRS	Requirement
3	From the date of expiry of the warranty period, the contractor agrees to provide, at the explicit request of the client, ongoing maintenance services for a minimum period of one year.
4	The proposed solution must be maintained for at least 1 year from the date of expiry of the warranty period.
5	The solution should be maintained for three (3) years from the expiration of the warranty period.

Table 19: Example of requirements expressing the same restriction stated in different ways

At least once a month, project stakeholders will meet to evaluate the good progress of the project. At each steering committee, a statement of progress will be prepared and signed by the parties. A report will be prepared by the provider and approved by the Public Research Centre Henri Tudor, if necessary after recovery.

- At least once a month, project stakeholders will meet to evaluate the good progress of the project.
- At each steering committee, a statement of progress will be prepared and signed by the parties.
- A report will be prepared by the provider and approved by the Public Research Centre Henri Tudor, if necessary after recovery.

Figure 47: Requirement that states more than one restriction

Each requirement is refined later in the catalogue, achieving a summary of the requirements that clears (see table 20):

- The separation of the requirement.
- The purpose of the requirement.
- The assignment of keywords.

Purpose	Requirement	Keywords	SRS	Reference
Documentation Form (format medium)	All documents must be submitted on paper and electronically in a standard format.	documentation, form, medium, format	4	D.6.4
Documentation Form (characteristics for each format medium)	For the electronic medium, a document in an easily exploitable format is required (RTF for text, CSV spreadsheet, JPG or PNG or SVG images, PDF for the other cases).	documentation, form, medium, format, standard, characteristic	4	D.6.4
Documentation Form (characteristics for each format medium)	For the paper format document A4 is required.	documentation, form, medium, format, standard, characteristic	5	D.3.4

Table 20: Requirement refinement examples

7.1.3. Insertion of the candidate patterns in the catalogue

Each refined requirement is inserted in a catalogue of candidate patterns as a candidate requirement. This requires to identify whether the requirement candidate already exists in the catalogue searching coincidences (i.e., identifying similar requirements) based on the keywords.

7.1.3.1 Search of coincidences in the candidates' catalogue

This search of coincidences is made in order to group requirements that later can be formalized as an only pattern (see figure 48). To add the requirement as a candidate in the catalogue we have to validate first if there are coincidences.

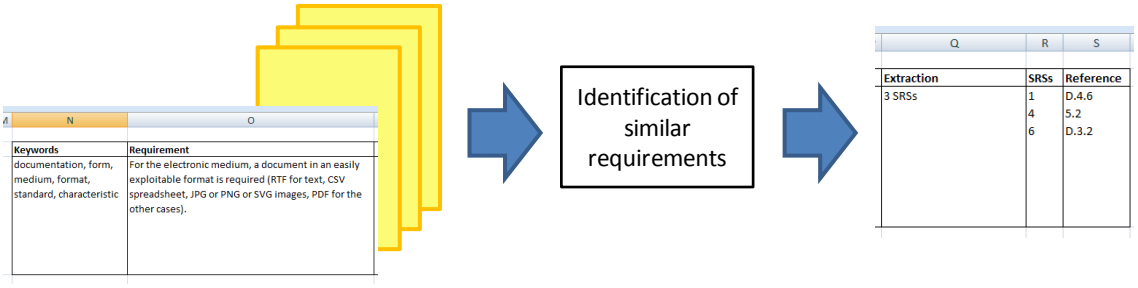


Figure 48: Identification of pattern candidates

7.1.3.2 Coincidences validation and updating or insertion in the candidates' catalogue

In case of not finding coincidences, the requirement is added to the catalogue of candidates as a new candidate (table 21). Then it is analyzed (based on the analysis of requirement type previously done) if the candidate is suitable for being a pattern.

id	#	rel	Purpose	Requirement	Keywords	SRS	Reference
1	0	0	Documentation Form (format medium)	All documents must be submitted on paper and electronically in a standard format.	documentation, form, medium, format	4	D.6.4

Table 21: Catalogue of candidates patterns, example of new candidate

If coincidences are found, the requirement is added as an associated candidate of the requirement to which it matches. This association is reflected by updating the counter of associated candidates and the relationships with the associated ones, writing the identifier of the candidate that has matched (table 22) and increasing the number of occurrences of this candidate, updating in this way the catalogue of candidate patterns. With these coincidences it is possible to obtain the fixed part and the extensions of a pattern iteratively. Then the question of whether a created pattern already exists for the candidate that has matched has to be analyzed.

id	#	rel	Purpose	Requirement	Keywords	SRS	Reference
1	0	0	Documentation Form (format medium)	All documents must be submitted on paper and electronically in a standard format.	documentation, form, medium, format	4	D.6.4
1	1	1	Documentation Form (characteristics for each	For the electronic medium, a document in an easily exploitable format is required (RTF	documentation, form, medium, format, standard,	4	D.6.4

			format medium)	for text, CSV spreadsheet, JPG or PNG or SVG images, PDF for the other cases).	characteristic		

Table 22: Catalogue of candidates patterns, example of associate candidate

7.1.4. Insertion in the patterns catalogue

The insertion in the catalogue of patterns depends on the type of candidate being analyzed, i.e., whether it is a *new candidate* or an *associated* one.

If it is a *new candidate*, once it has been inserted in the catalogue of candidates, we validate it based on the analysis of the type of requirement in order to know if it is suitable to become a pattern. This has to be done because it could be the case that it is a requirement of the organization itself, not necessarily considered basic. As result of this validation, we have:

- If it is not a candidate suitable for being a pattern, the process is finished.
- In case of a candidate that is valid to be a pattern, the requirement is formalized following the pattern structure presented in Section 6. Once we have created the pattern, it is added to the repository of the patterns catalogue.

If it is an *associated requirement*, once it has been inserted in the candidates' catalogue, we must validate whether there is a pattern created for the candidate and their associations in the patterns catalogue. This search of coincidences is made in order to find patterns created previously that match with what is described in the associated candidate.

- In the absence of a created pattern, a new pattern is created following the structure of pattern already defined and stored in the catalogue of patterns.
- If there is a pattern created, it is refined (according to the specifications of refinement patterns described in the next section) and the repository of patterns is updated with the references of the candidate.

7.1.4.1 Pattern refinement

The difference between adding a candidate formalized as a fixed part or as an extension of the pattern depends on the requirement parts of the candidates (C_1, C_2) where both match. That is, having two candidates C_1 and C_2, the pattern P_1 is built the first time, taking into account the fixed part F_1 and the extensions E_1 and E_2 (in case they exist), where (figure 49):

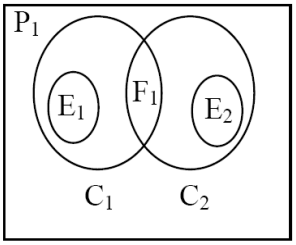


Figure 49: Diagram of the parts of a requirement pattern

- E_1 is a specific part contained only in the candidate C_1 that directly depends on P_1.

- E_2 is a specific part contained only in the candidate C_2 that directly depends on P_1.
- F_1 is the common part of C_1 and C_2.
- P_1 is the set of F_1, E_1 and E_2.

When we need to add a third candidate C_3, we have to determine whether it will be added as part of the fixed part and / or as an extension of the pattern, so we can find three different situations:

- Case 1: C_3 is included entirely in P_1, so we only add the reference of the SRS where it has obtained (figure 50).

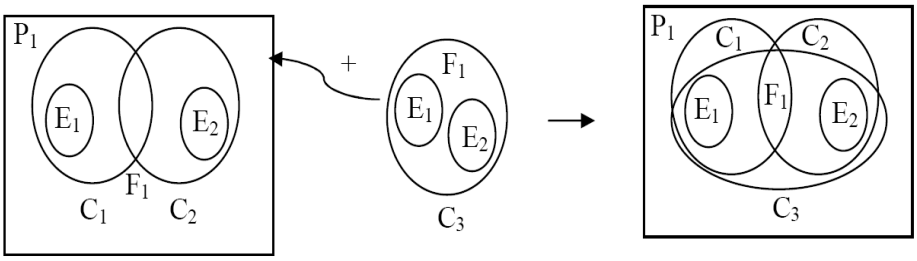


Figure 50: Refinement pattern case 1, candidate is contained in the pattern

- Case 2: C_3 contains the fixed part F_1 of the pattern and a specific part of E_3 that is not contained in P_1. E_3 would be added to P_1 as a new extension (figure 51).

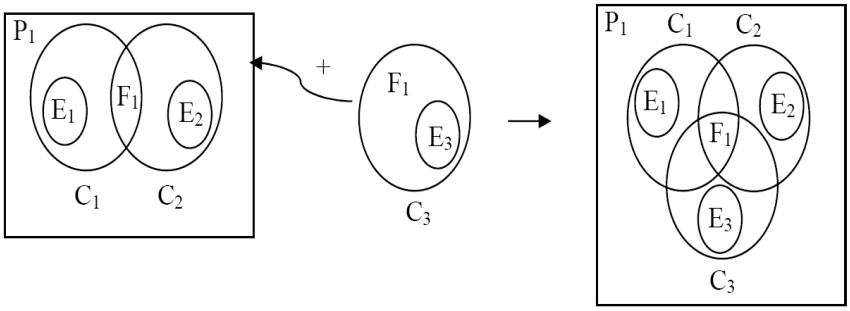


Figure 51: Refinement pattern case 2, a specific part of the candidate is not contained in the pattern

- Case 3: C_3 does not contain the fixed part F_1 of the pattern, but a specific part E that is contained in P_1 as an extension (E_1 or E_2). In this case it has to be analyzed whether F_1 is expressed implicitly in E, and if this is the case we incorporate a reference of the SRS of C_3 to E_1 or E_2, depending on the case (figure 52).

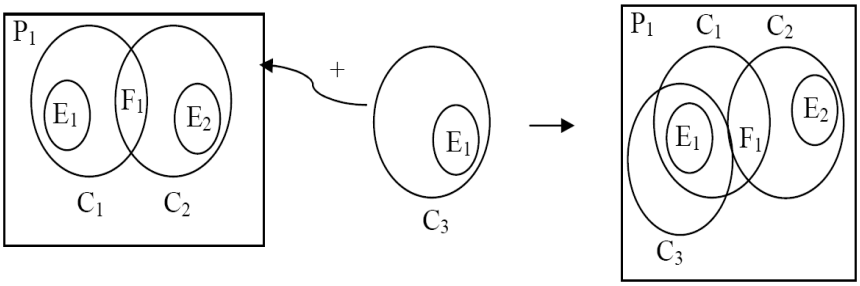


Figure 52: Refinement pattern case 3, a specific part of the candidate is not contained in the pattern and the fixed part of the pattern is not contained in the candidate

8. Developing the Non-Technical part of the Software Requirement Pattern (SRP) Catalogue

As defined in [45], Non-Technical (NT) requirements are those ones that do not refer directly to the intrinsic quality of software, but to the context of the system under analysis; they include economic, political and managerial issues. One of the objectives of this Master Thesis is to include this type of requirements in the SRP existent catalogue already presented in section 6.4, which can be found in [9] completely.

To build this part of the catalogue, we followed the methodology defined in the previous section. As a part of this method, an analysis of NT requirements had to be done, which is presented in section 8.1. After this, in section 8.2, an excerpt of the NT part of the SRP catalogue obtained after applying the method is shown (the complete set of NT SRP obtained is included in Appendix A).

8.1. Non-Technical requirements analysis

Non-technical requirements have always been considered a part of Non-Functional (NF) requirements. A good proof of this is the classification of NF requirements proposed by I. Sommerville in [12] (figure 53), where NT requirements correspond to the category *Process requirements* (and all its subcategories) and *Legislative* and *Cost requirements* subcategories of *External requirements* category.

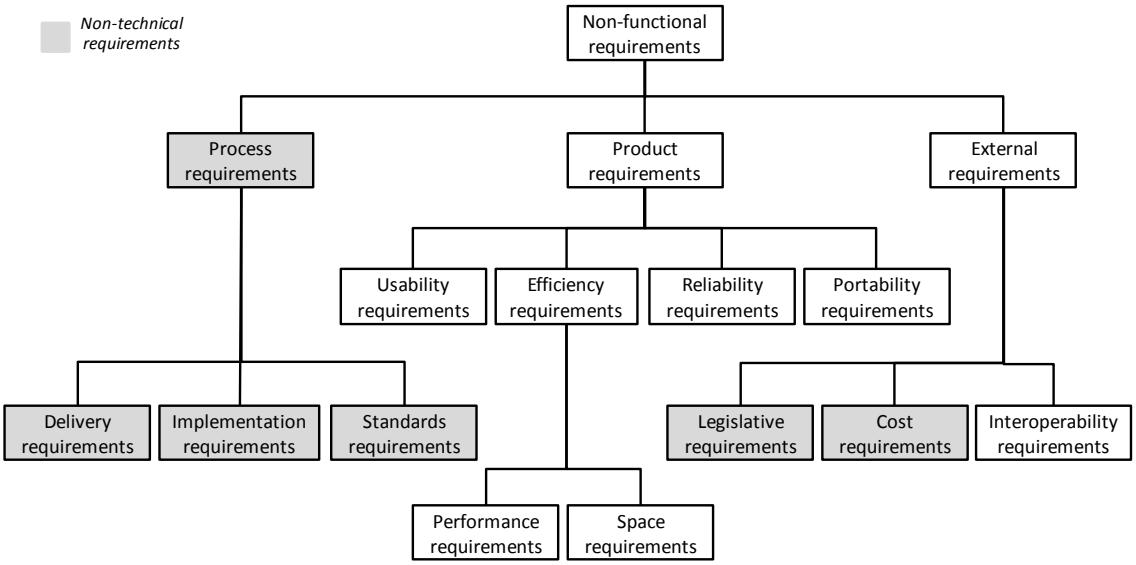


Figure 53: I. Sommerville types of non-functional requirements, with non-technical ones highlighted

If we rearrange Sommerville classification, we arrive to the distinction of non-functional and non-technical requirements that both parts of the SRP catalogue presented in this Master Thesis follow (figure 54).

Despite this usual inclusion of non-technical requirements in non-functional ones, during the last years some works have appeared where these types are distinguished [47, 48, 49]. Among them, J. P. Carvallo et al. proposal [45, 49, 50, 51] is the most consolidated one (figure 55). Their approach has been based on extending the ISO/IEC 9126-1 Quality Model [41] with non-

technical factors, adding to this quality model 126 non-technical features (between subcharacteristics and attributes).

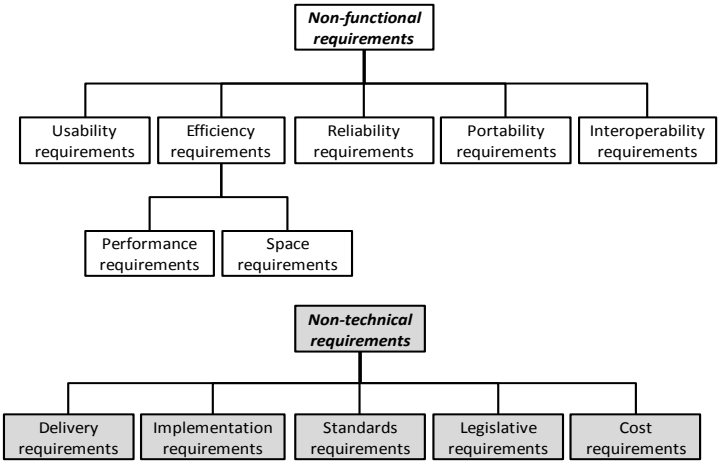


Figure 54: Distinction between non-functional and non-technical requirements

In this Master Thesis we will present the non-technical part of the catalogue organized by just one classification schema. Specifically it is the NT part of the Extended ISO/IEC 9126-1 quality model proposed by J. P. Carvallo et al. (which is completely available in [52]). This classification schema corresponds to the characteristics and subcharacteristics shown in figure 55, which allows classifying the non-technical patterns that we will build.

Selection criteria: Characteristics/subcharacteristics and attributes				Description	Metrics
Supplier	Characteristics of the supplier that can influence the quality of the software product				
	Organizational Structure	Description of the organizational structure of the supplier company			
	Positioning and Strength	Description of the position and orientation of the supplier company in the market			
	Reputation	Recognition of the capability of the supplier to perform similar projects based on past experiences and certifications			
		Supplier Company Existence		Years of the supplier company in the market from its foundation	Number of years: Integer
		Quality Process Certification		Certifications of the quality of the process followed by the supplier company given by recognized certification authorities	Qualification: (Good, Correct, Suitable) Formula for calculating the value from the values of the subattributes
			CMM Level	Capability Maturity Model Level granted to the supplier company	CMM Level: Integer (1 … 5)
			ISO 9000	ISO 9000 Certificate granted to the supplier company	ISO 9000: Boolean
			Other Certificates	Other quality process certificates	List of (Certificate, Level) Certificate: (Spice, SixSigma, …) Level: String
		Client Recommendations		References and recommendations of the supplier company that other clients have given	List of (Client, Comments) Client: String, Comments: List of String
	Services Offered	Description of the services offered by the supplier			
	Support	Description of the support mechanisms offered by the supplier company			
Business	Characteristics of the contract among the supplier and the client that can influence the quality of the software product				
	Licensing Schema	Description of the product-licensing options			
	Ownership	Description of the aspects in relation to the intellectual property rights			
	Guarantees	Detail of the guarantees provided over the product			
	Licensing Costs	Description of the costs components and total cost of ownership for the different licensing options available			
	Platform Cost	Estimation of the cost for the required production platform			
	Implementation Cost	Estimation of implementation costs based on similar past experiences			
	Network Cost	Estimation of additional costs for network operation			
Product	Characteristics of the commercial aspects of the software product that can influence its quality				
	History	Evolution of the product since it has been offered to the clients			
	Deliverables	Detail of the out-of-the-box and expected postimplementation deliverables			
	Parameterization & Customization	Description of the initial effort required for the product to operate			

Figure 55: An excerpt of the non-technical part of the extended ISO/IEC 9126-1 catalogue

8.2. Non-Technical part of the SRP catalogue

To achieve the current version of the non-technical part of the SRP catalogue we follow the method proposed in section 7 using 6 SRSs provided by the SSI-CPPHT department [37]. We have built 38 patterns, two of them showed in tables 23 and 24 (appendix A contains the set of all defined non-technical patterns, showing only a part of their metadata).

Requirement Pattern *Documentation Format*	**Description**	*This pattern expresses the need of following a specific format rules in the delivered documentation.*	
	Comments	---------	
	Pattern goal	*State the format rules of the delivered documentation*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Keywords (0..*)	*Format, Documentation, Deliverables*	
	Dependencies (0..*)	*IMPLIES: Delivered Documentation*	
Requirement Form *Homogeneous Documentation Format*	**Description**	*This form does not establish any relationship among the type of documentation and the type of format.*	
	Comments	*Each extension may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does your organization provide the documentation in specific formats?*
		Form Text	*The documentation should be submitted using FT standard formats.*
		Parameter	**Metric**
		FT: is a non-empty set of format types	*FT: Set (FormatType)* *FormatType: {paper, electronic, ...}*
	Extended Part *Paper Standard Format*	**Question Text**	*Does your organization provide the documentation on paper using a specific format?*
		Form Text	*The documentation on paper should be in PF formats.*
		Parameter	**Metric**
		PF: is a non-empty set of paper formats	*PF: Set(PaperFormat)* *PaperFormat: {A0, A1, A2, ...}*
	Extended Part *Electronic Standard Format*	**Question Text**	*Does your organization provide the electronic documentation using a specific format?*
		Form Text	*The electronic ET documentation should be in EF formats.*
		Parameter	**Metric**
		ET: is a non-empty set of electronic types	*ET: Set(ElectronicType)* *ElectronicType: {image, text, spreadsheet, ...}*
		EF: is a non-empty set of electronic formats	*EF: Set(ElectronicFormat)* *ElectronicFormat: {jpg, xls, doc, ...}*
Requirement Form *Heterogeneous Documentation Format*	**Description**	*This form establishes a dependency among the type of documentation and the type of format.*	
	Comments	*"Formats for Documentation Types" extension may be applied more than once. "Paper standard format" and "Electronic Standard Format" extensions may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does your organization provide documentation in specific formats depending on the type of documentation?*
		Form Text	*The format documentation should be submitted in a standard format depending on the type of documentation.*
	Extended Part *Formats for*	**Question Text**	*What are the possible formats depending on the type of documentation?*

	Documentation Types	**Form Text**	*The DT documentation should be submitted using FT standard format.*
		Parameter	**Metric**
		DT: is a non-empty set of documentation types	*DT: Set(DocumentationType)* *DocumentationType: {user manual, training material, ...}*
		FT: is a non-empty set of format types	*FT: Set (FormatType)* *FormatType: {paper, electronic, ...}*
	Extended Part *Paper Standard Format*	**Question Text**	*Does your organization provide the documentation on paper using a specific format?*
		Form Text	*The documentation on paper should be in PF formats.*
		Parameter	**Metric**
		PF: is a non-empty set of paper formats	*PF: Set(PaperFormat)* *PaperFormat: {A0, A1, A2, ...}*
	Extended Part *Electronic Standard Format*	**Question Text**	*Does your organization provide the electronic documentation using a specific format?*
		Form Text	*The electronic ET documentation should be in EF formats.*
		Parameter	**Metric**
		ET: is a non-empty set of electronic types	*ET: Set(ElectronicType)* *ElectronicType: {image, text, spreadsheet, ...}*
		EF: is a non-empty set of electronic formats	*EF: Set(ElectronicFormat)* *ElectronicFormat: {jpg, xls, doc, ...}*

Table 23: *Documentation Format* Non-Technical SRP

Requirement Pattern *Payment Method*	**Description**	*This pattern expresses the need of following a schedule to do the payment.*	
	Comments	---	
	Pattern goal	*State the payment schedule*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Keywords (0..*)	*Payment, Method, Schedule*	
	Dependencies (0..*)	*IMPLIES: Packaging the solution*	
Requirement Form *Agreed Payment Method*	**Description**	*This form does not establish any fixed schedule to do the payment.*	
	Comments	*Each extension may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does your organization accept a payment schedule that will be agreed with the client?*
		Form Text	*The payment should follow a schedule that will be agreed by the supplier and the client.*
	Extended Part *No automatic payment*	**Question Text**	*Does your organization accept not having automatic payment established?*
		Form Text	*No payment will be automatically established.*
Requirement Form *Established Payment Method*	**Description**	*This form establishes a fixed schedule to do the payment.*	
	Comments	*"Formats for Documentation Types" extension may be applied more than once. "Paper standard format" and "Electronic Standard Format" extension may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does your organization accept a payment schedule provided by the client?*
		Form Text	*The payment should follow a schedule provided by the client.*

	Extended Part *Payment by Stages*	**Question Text**	*Does your organization accept payments done in different stages?*
		Form Text	*The PER% of the payment will be done ST.*
		Parameter	**Metric**
		PER: percentage	*PER: Float >0 and <=100*
		ST: stage	*ST: String (e.g., after all deliveries, during development, ...)*
	Extended Part *Dependent Payments*	**Question Text**	*Does your organization accept dependent payments done by stages?*
		Form Text	*The payments done ST will depend on DP.*
		Parameter	**Metric**
		ST: stage	*ST: String (e.g., after all deliveries, during development, ...)*
		DP: dependency	*DP: String (e.g., the progress of the project, the approval of the minutes, ...)*
	Extended Part *Milestone payment schedule*	**Question Text**	*Does your organization want to provide a payment schedule based on project milestones?*
		Form Text	*The supplier can propose a payment schedule based on project milestones.*
	Extended Part *No automatic payment*	**Question Text**	*Does your organization accept not having automatic payment established?*
		Form Text	*No payment will be automatically established.*

Table 24: *Payment Method* Non-Technical SRP

8.2.1. Experts validation

The set of non-technical patterns has been revised by people of the SSI-CPPHT department, which are experts and have real experience in Requirements Engineering. They have said that the built patterns are a good first version to construct a stable set of non-technical patterns. However, they have given us some feedback and have suggested that another iteration done jointly between SSI-CRPHT and us could be a good idea to improve some aspects:

- They think that the level of detail is different in some of the build patterns: some patterns are too specific and other ones too general. SSI-CRPHT and us agree that this is due to the level of detail of the SRSs from which the pattern set has been built. A deep expert's revision done by both parts in the next month will be a good way to standardize the level of detail among patterns.
- They also propose to cross check the non-technical patterns set with a stable list of non-technical requirements that they use as a template to start their projects in order to check that the patterns set is complete.

Summarizing, the SSI-CRPHT revision let us with the idea that the set of non-technical patterns build is a good first version, but a little future work is needed to deal the above issues.

8.2.2. Non-Technical part of the SRP catalogue classified according the Extended NT-ISO/IEC 9126 quality model

The previous existent catalogue composed of just NF SRP had two classification schemas: the one based on ISO/IEC 9126-1 quality model, and the one based in the Volere requirements classification. In this Mater Thesis, and in order to provide a classification for the NT SRP obtained, we propose to extend the classification schema based on ISO/IEC 9126-1 using the

Extended Non-Technical ISO/IEC 9126 quality model proposed in [45, 49, 50, 51].

What we did, once obtained the set of extracted NT SRP was to classify these patterns on the Extended ISO characteristics and subcharacteristics. All the patterns could be correctly classified less 9 patterns. We studied the semantics of these SRP that could not be classified and we saw that all of them could be organized in a missing subcharacteristic in this quality model. This subcharacteristic would embrace how the business is undertaken between the client and the supplier (i.e., planning, progress control, payment method, etc.). Because of this, we add a new subcharacteristic *Business Planning* to the existing *Business* characteristic. Following the schema presented in figure 55, the definition of *Business Planning* is:

> *"Description of the processes related to the way of doing business (progress control, payment method, planning, etc.) that are undertaken between the client and the supplier."*

Table 25 contains the current set of non-technical requirement patterns classified according the NT-ISO/IEC 9126 quality model adding the new subcharacteristic *Business Planning*.

1. Supplier	
1.1 Organizational Structure	• Supplier Administrative Information • Supplier Organization • Supplier History • People Related to the Project
1.2 Positioning and Strength	• Supplier Economic Information • Supplier Workforce
1.3 Reputation	• Supplier Business Experience • Supplier Quality Certification
1.4 Services Offered	• Analysis • Data Migration • Development
1.5 Support	• Maintenance Procedure • Type of Maintenance
2. Business	
2.1 Licensing Schema	----------
2.2 Ownership	• Job Properties and Intellectual Rights • Privacy
2.3 Guarantees	• Warranty • Reviews
2.4 Licensing Costs	----------
2.5 Platform Costs	----------
2.6 Implementation costs	----------
2.7 Network Costs	----------
2.8 Business Planning	• Meeting Organization • Meeting Minutes • Payment Method • Planning • Progress Control • Project Management Method • Settlement of Disputes • Solution Packaging • Steering Committee
3. Product	
3.1 History	• Product History • Community Support
3.2 Deliverables	• Acceptance Tests • Convention of the documents • Documentation • Documentation Format • Source Code Documented
3.3 Parameterization and Customization	• Final Acceptance • Installation • Release • Start of the Solution • Training

Table 25: Non-technical SRPs classified according the extended NT-ISO/IEC 9126 quality model

9. Developing the Functional part of the Software Requirement Pattern (SRP) Catalogue

Functional requirements are those ones that establish the observable behaviour that must exhibit the system (calculations, manipulations, listings, evolution aspects, etc.), as well as the data types specification which the system has to deal with. One of the objectives of this Master Thesis was to include this type of requirements in the SRP catalogue already presented in section 6.4, which can be found in [9] completely.

Due to the variety of these requirements among the different domains that can be found in software systems (Enterprise Resource Planning (ERP), Computer-Aided Software Engineering (CASE), etc.), in order to incorporate them to the SRP catalogue it is necessary to follow the process presented in section 7 for each one of these domains.

Then, as stated before, our first plan was to develop a complete set of functional SRPs for Human Resources Management domain. However, due to some problems encountered (deeper details can be found in section 9.1) only some examples of functional SRPs could be developed for Content Management System domain (section 9.3).

9.1. Problems encountered

When starting this Master Thesis, one of the objectives was to develop the functional part of the SRP catalogue focused on Human Resource Management (HRM) systems. However, finally we only achieved to extract a little set of functional SRPs for a different domain, which is Content Management Systems (CMS).

The HRM domain was selected because people of SSI-CRPHT department [37] was going to provide new SRSs of the projects that they were going to develop in the following months, which were focused on HRM systems, and that were different from those SRSs used to develop the non-technical part of the catalogue. The problem was that by April SSI-CRPHT only had achieved two projects that were focused on HRM, so we only achieved two SRSs of this domain, which were too few to develop SRPs.

Another encountered problem was the fact that the non-technical part of the catalogue was not finished until early May, basically because the SRSs used to develop this part had more non-technical requirements than expected and that the whole process was done manually.

Due to the lack of SRSs from HRM domain and the lack of time that we had to achieve new SRPs and develop the complete functional part, we decided finally to focus on CMS domain, which was the domain of the SRSs used to develop the non-technical part, and also decided to develop only a few functionalities of this domain.

9.2. Functional domain analysis

A Content Management System (CMS) is used to manage the workflow in a collaborative environment. As a general perspective, the main objectives of CMSs are (figure 56) [59, 60, 61]:

- Allow a large number of people to contribute to and share stored data.

- Control the access to data, based on user roles (defining which information users or user groups can view, edit, publish, etc.).
- Aid in easy storage and retrieval of data.
- Reduce repetitive duplicate inputs.
- Improve communication between users.
- Version control.

The data to be managed in CMSs can be defined as nearly anything: documents, movies, pictures, phone numbers, scientific data, etc. CMSs are frequently used for storing, controlling, revising and publishing documentation and they are primarily used to facilitate the management of sites, whether on the Internet or on an Intranet. Serving as a central repository, the CMS increases the version level of new updates to an already existing file.

Figure 56: Main functionalities of CMSs

J. Robertson [53, 54] proposes a division of CMS functionalities into four categories: content creation, content management, publishing and presentation.

1. **Content Creation.** A CMS provides tools so that developers without experience in web pages can focus on the web content rather than in technical details. The most common is to provide a WYSIWYG (What You See Is What You Get) text editor, in which the user sees the end result as you type, following the style of the commercial publishers, but with a range of limited text formatting. This limitation makes sense, since the aim is that the creator can emphasize some points, but without changing much the overall style of the website.

   There are other tools that can be incorporated to a CMS, such as XML document editors, office applications that integrate with the CMS, import of existing documents, and editors that let you add markings, usually HTML, to indicate the format and structure of a document. A CMS can incorporate one or more of these tools, but it should always provide a WYSIWYG editor due to its ease of use and the convenience of access from any computer with a browser and Internet access.

   To create the website itself, CMS not only provide tools to define the structure, format and look of the pages, but also use patterns and a modular system that can include features not originally anticipated.

2. **Content Management.** The documents created are stored in a central database which also keeps the rest of the data web, such as data relating to documents (versions, author, publication and expiration date, etc...), data and preferences of users, and the structure of the web.

   The structure of the web can be configured with a tool that typically presents a hierarchical view of the site and allows its modification. With this structure a group to each area can be assigned, with publishers, authors and users with different permissions (figure 57). This is essential to facilitate the workflow with an edition circuit that goes from the author to the ultimate responsible for the publication. The CMS allows communication between group members and it also monitors the status of each step of the cycle.

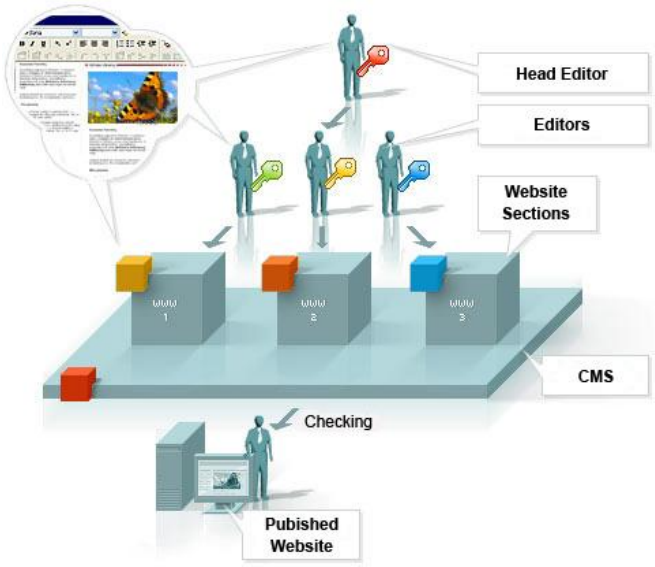


Figure 57: Different types of users in CMSs

3. **Publication.** An approved page is published automatically when the date of publication arrives, and when it expires is archived for future reference. In their publication, the pattern defined for the whole site or the particular section where it is located is applied, so that the end result is a website with a consistent look on every page. This separation between content and form allows modifying the look of a website without affecting the documents already created and release the authors to worry about the final design of pages.
4. **Presentation.** A CMS can automatically manage the accessibility of the web, with support of international standards for accessibility as Web Accessibility Initiative (WAI) [55], and adapt to the needs or preferences of each user. It can also provide compatibility with different browsers available on all platforms (Windows, Linux, Mac, Palm, etc.) and its globalization ability allows adapting it to the language, measurement system and culture of the visitor.

   The system manages many aspects such as navigation menus or the hierarchy of the current page within the site, adding links automatically. It also manages all the modules, internal or external, that the system incorporates. For example, with a news module, the news appeared on another site would be presented, with an

advertising module we would show an ad or animated message, and with a forum module we could display the title of last message received on the principal page. All this is done with the appropriate links and, of course, following the pattern that designers have created.

Some works [60, 61] had defined quality models specific for CMS domain. As any quality model, they are composed of characteristics and subcharacteristics that can be used as a classification schema of the patterns related to this domain in order to improve their accessibility.

9.3. Functional SRP examples

As stated before, due to the problems encountered explained in section 9.1, only some examples of functional SRPs for Content Management System domain have been extracted from 6 SSI-CPPHT SRSs. To build these examples, we also follow the methodology proposed in section 7, but starting with a small set of requirements that were related with two specific functionalities of this type of systems: the first one is the management of versions for the same document and the second one is the ability of doing searches over the objects present in the system. The SRPs build for version management can be found in tables 26 and 27; the SRP for search functionality is found in table 28.

Requirement Pattern *Version Number Management*	**Description**	*This pattern expresses the way of doing the management of the version numbers.*	
	Comments	---------	
	Pattern goal	*State the version number management way.*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Keywords (0..*)	*Version, Management, Number*	
	Dependencies (0..*)	*IMPLIES: Version Management*	
Requirement Form *Manual Version Number*	**Description**	*This form establishes the need of having a version number management done by the users.*	
	Comments	*Each extension may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does the system allow managing the version numbers manually?*
		Form Text	*The version numbers should be managed manually by the users.*
Requirement Form *Automatic Version Number*	**Description**	*This form establishes the need of having a version number management done automatically by the system.*	
	Comments	*Each extension may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does the system manage the version numbers automatically?*
		Form Text	*The version numbers should be managed automatically by the system.*

Table 26: *Version Number Management* Functional SRP

Requirement Pattern *Version Management*	**Description**	*This pattern expresses the need of having a system that manages versions of documents.*	
	Comments	---	
	Pattern goal	*State the functionalities of the version management*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Keywords (0..*)	*Version, Management*	
	Dependencies (0..*)	---	
Requirement Form *General Version Management*	**Description**	*This form establishes the need of having a version management over all documents stored in the system.*	
	Comments	*"Version history availability" extension may be applied more than once. The other extensions may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does the system manage versions of the stored documents?*
		Form Text	*The system should manage versions of the stored documents.*
	Extended Part *Version Retrieval*	**Question Text**	*Does the system allow the retrieval of previous versions of stored documents?*
		Form Text	*The system should allow the retrieval of a previous version of a stored document.*
	Extended Part *Version History*	**Question Text**	*Does the system store a history of the document versions?*
		Form Text	*The system should store a history identifying and tracking the HF in document versions.*
		Parameter	**Metric**
		HF: is a non-empty set of history fields	*HF: Set(HistoryField)* *HistoryField: {author, date, changes, ...}*
	Extended Part *Version History Availability*	**Question Text**	*Does the system allow configuring the visibility of the history fields depending on the type of user?*
		Form Text	*The system should show the HF fields of the version history to UT users.*
		Parameter	**Metric**
		HF: is a non-empty set of history fields	*HF: Set(HistoryField)* *HistoryField: {author, date, changes, ...}*
		UT: is a non-empty set of user types	*UT: Set(UserType)* *UserType: {validator, normal, etc.}*
	Extended Part *Automatic versions*	**Question Text**	*Does the system propose automatically the creation of new versions depending on the changes done in the document?*
		Form Text	*The system should propose automatically the creation of new versions depending on the changes done in the document.*
	Extended Part *Automatic versions*	**Question Text**	*Does the system allow creating a new version of a document as it was a previous one?*
		Form Text	*The system should not allow saving a new version of a document as it was a previous one.*
Requirement Form *Specific Version Management*	**Description**	*This form establishes the need of having a version management over specific documents stored in the system.*	
	Comments	*"Specific Version History" and "Version history availability" extensions may be applied more than once. The other extensions may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does the system manage versions only over specific stored documents?*

		Form Text	*The system should manage versions over specific stored documents.*
	Extended Part *Versioned documents*	**Question Text**	*Does the system allow stating the documents of which doing versioning?*
		Form Text	*The system should do versioning over DT documents.*
		Parameter	**Metric**
		DT: is a non-empty set of documentation types	*DT: Set(DocumentationType)* *DocumentationType: {user manual, training material, ...}*
	Extended Part *Specific Version Retrieval*	**Question Text**	*Does the system allow the retrieval of previous versions of specific stored documents?*
		Form Text	*The system should allow the retrieval of a previous version of DT stored document.*
		Parameter	**Metric**
		DT: is a non-empty set of documentation types	*DT: Set(DocumentationType)* *DocumentationType: {user manual, training material, ...}*
	Extended Part *Specific Version History*	**Question Text**	*Does the system store a history of versions of specific documents?*
		Form Text	*The system should store a history identifying and tracking the HF in versions of DT documents.*
		Parameter	**Metric**
		HF: is a non-empty set of history fields	*HF: Set(HistoryField)* *HistoryField: {author, date, changes, ...}*
		DT: is a non-empty set of documentation types	*DT: Set(DocumentationType)* *DocumentationType: {user manual, training material, ...}*
	Extended Part *Version History Availability*	**Question Text**	*Does the system allow configuring the visibility of the history fields depending on the type of user?*
		Form Text	*The system should show the HF fields of the version history to UT users.*
		Parameter	**Metric**
		HF: is a non-empty set of history fields	*HF: Set(HistoryField)* *HistoryField: {author, date, changes, ...}*
		UT: is a non-empty set of user types	*UT: Set(UserType)* *UserType: {validator, normal, etc.}*
	Extended Part *Specific Automatic versions*	**Question Text**	*Does the system propose automatically the creation of new versions depending on the changes done in specific documents?*
		Form Text	*The system should propose automatically the creation of new versions depending on the changes done in DT documents.*
		Parameter	**Metric**
		DT: is a non-empty set of documentation types	*DT: Set(DocumentationType)* *DocumentationType: {user manual, training material, ...}*
	Extended Part *Version History Availability*	**Question Text**	*Does the system allow configuring the visibility of the history fields depending on the type of user?*
		Form Text	*The system should show the HF fields of the version history to UT users.*

Table 27: *Version Management* Functional SRP

Requirement Pattern *Search*	**Description**	*This pattern expresses the need of having a system that allows searches over different objects of the system.*	
	Comments	---	
	Pattern goal	*State the functionalities of the system search engine*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Keywords (0..*)	*Search, Engine, Query, Sort*	
	Dependencies (0..*)	---	
Requirement Form *Search*	**Description**	*This form establishes the need of having a search engine over different objects of the system.*	
	Comments	*Each extension may be applied just once.*	
	Version	*Friday, 10/06/2011 - 2:25am*	
	Author	*Cristina Palomares*	
	Sources (0..*)	• *Requirements specifications from SSI.*	
	Fixed Part	**Question Text**	*Does the system include a search engine?*
		Form Text	*The system should include a search engine.*
	Extended Part *Search Objects*	**Question Text**	*Does the system allow doing searches only over specific objects?*
		Form Text	*The system should do searches over OT.*
		Parameter	**Metric**
		OT: is a non-empty set of object types	*OT: Set(ObjectType)* *ObjectType: {published material, list of subscribers, online services users , ...}*
	Extended Part *Normal Search Fields*	**Question Text**	*Does the system allow doing searches only over specific fields?*
		Form Text	*The system should do searches over objects using FT fields.*
		Parameter	**Metric**
		FT: is a non-empty set of field types	*FT: Set(FieldType)* *FieldType: {date, author, keyword, type of document, ...}*
	Extended Part *Advanced Metadata Search*	**Question Text**	*Does the system allow doing advanced searches over metadata?*
		Form Text	*The system should do advanced searches over metadata.*
	Extended Part *Multi-criteria search*	**Question Text**	*Does the system allow doing multi-criteria searches?*
		Form Text	*The system should do multi-criteria searches.*
	Extended Part *Complex Boolean Queries*	**Question Text**	*Does the system allow doing complex boolean queries using operators?*
		Form Text	*The system should allow advanced boolean search queries with OpT operators.*
		Parameter	**Metric**
		OpT: is a non-empty set of operator types	*OpT: Set(OperatorType)* *OperatorType: {AND, OR, NOT,NEAR, ...}*
	Extended Part *Sorting Results*	**Question Text**	*Does the system allow sorting the results of searches?*
		Form Text	*The system should allow sorting the results of searches.*
	Extended Part *Exporting Results*	**Question Text**	*Does the system allow exporting the results of searches?*
		Form Text	*The system should allow exporting the results of searches.*
	Extended Part *Specific Search Window*	**Question Text**	*Does the system allow having a specific window for searches?*
		Form Text	*The system should have a specific window for searches.*

Table 28: *Search* Functional SRP

10. The new Software Requirement Patterns (SRP) Metamodel

The current Software Requirement Pattern (SRP) structure (section 6), which is supported by the metamodel showed in section 6.3, has been proven to be also valid for non-technical and functional requirements. Putting special attention in the meaning that Requirement Forms and Fixed and Extended Parts have, the SRPs build in this Master Thesis reinforced the idea that having different contexts or ways (requirement forms) of expressing the restrictions over a concept (requirement pattern) is useful, as well as having inside a requirement form a part that has to be always used (fixed part) and other ones that are optional (extended parts). The examples of the build SRPs shown in the previous sections (tables 23, 24, 26, 27, 28) show how these parts of a SRP are used in these types of requirements.

The only lack that we have found to support completely non-technical and functional requirements is a specific type of *Set Metric*. In the metamodel presented in section 6, a *Set Metric* is understood as the union (AND) of values of its *Simple Metric*. For instance, the extended part *Alert Type* of the non-functional SRP *Failure Alerts* presented in table 17 states *Alerts provided by the solution shall be: AL*, where *AL* is a set of *Alert Types*, being they E-mail, SMS, Page, Fax, etc. When we apply this part in a real project, if we select for instance the alert types E-mail, SMS and Page, the requirement text that would be incorporated to the SRS is *Alerts provided by the system shall be: E-mail, SMS and Page*.

However, when extracting non-technical and functional requirements we found that sometimes a *Set Metric* should be an exclusion (OR), i.e., when we apply this new *Set Metric*, values are concatenated using an OR. Figure 56 contains 2 examples of parts when this new type of metric is used and some of the requirements from which they were extracted.

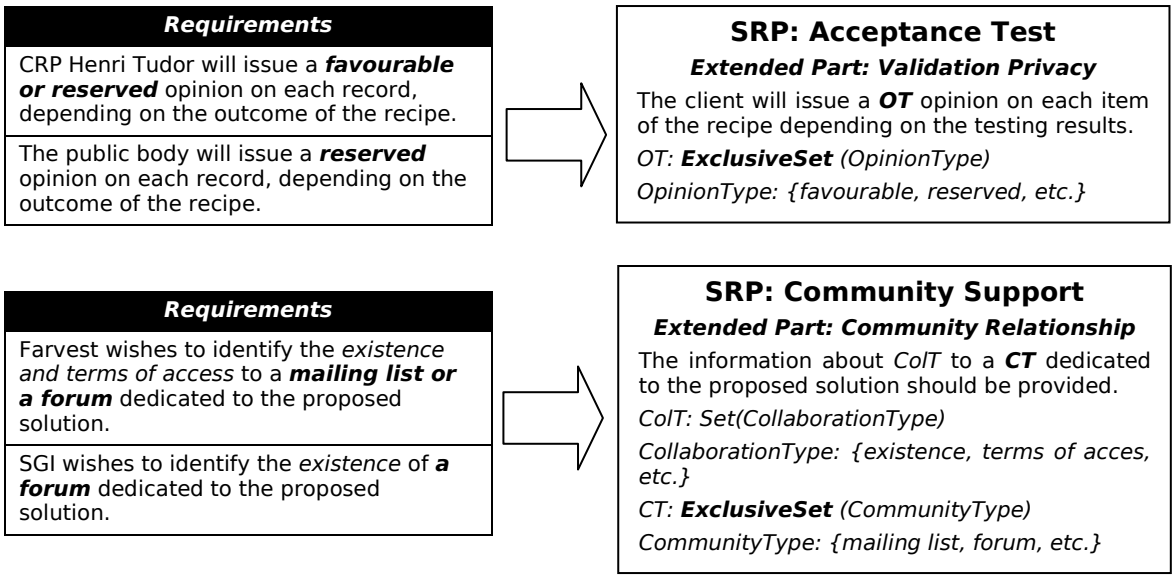


Figure 58: Examples of exclusive Set Metrics

To add this new type of *Set Metric* in the metamodel presented in section 6.3, we consider that the best option was to add a new boolean attribute *exclusive* in the class *Set Metric* that, when setting to true, implies that the set metric is an exclusive set metric (see figure 57).

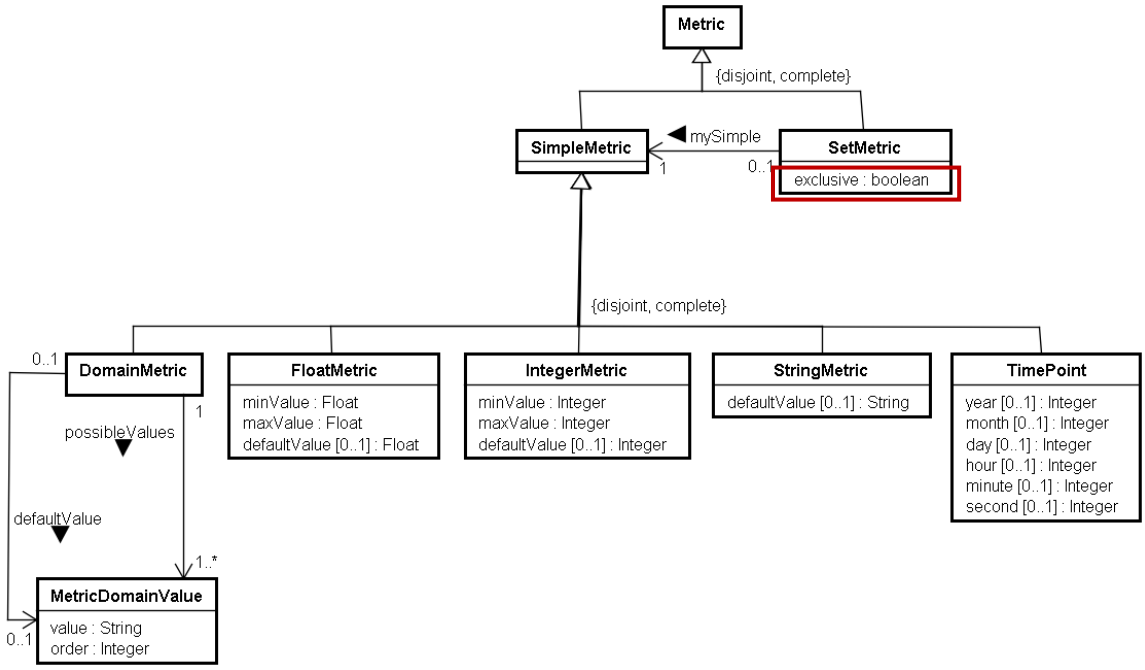


Figure 59: New SRP metrics metamodel

Part III

Validation

11. Retro-analysis of the Non-Functional part of the Software Requirement Pattern (SRP) Catalogue

The basis to construct the Software Requirement Patterns (SRP) of this Master Thesis has been the structure of SRP proposed in the PABRE framework. Because of this, it has been important to validate this structure before starting to construct the non-technical and functional parts of the patterns catalogue of this Master Thesis. Figure 60 contains the SRP structure already presented in section 6.3.

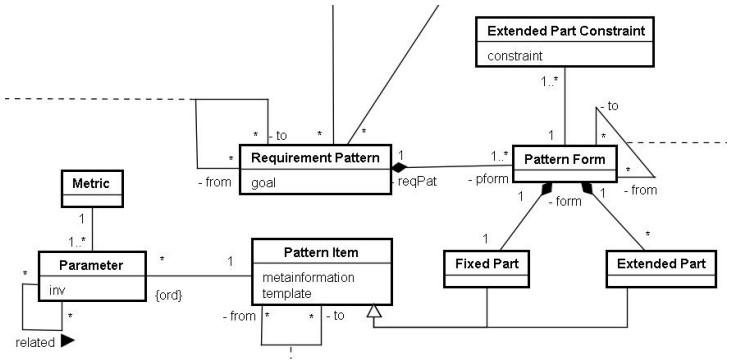


Figure 60: Excerpt of the software requirement pattern metamodel

11.1. Motivation

The process of construction of the metamodel that contains the structure of pattern proposed in the PABRE framework was constructed jointly by SSI-CPPHT department [1] and GESSI research group [37], to which I belong, using three different sources of knowledge, *Software Requirement Specifications (SRSs)*, *Expert's Knowledge* and *Literature Review* (see section 6 for further details). Among them, SRSs were the main source to elaborate the approach.

We grounded our theory on 6 SRSs coming from real projects undertaken by SSI-CPPHT from different domains. Since non-functional requirements are the requirements that are the least context-dependent, we started by extracting patterns from these types of requirements, creating in this way the first version of the non-functional part of the requirement patterns catalogue. These SRSs contained about 70 non-functional requirements in average. In these documents, we observed and analyzed the structure, relationships and organization of requirements, and we obtained the first version of the SRP metamodel shown in section 6.3. However, these observations weren't formally demonstrated.

We thought that it was important, before constructing the new parts of the SRP catalogue, to check if the proposed SRP structure is supported by the requirements in SSI-CPPHT requirement specifications.

11.2. Method

The selected method to do the validation of the SRP structure was the retro-analysis of the 6 SSI-CPPHT requirement specifications used to construct it. First of all, during this analysis we grouped requirements among the concepts they stated restrictions over, such as *Authorization*, *Authentication*, *Adaptation*, and *Ease of Use*, creating clusters of requirements that were related to the same concept in the SRSs. After this

step, we started to make the analysis of those clusters of requirements of different SRSs that are related to the same concept, putting especial attention in issues such as whether there is implicit knowledge or not in the clusters (which normally corresponds to the fixed part of the pattern), if they have parts in common or not (normally extended parts), if there are requirements expressing the same restriction but with little changes (giving the concept of parameter), etc.

The next section gives a more careful view of what were the specific points that the analysis overtook, jointly with the obtained results.

11.3. Obtained Results

The first observation that we found is that there are concepts that appear repeatedly in different Software Requirement Specifications (SRS). A concept is an abstraction of the main idea for which one or more requirements state constraints. Some of the concepts that we have found are *Authorization*, *Concurrent Users Capacity* and *Interface Language*. In the case of *Authorization*, we found requirements related to it in the 6 studied SRSs that restrict in different ways the access to the different resources in a system.

In figure 61, there are the 50 concepts found during the study. It can be seen that the 68%, which corresponds to 34 concepts, appear in more than one SRS. It would be logical to consider these concepts as the ones that can be appearing in future projects. However, we used expert advice to know their opinion on the complete set of concepts. The result has been that the experts consider as reusable concepts: the 85% of the ones that appear in more than one SRS (29 concepts), and just the 19% of the ones that appear in one SRS (3 concepts).

As a consequence, it has sense to have SRPs associated to the concepts that may appear in other projects, although they now appear in just one SRS. The most of the times the concepts for witch SRPs have to be defined appear in more than one SRS, but it could be the case that a concept that appear in just one SRS is considered to have a related pattern (because it could be reused in future projects) or even a concept that appear in more than one SRS is not considered to have a related pattern (because the concept is too specific to become a pattern or its low quality make the conversion impossible).

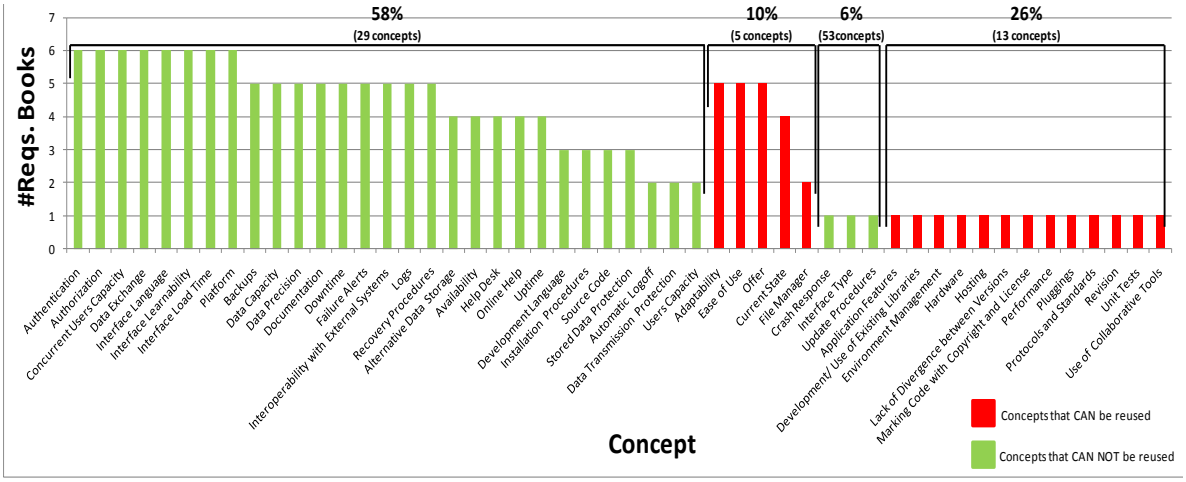


Figure 61: Number of SRSs in which a concept appears

The second observation is that one concept is represented in one SRS document by one cluster of requirements. As defined by Withall, "*a requirement pattern is a guide to writing a particular type of requirement*" [56]. In this sense, a type must be considered as the cluster of requirements related to a concept. For instance, Withall's patterns define more than one requirement in each pattern (see section 4.2.3).

Reinforcing this idea, inspection on SSI-CPPHT SRS document clearly shows that there exist clusters of requirements around concepts. As can be seen in figure 62 we found, in the total study, 177 clusters that include from one to ten requirements, and 70% of them are composed by more than one requirement. One example of concept with more than one cluster of requirements related to it is *Authentication*. One of the clusters related to this concept is: "*The system should authenticate the users, no matter their profile. Authentication should be based on Windows login.*" As it can be seen, there are two requirements in this cluster, one asking for a system that authenticates the users, and the other one that asks that the authentication system is based on Windows login. This observation justifies that SRPs are not simply an abstraction of one requirement, but of a group of requirements related to the concept represented by the pattern.

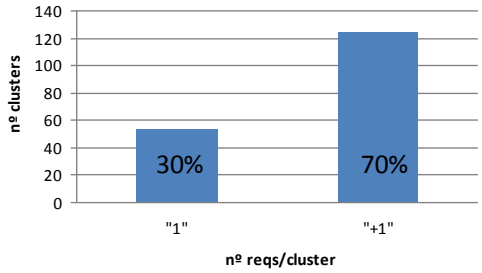


Figure 62: Number of requirements per cluster

The next observation done was that the concept represented by a SRP may take different forms in different requirement books. Inspection on SSI-CPPHT requirement specifications shows this situation several times. Specifically we found that 6 of the 32 concepts are represented by more than one form in the different SRS documents. This corresponds to a 19% of the concepts (see figure 63).

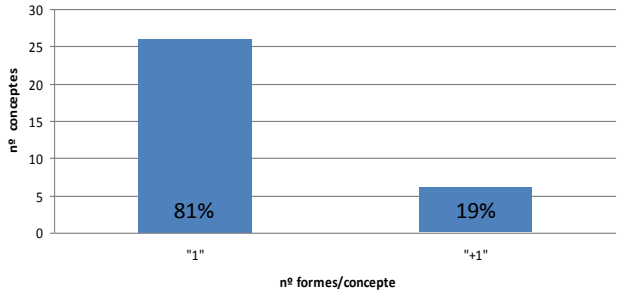


Figure 63: Concepts divided taking into account if they are represented by one or more forms

An example is the requirements related with the concept *Platform*. We found that these requirements took three different forms in the books (see table 29): the first one expresses the general need that the future system has to work on a certain technology (databases, operating systems,...); the second one that asks for one specific type of technology (relational databases, open source operating systems,...); and the third one that refers to the need of

working with a specific external component for the new system (Oracle database, Linux operating system,...). It is important to notice that any of these forms may have sense depending on the particular context that may be the most appropriate for a software project. As a consequence, a *Requirement Pattern* is defined as a composition of *Requirement Forms*.

Form 1: *Platform General Requirement*
The system currently uses MS Exchange/Outlook, but the mail system can be changed if the proposed system requires.
The system currently has no server database. The offer will propose an appropriate server for it.
Form 2: *Platform by Family*
The proposed system should work with a relational database manager.
The solution should be based on a standard web server market.
The solution should be based on (or be integrated with) a standard mail server market.
Form 3: *Platform by Product*
The solution should work with the current client systems where the operations systems are Win32 and Linux32.
The system will be compatible with Lotus Notes Databases.
The new system should use preferably Oracle.

Table 29: Forms detected in the requirements related to *Authorization* in SSI-CPPHT SRS documents

The fourth observation done was that each cluster of requirements, which represents a form of a concept, has a part that characterizes this form, and zero or more optional parts that may appear or not. This situation occurs always in SSI-CPPHT documents for the 100% of the clusters. However, in the 37% of the cases the part that characterizes the form is implicit in the cluster.

This idea of implicit knowledge can be seen in the example of table 30. In one of the SRSs, the requirements related to the concept *Authorization* required that the system had the capacity of defining certain profiles, but did not state the general capacity of defining profiles. If this second requirement is not fulfilled, the first one makes no sense. So, the second requirement is implicit in the cluster. In other documents, this part is explicit. In any case, we may consider that this part (sometimes implicit, sometimes explicit) characterizes the form, because the other parts make no sense if it is not fulfilled.

Authorization		
	(As it appears in the documents)	*(Adding the implicit part)*
SRS 1	The solution should define profiles for access to the contents, which may be configurable by the administrator. It has to manage at least the following profiles: Director, Business Manager, Project Manager, Secretariat, other. The possibility for the administrator to define access permissions for field and feature will be a plus.	There is not an implicit part.
SRS 2	 The solution should permit to generate at least three profiles (administrator, author, validator).	The solution shall allow an administrator to define user profiles that control the authorization mechanisms. It should permit to generate at least three profiles (administrator, author, validator).

Table 30: Example of parts of a cluster of requirements showing implicit knowledge for the *Authorization* concept

Given the previous observation, a *Requirement Form* is a composition of a *Fixed Part* and zero or more *Extended Parts*. The *Fixed Part* is the part that characterizes the form and because of this it is always present in all the applications of one pattern. However, *Extended Parts* are optional and allow adding restrictions about how the fixed part has to be fulfilled.

Another observation done was that optional parts related to a same concept exhibit different behavior in the different SRSs. Specifically we observed two behaviors:

- The case of optional parts expressing the same type of restriction appearing more than once in one cluster of requirements. Of 39 clusters that have more than one optional part, 22 of them follow this rule. For instance, this situation appears in one cluster related to the concept *Backups*. Some of the requirements of this cluster are: "*The system should permit a manual backup of the documents in the system"* and "*If the system works in its own server an automatic backup should exist*." As can be seen, this optional part stating the type of backup on different situations could appear in SRS as many times as required.
- The case of optional parts of the same concept that can not appear together in one cluster (exclusion dependency). We have found 2 *Requirement Forms* that fall into this category. This is the case of the concept *Users Capacity*, that expresses the need of having a certain number of users in the system and what has to be its growing. In two different requirement specifications we found two optional parts related to the same form (the global users capacity and its growing) that follow this rule: *"The system should work with a 50% increment in the number of users."* and *"The system should allow a growing of a minimum of 50 users."* As can be seen, these optional parts cannot appear in the same SRS because they define the growing of users in two different ways: the first one with percentage and the second one with an absolute number of users. It is not possible to give data of how many times this occurs, since it cannot be observed in just one SRS. However, this observation was identified after the analysis of the optional parts made by the experts.

Due to these different behaviours of the optional parts in SRSs, the *Extended Parts* of a *Requirement Form*, which correspond to optional parts in a form of a concept in the SRSs, have a specific behaviour that constraints their application to a certain project. Then, *Requirement Form* is bound to a class *Extended Part Constraint* that states possible behaviors in the application of the form parts. Each behaviour is represented by a formula that should allow at least representing: possible *multiplicity* in the application of an *Extended Part*; *exclusion* in case that the application of some part makes not applicable some other parts of the form.

The last observation done was that requirements related to the same concept form in different SRSs that states the same type of restriction contain specific information related with the context of the project to which the SRS corresponds. Considering the concepts for which experts have considered to have an SRP associated, we analyzed all the requirements appearing in the different books. From this analysis, we found that of the requirements expressing the same type of restriction, there are 65% that follow the template of having a specific text related to the context on the

project. In figure 64 it is highlighted the information dependent of the project for some requirements of the SRSs of the same concept form stating the same restriction.

As a consequence, *Fixed* and *Extended Parts* of a *Requirement Form* have a text that can contain references to *Parameters* representing the "variable information" to be specified for a project. Also, due to the similarity between *Fixed Parts* and *Extended Parts*, an abstract class *Pattern Item* is defined as a superclass of both. This class included the attribute *template* that represents the text of the item. The text contains the references to *Parameters*. Thus in the metamodel the class *Parameter* has an association with *Pattern Item* with an ordered role.

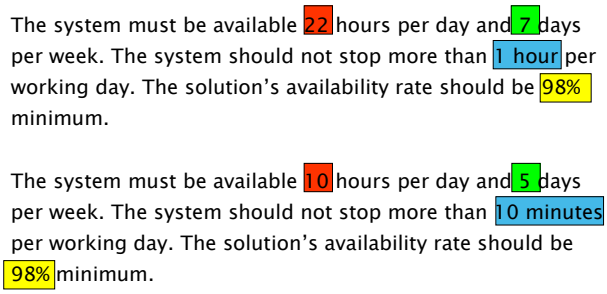


Figure 64: Information dependent on the project of requirements of SRSs of the same concept form stating the same restriction

11.4. Conclusions of the Retro-Analysis

The observations and their later conclusions done during the retro-analysis of the non-functional part of the SRP catalogue had proved that the SRP structure in which this Master Thesis is based is supported by the requirements in SSI-CRPHT requirements specifications.

Each one of the parts of a SRP and the relationships and cardinalities among them had been justified in terms of requirements appearing in requirements specifications, concluding in this way that this Master Thesis is based in an approach which is validated.

12. Survey

During the last months we have been building a survey to know what is the importance that requirements engineers attach to a Software Requirement Patterns (SRP) Catalogue and if its use will be feasible in their respective organizations in order to validate the usability of one in general, but especially the non-functional and non-technical parts.

To construct and execute this survey, the survey process presented in Ciolkowski et al. [57] has been followed. This process defines six different steps:

1. Study definition. Determine the goal of the study.
2. Design. Operationalize survey goal into questions.
3. Implementation. Operationalize design to make survey executable. This step will be avoided in our process because the selected execution method (see section 11.2 for further details).
4. Execution. Data collection and data processing.
5. Analysis. Interpretation of data.
6. Packaging. Reporting the results.

As shown in figure 65, these steps are performed in an iterative manner. Dotted boxes show typical iterations.

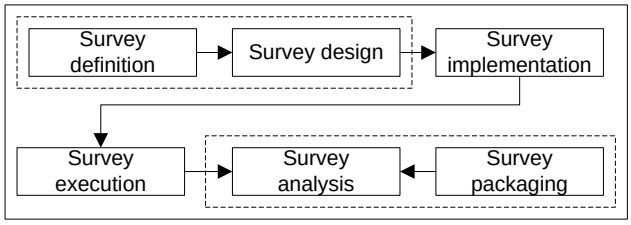


Figure 65: Ciolkowski et al. survey process overview

12.1. Study definition

A full background on requirements reuse using patterns in the academic world was obtained by doing the systematic review (Part I of the Master Thesis). To have a full overview of this type of requirement reuse, and specifically of the requirement pattern approach presented in the Part II of this Master Thesis, it is necessary to consider the perspective of companies and organizations about it. The way chosen to obtain this knowledge was a survey.

It is expected that the responses of the survey will reinforce the hypothesis behind this Master Thesis: “requirements reuse using requirement patterns is possible in organizations, achieving the consequent benefits that it entails”. Also the responses will guide the future evolution of the non-functional and non-technical part of the requirement patterns catalogue, as well as the evolution of the current requirement patterns structure.

12.2. Survey design

As a brief summary of the survey content, the constructed survey has three main sections:

- General information about the interviewee and its organization.

- Information about the context in which the interviewee works. Deeper information about the organization projects, its requirement engineering process and the problems faced during this process, and the role of the interviewee during it; interviewee knowledge about Non-Functional(NF) and Non-Technical (NT) requirements (in order to process better the future results) and analysis of the type of NF and NT requirements used in the organization projects,
- Usability in the organization context of a Software Requirement Patterns Catalogue and feasibility of its incorporation to the organization, as well as the feasibility of adapting to the current requirement pattern structure to requirements stated in the organization projects.

A transcription of the English version of the survey is available in Appendix B.

12.3. Survey execution

The execution of this survey will be done following a guided interview process. The basic idea of this process is that the interviewer meets in person with the interviewee, asking the questions that have been planed during the survey design. It is possible, however, that during the execution of the interview the interviewer changes or adapt some questions depending on the answer given to previous ones. So it is of great importance that the interviewer is a person who knows well the content of the survey and what is the necessary information to gather from the interviewee, as well as having good speaking skills.

In the case of this survey, although it is planned to execute it also in Spain, for the moment people of SSI-CRPHT department [37], with who we are collaborating in the requirement patterns area, will conduct the interview for different requirements engineers of the organizations they work with, which are placed in Luxembourg.

12.4. Current state

At the time of writing, people of the SSI-CPPHT department had not executed any interview yet. They plan to start doing the first interviews among different requirements engineers the first week of July, although most of them will be conducted in September due to holiday reasons.

After these interviews are executed, steps 5 and 6 of Ciolkowski et al. survey process, which correspond to data analysis and data report, will be carried out. As a result, it is planned to publish a paper with the results of the survey before next year.

Part IV

Final Remarks

13. Conclusions

In this Master Thesis we have, first of all, presented a systematic review of the existent published works on reuse in Requirements Engineering, particularly on the use of patterns to achieve the reuse of requirements during Requirements Engineering. The main conclusion is that only two proposals, those ones from S. Withall and X. Franch et al., have a well-established approach of requirements patterns, defining how to use them and having a set of requirement patterns created that can be used in real projects.

The second objective was to create the non-technical and functional parts of the Software Requirement Patterns (SRP) catalogue proposed by X. Franch et al., which before starting this Master Thesis, only embraced non-functional requirements. We have constructed the non-technical part of this catalogue applying a systematic method over requirement specifications corresponding to 6 real projects that SSI-CRPHT gave us. Regarding the functional part, the first idea was to construct it focused on Human Resource Management (HRM) systems domain. However, due to some problems encountered already explained, we have only achieved to extract a little set of functional SRPs for a different domain, which is Content Management Systems (CMS).

After constructing the non-technical and functional parts for this SRP catalogue, we checked the validity of the current SRP metamodel for its suitability for non-technical and functional SRPs. We found that a little change was needed in the metrics part to add a new type of metric and we have applied the changes necessary to the metamodel to reflex this new metric.

Finally, we have validated the structure of SRPs (as it was the base of this thesis) already proposed by X. Franch et al., coming to the conclusion that the SRP structure was supported by real requirements specifications. We also constructed a survey which will be used to know what requirements engineers think about the usability of a SRP catalogue in real projects in their different enterprises or organizations and if it would be applicable or not.

14. Future Work

After the initial validation that SSI-CRPHT department [37] did of the non-technical SRPs obtained in this Master Thesis (see section 8.2.1), they and us agree on doing jointly a second iteration over this set to improve some of its aspects, which will start in the next month. After this second iteration, the SRPs will be introduced in the PABRE tools catalogue, will be included in the existent IPR agreement among CRPHT and UPC, and will be used in trial projects.

Also related to the construction of the SRP catalogue, another goal is to construct in the near future a well-established set of functional SRPs for Content Management Systems (CMS), domain that was already analyzed in this Master Thesis and from which we extracted some examples of SRPs (section 9). This will be necessary in order to validate completely the applicability of patterns in all types of requirements.

A different line of future work is to prove that SRPs are applicable in other contexts different from the SSI-CRPHT department in order to validate that the PABRE framework can be used in any type of project and organizational environment. To do this, it is possible that previously we have to construct other parts for the SRP catalogue (especially for the functional part) and afterwards use the SRP catalogue to elicit requirements in real software projects. Particularly related to the use of SRPs, it is also important to prove that the proposed process [24, 25] to evolve the catalogue is useful in real environments as it has not been proved yet.

As explained in section 12, one of the points of this Master Thesis has been the construction of a survey to know what requirements engineers think about the usability of SRP catalogues in real projects in their different enterprises or organizations and if it will be really applicable or not. In the next months this survey will be executed by SSI-CRPHT people and after this we will have to analyze the results, finishing on this way the survey process shown in the same section.

Finally, the most ambitious point in this future work is the computation of SRPs from requirements specifications in an automatic, or at least semi-automatic, way. As explained through this Master Thesis, the current version of the SRP catalogue was done manually and it took us a lot of time and effort. To use SRPs in different real environments it could be necessary to build different SRP catalogues and if they have to be build manually probably the real use of the PABRE framework and its SRPs could be unfeasible. Because of this we think that is important to use some natural language processing techniques to facilitate the construction of SRP catalogues, achieving a feasible way to use SRPs in industry for any enterprise or type of project.

References

References

[1] Grup de recerca en Enginyeria del Software per als Sistemes d'Informació, *s (Last access: June, 2011).*

[2] Enginyeria de Serveis i Sistemes de la Informació, *http://www.essi.upc.edu/ (Last access: June, 2011).*

[3] Universitat Politècnica de Catalunya, *http://www.upc.edu/ (Last access: June, 2011).*

[4] D. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering", *Computer IEEE*, vol. 39, pp. 25-31, 2006.

[5] B. Kitchenham and S. L. Pfleeger, "Software Quality: The Elusive Target", *IEEE Software*, vol. 13, pp. 12-21, 1996.

[6] E. Yu, P. Giorgini, N. Maiden and J. Mylopoulos, "Social Modelling for Requirements Engineering", *The MIT Press*, 2011.

[7] S. Benbernou, L. Cavallaro, M.S. Hacid, R. Kazhamiakin, G. Kecskemeti, J.-L. Poizat, F. Silvestri, M. Uhlig, B. Wetzstein, "State of the Art Report, Gap Analysis of Knowledge Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs", *Technical Report, S-Cube European Network,* 2008.

[8] I. Sommerville, "Integrated Requirements Engineering: A Tutorial," *IEEE Software*, vol. 22, pp. 16-23, 2005.

[9] PABRE (PAtterns Based Requirement Elicitation) website *http://www.upc.edu/gessi/PABRE/ (Last access: June, 2011).*

[10] C. Palomares, "Implementació del procés d'utilització de patrons de requisits", Final Year Project, 2010.

[11] X. Franch, C. Palomares, C. Quer, S. Renault, F. de Lazzer, "A Metamodel for Software Requirement Patterns", in *Requirements Engineering: Foundations for Software Quality (REFSQ)*, pp. 85-90. Essen, Germany, 2010.

[12] Sommerville, I., "Software Engineering 8", 8th edition, *International Computer Science Series*, 2006.

[13] Prieto-Diaz, R., "Classification of Reusable Modules", in *Concepts and Models. In software Reusability, ACM press*, pp. 99-123, 1989.

[14] Maiden, N., Sutcliffe, A., "Exploiting reusable specification through analogy", *Communications of the ACM*, 35 (4), pp. 55-64. 1993.

[15] Barber, K., Graser, T., Grisham, S., Jernigan, S., "Requirements Evolution and Reuse Using the Systems Engineering Process Activities (SEPA)", *Australian Journal of Information Systems (AJIS)*, 6(2), 1999.

[16] Lam, W., McDermind, J., Vickers, A., "Ten Steps Towards Systematic Requirements Reuse", *Third IEEE International Symposium on Requirements Engineering (RE)*, 1997.

[17] Finkelstein, A., "Reuse of formatted requirements specifications", *Software Engineering*, 3 (5), pp. 186-197, 1988.

[18] Bolton, D., Jones, S., Till, D., Furber, D., Green, S., "Using Domain Knowledge in Requirements Capture and Formal Specification Construction", *Requirements Engineering: Social and Technical Issues, Academic Press*, 1994.

[19] Miriyala, K., Harandi, T., "Automatic Derivation of Formal Software Specification From Informal Descriptions", *IEEE Transactions on Software Engineering*, 17(10), p. 1126-1142, 1991.

[20] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., "Design Patterns: Elements of Reusable Object-Oriented Software", *Addison-Wesley*, 2000.

[21] M. Glinz, "On Non-Functional Requirements", in *IEEE Int. Requirements Engineering Conf. (RE)*, New Delhi, India, 2007.

[22] Yu, E., "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE)*, 1997.

[23] O. Mendez-Bonilla, X. Franch, C. Quer, “Requirements Patterns for COTS Systems”, *International Conference on Composition-Based of Software Systems (ICCBSS)*, pp.232-234, 2008.

[24] S. Renault, O. Mendez-Bonilla, X. Franch, C. Quer, “PABRE: PAttern-Based Requirements Elicitation”, in *Third International Conference on Research Challenges in Information* Science (RCIS), 2009.

[25] S. Renault, O. Mendez-Bonilla, X. Franch, C. Quer, “A pattern-based method for building requirements documents in call-for-tender processes”, *International Journal of Computer Science & Applications* (IJCSA), 6(5), pp. 175-202, 2009.

[26] B. Kitchenham, "Procedures for performing systematic reviews", *Keele University TR/SE-0401/NICTA Technical Report 0400011T*, vol. 1, 2004.

[27] Google Scholar, *http://scholar.google.com/, (Last access: June, 2011)*.

[28] IEEE Xplore, *http://ieeexplore.ieee.org, (Last access: June, 2011)*.

[29] ACM, *http://portal.acm.org/, (Last access: June, 2011)*.

[30] Springer, *http://www.springerlink.com/, (Last access: June, 2011)*.

[31] ISI Web of Knowledge, *http://www.accesowok.fecyt.es/wos/, (Last access: June, 2011)*.

[32] Science Direct, *http://www.sciencedirect.com/, (Last access: June, 2011)*.

[33] RefWorks, *http://www.refworks.com/, (Last access: June, 2011)*.

[34] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, “Non-Functional Requirements Software Engineering”, *Springer*, 2000.

[35] X. Franch, C. Guerlain, C. Palomares, C. Quer, S. Renault, “Interested in Improving Your Requirements Engineering Process? Try Requirement Patterns!”, in *Empirical Fair Track on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2011.

[36] C. Alexander, “The Timeless Way of Building”, *Oxford Books*, 1979.

[37] CRPHT Service Science and Innovation Department (SSI-CPPHT), *http://www.ssi.tudor.lu/, (Last access: June, 2011)*.

[38] Centre de Recherche Publique Henri Tudor (CRPHT), *http://www.tudor.lu/, (Last access: June, 2011)*.

[39] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, “Non-functional Requirements in Software Engineering”, *Kluwer Publishing*, 2000.

[40] A. Egyed, P. Grünbacher, “Identifying Requirements Conflicts and Cooperation: How Quality Attributes and Automated Traceability Can Help”, *IEEE Software*, 21(6), November-December 2004.

[41] “ISO/IEC Standard 9126-1”, *Software Engineering – Product Quality – Part 1: Quality Model*, 2001.

[42] S. Robertson, J. Roberson, “Mastering the Requirements Process”, *Addison-Wesley*, 1999.

[43] C. Palomares, C. Quer, X. Franch, “PABRE-Man: Management of a Requirement Patterns Catalogue”, accepted in International *Symposium on Requirements Engineering (RE)*, 2011.

[44] C. Rolland, N. Prakash, “Matching ERP System Functionality to Customer Requirements”, *IEEE Computer*, pp.66-75, 2001.

[45] J.P. Carvallo, X. Franch, “Extending the ISO/IEC 9126-1 quality model with non-technical factors for COTS components selection”, in *International workshop on Software Quality (WoSQ)*, 2006.

[46] C. Ebert, “Dealing with nonfunctional requirements in large software systems”, *Annals of Software Engineering*, 3(1), pp.367-395, 1997.

[47] J. Leiwo, Y. Zheng, "A framework for the management of information security", *Information Security (Lecture Notes in Computer Science)*, 1396, pp.232-245, 1998.

[48] K. El. Emam, N.H. Madhavji, “A field study of requirements engineering practices in information systems development”, in *IEEE International Symposium on Requirements Engineering (RE)*, 1995

[49] J.P. Carvallo, X. Franch, C. Quer, “Managing Non-Technical Requirements in COTS Components Selection”, in *IEEE International Symposium on Requirements Engineering (RE)*, 2006.

[50] J. P. Carvallo, X. Franch, C. Quer, “Determining Criteria for Selecting Software Components: Lessons Learned”, *IEEE Software*, 24(3), pp. 84-94, 2007.

[51] J. P. Carvallo, X.Franch, C. Quer, “Supporting CMMI Level 2 SAM PA with Non-technical Features Catalogues”, *Software Process: Improvement and Practice*, 13(2), pp. 171-182, 2008.

[52] Extended Non-Technical ISO/IEC 9126-1 Catalogue, *http://www.essi.upc.edu/~gessi/QMTool/CQM/ExtNTISO.html*, *(Last access: June, 2011)*.

[53] J. Robertson, “Looking towards the future of content management”, *Step Two Designs*, January 2003.

[54] J. Robertson, “So, what is a content management system?”, *Step Two Designs*, June 2003.

[55] Web Accessibility Initiative (WAI), *http://www.w3.org/WAI/*, *(Last access: June, 2011).*

[56] S. Withall, “Software Requirements Patterns”, *Barnes & Noble*, 2008.

[57] M. Ciolkowski, O. Laitenberger, S. Vegas, and S. Biffl, "Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering," in *Empirical Methods and Studies in Software Engineering*, R. Conradi and A. I. Wang, Eds. Berlin / Heidelberg: Springer Verlag, 2003, pp. 104-128.

[58] C. Carreon, “Construcción de un catálogo de patrones de requisitos funcionales para ERP”, *Master Thesis*, June 2008.

[59] X. Franch, C. Quer, J. A. Cantón, R. Salietti, “Experience Report on the Construction of Quality Models for Some Content Management Software Domains”, in *International Conference on Composition-Based Software Systems (ICCBSS)*, 2008.

[60] J.A.Cantón, “Construcción de un modelo de calidad para herramientas DMS / ECM”, *Final Year Project*, 2006.

[61] R. Salietti i Anglarill, “Definició d’un model de qualitat per a eines de gestió de continguts web”, *Final Year Project*, 2007.

[62] i* web page, *http://www.cs.toronto.edu/km/istar/#Overview*, *(Last access: June, 2011)*.

[63] I. Sommerville, P. Sawyer, “Requirements Engineering: A good practice guide,” *John Wiley and Sons*, 1997.

Systematic Literature Review References

[SLR1] M. A. Laguna, O. López, Y. Crespo, “Reuse, Standardization, and Transformation of Requirements”, in *Software Reuse: Methods, Techniques and Tools: 8th International Conference (ICSR)*. Madrid, Spain, 2004.

[SLR2] M.A. Martínez, J.A. Toval Álvarez}, “COTSRE: A COmponenTs Selection Method Based on Requirements Engineering”, in *Seventh International Conference on Composition-Based Software Systems (ICCBS)*. Madrid, Spain, February 2008.

[SLR3] M. Ilyas, J. Kung, “A Similarity Measurement Framework for Requirements Engineering”, in *Fourth International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, 2009.

[SLR4] O. Roudies, M. Fredj, “A Reuse Based Approach for Requirements Engineering}, in *ACS / IEEE International Conference on Computer Systems and Applications (AICCSA)*. Beirut, Lebanon, June 2001.

[SLR5] A.G. Sutcliffe, N.A.M. Maiden, S. Minocha, D. Manuel, “Supporting Scenario-Based Requirements Engineering”, *IEEE Trans. Software Eng. Journal*, 24(12), pp.1072-1088, 1998.

[SLR6] M.Daneva, “Evaluating the value-added benefits of using requirements reuse metrics in ERP projects”, in *Symposium on Software Reusability: putting software reuse in context (SSR)*, 2001.

[SLR7] M. Fredj, O. Roudies, “A Pattern Based Approach for Requirements Engineering”, in *DEXA Workshop*, 1999.

[SLR8] H. Gomaa, “Reusable software requirements and architectures for families of systems”, *Journal of Systems and Software*, 28(3), pp.89-202, 1995.

[SLR9] I. Markovic, A. Costa Pereira, “Towards a Formal Framework for Reuse in Business Process Modeling”, in *Business Process Management Workshops (BPM)*. Brisbane, Australia, 2007.

[SLR10] B. Gallina, N. Guelfi, “A Template for Requirement Elicitation of Dependable Product Lines”, in *Requirements Engineering: Foundation for Software Quality (REFSQ)*. Trondheim, Norway, June 2007.

[SLR11] F. Kiedaisch, M. Pohl, J. Weisbrod, S. Bauer, S. Ortmanns, “Requirements Archaeology: From Unstructured Information to High Quality Specifications”, in *International Symposium on Requirements Engineering (RE)*. Toronto, Canada, August 2001.

[SLR12] P.W. Fach, “Design Reuse through Frameworks and Patterns”, *IEEE Software*, 18(5), pp.71-76, 2001.

[SLR13] O. Al-Shara, “Usability tri-parametric artifacts: a knowledge reporting artifact for software development reusability”, *IEEE International Conference on Computer Systems and Applications (AICCSA)*, 2005.

[SLR14] C. Paredes, J. Luiz Fiadeiro, “Reuse of Requirements and Specifications: A Formal Framework”, *ACM SIGSOFT Symposium on Software Reusability (SSR)*, 1995.

[SLR15] A. G. Sutcliffe, N. A. M. Maiden, “The Domain Theory for Requirements Engineering”, *IEEE Transaction on Software Engineering*, 24(3), pp.174-196, 1998.

[SLR16] Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, H. Mei, “An Online Monitoring Approach for Web Service Requirements”, *IEEE Transactions on Services Computing*, 2(4), pp.338-351, 2009.

[SLR17] K. Rocha De Oliveira, M. De Mesquita Spínola, “POREI: patterns-oriented requirements elicitation integrated -- proposal of a metamodel patterns-oriented for integration of the requirement elicitation process”, in *Proceedings of the Euro American conference on Telematics and Information Systems (EATIS)*. Faro, Portugal, May 2007.

[SLR18] A. Monzon, “A Practical Approach to Requirements Reuse in Product Families of On-Board Systems”, in *IEEE International Requirements Engineering Conference (RE)*, 2008.

[SLR19] C.W. Lu, C.W. Chu, C.H. Chang, and C.H. Wang, "A Model-based Object-oriented Approach to Requirement Engineering (MORE)", *Annual International Computer Software and Applications Conference*, IEEE Computer Society Press, July 2007.

[SLR20] C.W. Chu, C.H. Chang, D. Hsu, C.W. Lu and N.L. Hsueh, "Model-based Object-oriented Requirement Engineering (MORE) to Support Software Maintenance and Integration," *Proceedings of the APSEC2007 AGTSE Workshop*, 2007.

[SLR21] M. Mainnon, H. Kaindl, “Using parameters and discriminants for product line requirements”, *Systems Engineering Journal*, 11(1), pp.61-80, February 2008.

[SLR22] A. Durán Toro, B. Bernard Jimenez, A. Ruiz Cortés, M. Toro Bonilla, "A Requirements Elicitation Approach Based in Templates and Patterns", in *Workshop em Engenharia de Requisitos (WER)* , 1999.

[SLR23] S. Withall, “Software Requirements Patterns”, *Barnes & Noble*, 2008.

[SLR24] T. Tsumaki, “Requirements Engineering Pattern Structure”, in *Asia-Pacific Software Engineering Conference (APSEC)*, 2004.

[SLR25] C. W. Lu, C. H. Chang, W.C. Chu, Y. W. Cheng, H.-C. Chang, “A Requirement Tool to Support Model-Based Requirement Engineering”, in *IEEE International Computer Software and Applications Conference* (*COMPSAC),* 2008.

[SLR26] C. Peper, R. Gotzhein, M. Kronenburg, “A Generic Approach to the Formal Specification of Requirements”, in *IEEE International Conference on Formal Engineering Methods (ICFEM)*, 1997.

[SLR27] R. Gotzhein, M. Kronenburg, C. Peper, “Reuse in Requirements Engineering: Discovery and Application of a Real-Time Requirement Pattern”, in *International Symposium Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT)*, 1998.

[SLR28] V. de Freitas Sodré, J. Lisboa Filho, V. Moysés Vilela, M. Vinícius Alvim Andrade, “Improving Productivity and Quality of GIS Databases Design using an Analysis Pattern Catalog”, *Second Asia-Pacific Conference on Conceptual Modelling (APCCM)*, 2005.

[SLR29] G. Grosz, “Building Information System Requirements Using Generic Structures”, in *Computer Software & Applications Conference (COMPSAC)*, 1992.

[SLR30] K. Watahiki, M. Saeki, “Scenario Patterns Based on Case Grammar Approach”, *IEEE International Symposium on Requirements Engineering (RE)*, 2001.

[SLR31] K. Watahiki, M. Saeki, “Scenario evolution in requirements elicitation processes: scenario pattern and framework approach”, *International Workshop on Principles of Software Evolution (IWPSE)*, 2001.

[SLR32] W. Liu, K. He, K. Zhang, J. Wang, “Combining Domain-Driven Approach with Requirement Assets for Networked Software Requirements Elicitation”, *IEEE International Conference on Semantic Computing (ICSC)*, 2008.

[SLR33] B. Estes, D. Pritchard, “Managing system design requirements information”, *International Carnahan Conference on Security Technology*, 2004.

[SLR34] J. Jensen, I. Anne Tøndel, M. Gilje Jaatun, P. Håkon Meland, H. Andresen, “Reusable Security Requirements for Healthcare Applications”, *International Conference on Availability, Reliability and Security (ARES)*, 2009.

[SLR35] W. Lam, “Developing component-based tools for requirements reuse: a process guide”, *International Workshop on Software Technology and Engineering Practice (STEP)*, 1997.

[SLR36] W. Lam, “Achieving requirements reuse: A domain-specific approach from avionics”, *Journal of Systems and Software*, 38(3), pp.197-209, 1997.

[SLR37] W. Lam, S. Jones, “Mechanising Requirements Engineering: Reuse and the Application of Domain Analysis Technology”, *International Conference on Automated Software Engineering (ASE)*, 1997.

[SLR38] L. Chung, S. Supakkul, “Capturing and reusing functional and non-functional requirements knowledge: A goal-object pattern approach”, *IEEE International Conference on Information Reuse and Integration (IRI)*, 2006.

[SLR39] A. A. Issa, A. Al-Ali, “Use Case Patterns Driven Requirements Engineering”, *International Conference on Computer Research and Development (ICCRD)*, 2010.

[SLR40] M. Saeki, “Patterns and Aspects for Use Cases: Reuse Techniques for Use Case Descriptions”, *International Conference on Requirements Engineering (ICRE)*, 2000.

[SLR41] M. Saeki, “Reusing Use Case Descriptions for Requirements Specification: Towards Use Case Patterns”, *Asia-Pacific Software Engineering Conference (APSEC)*, 1999.

[SLR42] J.A. Toval, J. Nicolás, B. Moros, F. Garcia, “Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach”, *Requirements Engineering*, 6(4), pp.205-219, 2002.

[SLR43] R. S. Wahono, J. Cheng, “Extensible Requirements Patterns of Web Application for Efficient Web Application Development”, *International Symposium on Cyber Worlds (CW)*, 2002.

[SLR44] M. Mannion, B. Keepence, D. Harper, “Using Viewpoints to Define Domain Requirements”, *IEEE Software* ,15(1), pp.95-102, 1998.

[SLR45] M. Mannion, H. Kaindl, J. Wheadon, “Reusing Single System Requirements from Application Family Requirements”, *International Conference on Software Engineering (ICSE)*, 1999.

[SLR46] M. Mannion, H. Kaindl, “Requirements-based product line engineering”, *European Software Engineering Conference held jointly with ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC / SIGSOFT FSE)*, 2001.

[SLR47] S. Konrad, B. H. C. Cheng, “Requirements Patterns for Embedded Systems”, *IEEE Joint International Conference on Requirements Engineering (RE)*, 2002.

[SLR48] O. Mendez-Bonilla, X. Franch, C. Quer, “Requirements Patterns for COTS Systems”, *International Conference on Composition-Based of Software Systems (ICCBSS)*, pp.232-234, 2008.

[SLR49] S. Renault, O. Mendez-Bonilla, X. Franch, C. Quer, “PABRE: PAttern-Based Requirements Elicitation”, *in Third International Conference on Research Challenges in Information Science (RCIS)*, 2009.

[SLR50] X. Franch, C. Palomares, C. Quer, S. Renault, F. de Lazzer, “A Metamodel for Software Requirement Patterns”, *in Requirements Engineering: Foundations for Software Quality (REFSQ)*, pp. 85-90. Essen, Germany, 2010.

[SLR51] S. Supakkul, T. Hill, E. Oladimeji, L. Chung, “Capturing, Organizing, and Reusing Knowledge of NFRs: An NFR Pattern Approach", *International Workshop on Managing Requirements Knowledge (MaRK) in conjunction with RE*, 2009.

[SLR52] S. Supakkul, L. Chung, "Visualizing Non-Functional Requirements Patterns", *International Workshop on Requirements Engineering Visualization (REV) in conjunction with RE*, 2010.

[SLR53] S. Supakkul, T. Hill, L. Chung, T.T. Tun, J.C.S.P. Leite, "An NFR Pattern Approach to Dealing with NFRs", *IEEE International Requirements Engineering Conference (RE)*, 2010.

[SLR54] R. Darimont, A. van Lamsweerde, “Formal refinement patterns for goal-driven requirements elaboration”, *ACM Symposium on Foundations of Software Engineering (SIGSOFT)*, 1996.

[SLR55] N. Heumesser, F. Houdek, "Towards systematic recycling of systems requirements", in *International Conference on Software Engineering (ICSE)*, 2003.

[SLR56] J. Dehlinger, R. R. Lutz, "A Product-Line Approach to Promote Asset Reuse in Multi-agent Systems", in *Software Engineering for Multi-Agent Systems (SELMAS)*, 2005.

[SLR57] J. Dehlinger, R. R. Lutz, "A product-line requirements approach to safe reuse in multi-agent systems", *ACM SIGSOFT Software Engineering Notes*, 30(4), pp.1-7, 2005.

Appendix

Appendix A: Non-Technical part of the Software Requirement Patterns Catalogue

1. Supplier Administrative Information

Supplier Administrative Information			
Goal: Provide supplier contact details			
Requirement Form *Detailed Administrative information of the supplier*	**Fixed Part**	**Form Text**	Specific administrative information of the supplier should be provided.
	Extended Part *Specific details of the company*	**Form Text**	DT details of the company should be provided.
		Param	**Metric**
		DT: is a non-empty set of details types	DT: set(DetailsType) DetailsType: {name, address, telephone, etc.}
	Extended Part *Specific details of the person who draft the offer*	**Form Text**	DT details of the person who drafted the offer should be provided.
		Param	**Metric**
		DT: is a non-empty set of details types	DT: set(DetailsType) DetailsType: {name, address, telephone, etc.}
Requirement Form *General Administrative information of the supplier*	**Fixed part**	**Form Text**	Specific details about SP should be provided.
		Param	**Metric**
		SP: is a non-empty set of supplier parts	SP: set (SupplierPart) SupplierPart: {company, person who draft the offer, etc.}

2. Supplier Workforce

Supplier Workforce			
Goal: Provide supplier workforce information			
Requirement Form *Workforce of the supplier*	**Fixed Part**	**Form Text**	Workforce information of the company should be provided.
	Extended Part *Company size*	**Form Text**	Size information of the company on DA should be provided.
		Param	**Metric**
		DA: date	DA: date
	Extended Part *Workforce distribution*	**Form Text**	Information about the distribution of the workforce that the company trades between the TeT teams should be provided.
		Param	**Metric**
		TeT: is a non-empty set of teams type	TeT: Set (TeamType) TeamType: {development, maintenance, etc.}
	Extended Part *Number of deploying sites*	**Form Text**	Information about the number of sites and the location where the company is deploying should be provided.

3. Supplier Economic Information

Supplier Economic Information			
Goal: Provide supplier economic details			
Requirement Form *Economic information of the company*	**Fixed Part**	**Form Text**	Specific economic information of the company should be provided.
	Extended Part *Effective of the company*	**Form Text**	Company's effective information on the last NAT TT should be provided.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Turnover of the company*	**Form Text**	Company's turnover information on the last NAT TT should be provided.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Net income of the company*	**Form Text**	Company's net income information on the last NAT TT should be provided.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
Requirement Form *Economic information of the consortium*	**Fixed Part**	**Form Text**	Specific economic information of the consortium of which the supplier is part should be provided.
	Extended Part *Effective of the consortium*	**Form Text**	Consortium's effective information on the last NAT TT for each company associated to the consortium should be provided.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Turnover of the consortium*	**Form Text**	Consortium's turnover information on the last NAT TT for each company associated to the consortium should be provided.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Net income of the consortium*	**Form Text**	Consortium's net income information on the last NAT TT for each company associated to the consortium should be provided.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}

4. Supplier Organization

Supplier Organization			
Goal: Provide supplier organization chart			
Requirement Form *Organization of the supplier*	**Fixed Part**	**Form Text**	An organization chart of your company should be provided.

5. Supplier Business Experience

Supplier Business Experience			
Goal: Provide information about supplier experience			
Requirement Form *Experience related with the project*	**Fixed Part**	**Form Text**	The LT list of the comapny regarding the SolutionType proposed solution should be provided.
		Param	**Metric**
		LT: is a non-empty set of list types	LT: set (ListType) ListType: {maintenance, dissemination, etc.}
		SolutionType	SolotionType: {the same, the most similar, etc.}
Requirement Form *General experience*	**Fixed Part**	**Form Text**	The global list of the companny regarding the major services or supplies should be provided.
	Extended Part *Specific Time*	**Form Text**	The list have to contain services or supplies over the last NAT TT.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Information for each item*	**Form Text**	We want to know for each service or supply: IT.
		Param	**Metric**
		SI: is a non-empty set of inforation type	IT: Set (InformationType) InformationType: {customer name, amount of benefit, etc.}

6. Supplier Quality Certification

Supplier Quality certification			
Goal: Provide information about supplier quality certification			
Requirement Form *Quality certification*	**Fixed Part**	**Form Text**	The quality certification of the company should be provided.
	Extended Part *Specific information*	**Form Text**	The CI information regarding certification(s) should be provided.
		Param	**Metric**
		CI: is a non-empty set of certification information	CI: Set (CertificationInformation) CertificationInformation: {type of certification, name of the certifying body, date of certification, etc.}

7. People Related to the Project

<table>
<tr><th colspan="4">People Related to the Project</th></tr>
<tr><td colspan="4">Goal: Provide information and restrict the people related to the project</td></tr>
<tr><td rowspan="5">Requirement Form
People related to the project in general</td><td rowspan="2">Fixed Part</td><td>Form Text</td><td>DT information of people assigned to the project should be provided.</td></tr>
<tr><td>Param</td><td>Metric</td></tr>
<tr><td>DT: is a non-empty set of details types</td><td>DT: set(DetailsType)
DetailsType: {name, address, telephone, role, CVs, etc.}</td></tr>
<tr><td rowspan="3">Extended Part
Previous experience</td><td>Form Text</td><td>People related to the project should justify a NAT TT experience on ET.</td></tr>
<tr><td>Param</td><td>Metric</td></tr>
<tr><td>NAT: natural
TT: time type
ET: is a non-empty set of experience type</td><td>NAT: integer > 0
TT: {years, months, etc.}
ET: Set(ExperienceType)
ExperienceType: {technologies used on the project, Java, C++, etc.}</td></tr>
<tr><td rowspan="8">Requirement Form
Specific people related to the project</td><td rowspan="3">Fixed Part</td><td>Form Text</td><td>DT information of PT people assigned to the project should be provided.</td></tr>
<tr><td>Param</td><td>Metric</td></tr>
<tr><td>DT: is a non-empty set of details types
PT: is a non-empty set of people type</td><td>DT: set(DetailsType)
DetailsType: {name, address, telephone, role, CVs, etc.}
PT: Set (PeopleType)
PeopleType: {overall project manager, implementation responsible, etc.}</td></tr>
<tr><td rowspan="3">Extended Part
CV's information</td><td>Form Text</td><td>The CV's of people who work on the WT should be provided.</td></tr>
<tr><td>Param</td><td>Metric</td></tr>
<tr><td>WT: is a non-empty set of work types</td><td>WT: Set (WorkType)
WorkType: {project, project implementation, etc.}</td></tr>
<tr><td rowspan="4">Extended Part
Previous experience</td><td>Form Text</td><td>PT people related to the project must justify a NAT TT experience on ET.</td></tr>
<tr><td>Param</td><td>Metric</td></tr>
<tr><td>PT: is a non-empty set of people type
NAT: natural
TT: time type
ET: is a non-empty set of experience type</td><td>PT: Set (PeopleType)
PeopleType: {overall project manager, implementation responsible, etc.}
NAT: integer > 0
TT: {years, months, etc.}
ET: Set(ExperienceType)
ExperienceType: {technologies used on the project, Java, C++, etc.}</td></tr>
</table>

8. Product History

Product history			
Goal: Provide information about the history of the product			
Requirement Form *Product history*	**Fixed Part**	**Form Text**	Main developments of the proposed solution information should be provided.
	Extended Part *General abnormalities*	**Form Text**	Solution abnormalities information should be provided.
	Extended Part *Updating frequency*	**Form Text**	Solution frequency updating information should be provided.
	Extended Part *Next evolution*	**Form Text**	The date of providing the next evolution of the solution should be provided.
	Extended Part *Time to correct abnormalities*	**Form Text**	The average time to resolve a critical flaw or defect should be provided.
	Extended Part *Specific Information Abnormalities*	**Form Text**	The following information about abnormalities should be provided: AI.
		Param	**Metric**
		AI: is a non-empty set of abnormalities information	AI: Set(AbnormalityInformation) AbnormalityInformation: {current number of defects, number of defects corrected on the last months, etc.}

9. Community Support

Community Support			
Goal: Provide information about the community tha support the solution if it exists			
Requirement Form *Community Support*	**Fixed Part**	**Form Text**	Information about if the solution is supported by a community should be provided.
	Extended Part *Independence of the community*	**Form Text**	Information about if the community that supports the solution is independet of your company should be provided.
	Extended Part *Community Relationship*	**Form Text**	The information about ColT to a CT dedicated to the proposed solution should be provided.
		Param	**Metric**
		ColT: is a non-empty set of collaboration types	ColT: Set(CollaborationType) CollaborationType: {existence, term of access, etc.}
		CT: is a non-empty exclusive set of community types	CT: ExclusiveSet(CommunityType) CommunityType: {mailing list, forum, etc.}

10. Maintenance Procedure

Maintenance Procedure			
Goal: State the maintenance procedures that the supplier should provide			
Requirement Form *Maintenance Procedures*	**Fixed Part**	**Form Text**	Information about the maintenance procedures should be provided.
	Extended Part *Quality of maintenance*	**Form Text**	Information about the quality of maintenance that the company offers should be provided.
	Extended Part *Maintenance Experience*	**Form Text**	Information about MI on ProjectType projects should be provided.
		Param	**Metric**
		MI: is a non-empty of maintenance information	MT: Set(MaintenanceInformation) MaintenanceInformation: {numbre of maintenance contracts in progress, types of maintenance contracts, etc.}
		ProjectType	ProjectType: {all, similar, etc.}
	Extended Part *Presence of a maintenance team*	**Form Text**	Information about if there is a maintenance team for the proposed solution should be provided.
	Extended Part *Proximity to the maintenance team*	**Form Text**	Information about the proximity of the maintenance team of the proposed solution should be provided.
	Extended Part *Availability of the maintenance team*	**Form Text**	Information about the availability of the maintenance team of the proposed solution should be provided.
	Extended Part *Maintenance during the warranty*	**Form Text**	The proposed solution should be maintained during the warranty period.
	Extended Part *Maintenance after the warranty*	**Form Text**	The proposed solution should be maintained for NAT TT from the expiration of the warranty period.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Working hours during the warranty*	**Form Text**	The provider should be available from NAT1 hour to NAT2 hour to provide substantive assistance to the UserType of the site during the warranty period.
		Param	**Metric**
		NAT1: natural	NAT1: integer > 0
		NAT2: natural	NAT2: integer > 0
		UserType	UserType: {administrator, etc.}

11. Type of Maintenance

Type of maintenance			
Goal: Stablish the type of maintenance offered			
Requirement Form *Type of maintenance*	**Fixed Part**	**Form Text**	The offered maintenance service should include a minimum MT maintenance of the system.
		Param	**Metric**
		MT: is a non-empty set of maintenanace type	MT: Set(MaintenanceType) MaintenanceType: {ongoing, corrective, functional, etc.}
	Extended Part *Corrective maintenance*	**Form Text**	The corrective maintenance should include the following points: MP.
		Param	**Metric**
		MP: is a nonn-empty set of maintenance points	MP: Set(MaintenancePoint) MaintenancePoint: {correction of all non-conformities on the settings, specific developments identified, etc.}
	Extended Part *Ways of maintenance by type on maintenance*	**Form Text**	For MaintenanceType assistance, the provider should provide ST1 first level support and ST2 second level support.
		Param	**Metric**
		Maintenance Type	MaintenanceType: {ongoing, corrective, functional, etc.}
		ST1: SupportType	SupportType: {remote, onsite, etc.}
		ST2: SupportType	SupportType: {remote, onsite, etc.}
	Extended Part *Functional maintenance*	**Form Text**	The functional maintenance should focus on FF related to functional or operational solution.
		Param	**Metric**
		FM: is a non-empty set of functional maintenance type	FM: Set(FunctionalMaintenanceType) FunctionalMaintenanceType: {diagnosing problems, solving problems, etc.)
	Extended Part *Ongoing maintenance*	**Form Text**	The ongoing maintenance should focus on OM.
		Param	**Metric**
		OM: is a non-empty set of ongoing maintenance type	OM: Set(OngoingMaintenanceType) OngoingMaintenanceType: {provision of development versions of the propsoed solution, etc.)

	Extended Part *Not renewing of a maintence type*	**Form Text**	MT maintenance should not be renewed once its duration has finished.
		Param	**Metric**
		MT: is a non-empty set of maintenanace type	MT: Set(MaintenanceType) MaintenanceType: {ongoing, corrective, functional, etc.}
	Extended Part *Duration of the maintenance by type of maintenance*	**Form Text**	The MT maintencance will have a duration of NAT TT of the maintenance period.
		Param	**Metric**
		MT: is a non-empty set of maintenanace type	MT: Set(MaintenanceType) MaintenanceType: {ongoing, corrective, functional, etc.}
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}

12. Supplier History

Supplier History			
Goal: Provide information about the history of the supplier			
Requirement Form *History of the supplier*	**Fixed Part**	**Form Text**	The history of the company should be provided.

13. Project Management Method

Project Management Method			
Goal: Provide information about how the project management method			
Requirement Form *Project Management Method*	**Fixed Part**	**Form Text**	Information about the method for managing the project should be provided.
	Extended Part *Needed Adaption*	**Form Text**	Information about how the proposed method will be adapted if it is necessary should be provided.
	Extended Part *Characteristics of the method*	**Form Text**	The proposed method should be consistent with SMM.
		Param	**Metric**
		SMM: is a non-empty set of standards management method	SMM: Set(StandardManagementMethod) StandardManagementMethod: {industry standards, CMMI, PMBOK, PRINCE2, etc.}

14. Progress Control

Progress Control			
Goal: Provide information about how the progress control should be done			
Requirement Form *Progress Control*	**Fixed Part**	**Form Text**	Information about the visibility of the progress of the project through development of indicators should be provided.
	Extended Part *Concrete Indicators*	**Form Text**	The concrete indicators to follow the progress of the project are PI.
		Param	**Metric**
		PI: is a non-empty set of progress indicators	PI: Set(ProgressIndicator) ProgressIndicator: {number of defects removed, timesheets, rate of progress of each phase, etc.}

15. Steering Committee

Steering committee			
Goal: Stablishment of the steering committee meetings			
Requirement Form *Steering commitee*	**Fixed Part**	**Form Text**	The steering committee should meet during the project to evaluate the good progress of the project.
	Extended Part *Stablishment of the sttering committee*	**Form Text**	The steering committee should be stablished TP.
		Param	**Metric**
		TP: TimePoint	TimePoint: String (e.g. "before starting the project")
	Extended Part *Frequency of the meetings*	**Form Text**	The steering commitee should meet during the project Freq NAT TT.
		Param	**Metric**
		Freq: Frequency	Freq: {at least, every, etc.}
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {years, months, etc.}
	Extended Part *Place of the meeting*	**Form Text**	The meetings should be held on Place.
		Param	**Metric**
		Place	Place: {PCF, PCB, company offices, etc.}
	Extended Part *Members of the steering commitee*	**Form Text**	The steering commitee should be composed by PT.
		Param	**Metric**
		PT: is a non-empty set of people type	PT: Set (PeopleType) PeopleType: {overall project manager, implementation responsible, etc.}
	Extended Part *Progress report*	**Form Text**	At each steering commitee a progress report should be prepared by the supplier.
	Extended Part *Signature of the progress report*	**Form Text**	The validation and signature of progress reports will be done once it fulfill all the requirements.
	Extended Part *Correction of discrepances*	**Form Text**	If it is necessary, the action to correct any discrepances after presenting the progress report should be taken.

16. Documentation Format

Documentation Format			
Goal: State the format rules of the delivered documentation			
Requirement Form *Homogeneous Documentation Format*	**Fixed Part**	**Form Text**	The documentations should be submitted using ST standard formats.
		Param	**Metric**
		ST: is a non-empty set of format types	ST: Set(FormatTypes) FormatType: {paper, electronic, etc.}
	Extended Part *Paper standard format*	**Form Text**	The documentation on paper should be in PF formats.
		Param	**Metric**
		PF: is a non-empty set of paper formats	PF: Set(PaperFormat) PaperFormat: {A0, A1, A2, etc.}
	Extended Part *Electronic standard format*	**Form Text**	The electronic ET documentation should be in EF formats.
		Param	**Metric**
		ET: is a non-empty set of electronic types	ET: Set(ElectronicType) ElectronicType: {image, text, spreadsheet, etc.}
		EF: is a non-empty set of electronic formats	EF: Set(ElectronicFormat) ElectronicFormat: {jpg, xls, doc, etc.}
Requirement Form *Heterogeneous Documentation Format*	**Fixed Part**	**Form Text**	The DT documentations should be submitted using ST standard formats.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DoucmentationType: {user manual, training material, etc.}
		ST: is a non-empty set of format types	ST: Set(FormatTypes) FormatType: {paper, electronic, etc.}
	Extended Part *Paper standard format*	**Form Text**	The documentation on paper should be in PF formats.
		Param	**Metric**
		PF: is a non-empty set of paper formats	PF: Set(PaperFormat) PaperFormat: {A0, A1, A2, etc.}
	Extended Part *Electronic standard format*	**Form Text**	The electronic ET documentation should be in EF formats.
		Param	**Metric**
		ET: is a non-empty set of electronic types	ET: Set(ElectronicType) ElectronicType: {image, text, spreadsheet, etc.}
		EF: is a non-empty set of electronic formats	EF: Set(ElectronicFormat) ElectronicFormat: {jpg, xls, doc, etc.}

17. Convention of the Documents

Convention of the documents			
Goal: State the convention rules of the delivered documentation			
Requirement Form *Convetion of the documents for all the documents*	**Fixed Part**	**Form Text**	All documents should be provided in several languages and following some rules.
	Extended Part *Language of the documents*	**Form Text**	The documents should be written in Lan.
		Param	**Metric**
		Lan: Language	Lan: {English, French, etc.}
	Extended Part *Naming of the documents*	**Form Text**	The documents should be namen using the following convention: DN.
		Param	**Metric**
		DN: is a non-empty order set of documenta-tion naming	DN: OrderSet(DocumentationNaming) DocumentationNaming: {project name, document name, version, etc.}
	Extended Part *Metadata of the documents*	**Form Text**	The documents should include the following metadata: MeT.
		Param	**Metric**
		MeT: is a non-empty set of metadata type	MT: Set (MetadataType) MetadataType: {date of writing, page number, number of total pages, etc.}
Requirement Form *Conventions of the documents for specific documents*	**Fixed Part**	**Form Text**	The DT documentations should be be provided in several languages and following some rules.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DoucmentationType: {user manual, training material, etc.}
	Extended Part *Language of the documents*	**Form Text**	The specified documents should be written in Lan.
		Param	**Metric**
		Lan: Language	Lan: {English, French, etc.}
	Extended Part *Naming of the documents*	**Form Text**	The specified documents should be namen using the following convention: DN.
		Param	**Metric**
		DN: is a non-empty order set of documenta-	DN: OrderSet(DocumentationNaming) DocumentationNaming: {project name, document name, version, etc.}
	Extended Part *Metadata of the documents*	**Form Text**	The specified documents should include the following metadata: MeT.
		Param	**Metric**
		MeT: is a non-empty set of metadata type	MT: Set (MetadataType) MetadataType: {date of writing, page number, number of total pages, etc.}

18. Documentation

Documentation			
Goal: State the documentation that should be delivered			
Requirement Form *Documentation*	**Fixed Part**	**Form Text**	The project documentation should be composed by at least: DT.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
	Extended Part *Aspects to be documented*	**Form Text**	The AT aspects should be documented.
		Param	**Metric**
		AT: is a non-empty set of aspect types	AT: Set(AspectType) AspectType: {solution, project, quality, etc.}
	Extended Part *Content of the documents*	**Form Text**	The DT document should contain Con.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
		Con: Content	Content: Set(String) (eg, "descriptions of all new features", "descriptions of parlamientary wrokflows", "publishing workflows", etc.)
	Extended Part *Notation*	**Form Text**	The DT document should use NT to describe its contents.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
		NT: is a non-empty set of notation type	NT: Set(NotationType) NotationType: {UML, videos, text, etc.}
	Extended Part *Assets*	**Form Text**	The DT document should contain the following assets: AsT.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
		AstT: is a non-empty set of asset types	AT: Set(AssetType) AssetType: {use cases, sequence diagrams, state diagrams, etc.}

	Extended Part *Assets content*	**Form Text**	The assets of DT document refer explicitly to Asp.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
		Asp: Aspect	Aspect: Set(String) (eg, "the functional and nonfunctional requirements", "the elements in the analysis document", etc.)
	Extended Part *Traceability*	**Form Text**	The DT document should establish traceability between Con.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
		Con: Content	Content: Set(String) (eg, "analysis elements", "requirements", "architectural elements", etc.)
	Extended Part *Understandability*	**Form Text**	The information included in the DT document should be easily understandable by UT.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}
		UT: is a non-empty set of user types	UT: Set(UserType) UserType: {administrator, users, etc.}
	Extended Part *Revision*	**Form Text**	The contents of DT documents should be adjusted if it proves insufficient.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports, etc.}

19. Meeting Minutes

Meetings minutes			
Goal: Stablish the need of having minutes about meetings done			
Requirement Form *Minutes of the meetings*	**Fixed Part**	**Form Text**	For each meeting, minutes should be prepared by the supplier.
	Extended Part *Content of the minutes Minutes*	**Form Text**	The minutes should include MI.
		Param	**Metric**
		MI: is a non-empty set of minutes information	Mi: Set(MinutesInformation) MinutesInformation: {list of participants, agenda, taken decisions, unresolved problems, etc.}
	Extended Part *Time to prepare the minutes*	**Form Text**	The minutes should be ready no later than NAT TT after the date of the meeting.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {weeks, months, etc.}
	Extended Part *Minutes client approval*	**Form Text**	The diffusion of the minutes should take place after client approval.

20. Warranty

Warranty			
Goal: Stablish the product warranty			
Requirement Form *Warranty*	**Fixed Part**	**Form Text**	The product should have an stablished period of warranty.
	Extended Part *Warranty Period*	**Form Text**	The warranty period should be NAT TT, starting on TP.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {weeks, months, etc.}
		TP: TimePoint	TimePoint: String (e.g. "the date of notification of final acceptance")
	Extended Part *Extensibility of the warranty*	**Form Text**	The warranty period should be extended taking into account EW.
		Param	**Metric**
		EW: ExtensionWarrantyReasons	ExtensionWarrantyReasons: String. (eg. "the proportion of the recorded delays on the time for the correction of errors")
	Extended Part *Supplier responsability*	**Form Text**	All defects detected during the warranty period should be solved.
	Extended Part *Warranty coverage*	**Form Text**	The warranty covers DT.
		Param	**Metric**
		DT: DefectTypes	DefectTypes: Set(String) (eg, "defects over technical specifications", "defects over functional specifications", "design defects", etc.)

21. Reviews

Reviews			
Goal: State the client right of doing reviews			
Requirement Form *General Reviews*	**Fixed Part**	**Form Text**	Reviews of RA during the project can be conducted.
		Param	**Metric**
		RA: is a non-empty set of reviews assets	RA: Set(ReviewAsset) ReviewAsset: {supplier, production, etc.}
	Extended Part *Review Focus*	**Form Text**	The reviews will be focused on RF
		Param	**Metric**
		RF: is a non-empty set of reviews focusses	RF: Set(ReviewFocus) ReviewFocus: {specific development, treatment of the reproted abnormalies, quality procedures, etc.}
Requirement Form *Reviews focused on a quality standard*	**Fixed Part**	**Form Text**	Reviews of RA during the project on the basis of quality standards can be conducted.
		Param	**Metric**
		RA: is a non-empty set of reviews assets	RA: Set(ReviewAsset) ReviewAsset: {supplier, production, etc.}
	Extended Part *Review Focus*	**Form Text**	The reviews will be focused on RF
		Param	**Metric**
		RF: is a non-empty set of reviews focusses	RF: Set(ReviewFocus) ReviewFocus: {specific development, treatment of the reproted abnormalies, quality procedures, etc.}
	Extended Part *Quality criteria agreement*	**Form Text**	The level of quality expected for the various artifacts will be agreed with the supplier.
	Extended Part *Divulgation of quality criteria*	**Form Text**	Quality criteria to be applied will be divulgated TP.
		Param	**Metric**
		TP: TimePoint	TimePoint: String (e.g. "before starting the project", "in the initial phase of the project")
	Extended Part *Quality criteria based on standars*	**Form Text**	The quality criteria used during the review will be based on the QS
		Param	**Metric**
		QS: is a non-empty set of quality standards	QS: Set(QualityStandard) QualityStandard: {IEEE380, IEEE829, IEEE1016, ISO/IEC9126, etc.}
	Extended Part *Periodically document revision*	**Form Text**	The DT should be reviewed TiP to ensure QA.
		Param	**Metric**
		DT: is a non-empty set of documentation types	DT: Set(DocumentationTypes) DocumentationType: {project quality plan, meeting minutes, user manual, training materials, progress reports,
		TiP: TimePeriod	TimePeriod: String (eg. regularly, every 2 weeks, etc.)
		QA: Quality-Aspects	QualityAspects: String. (eg. "they comply with the rules of art")

22. Meetings Organization

Meetings Organization			
Goal: Stablish the way in which meetings should be organized			
Requirement Form *Organization of meetings*	**Fixed Part**	**Form Text**	A notice should be sent to all persons who must attend the meetings.
	Extended Part *Agreement*	**Form Text**	Before sending the notice, the supplier should agree the meeting
	Extended Part *Content of the notice*	**Form Text**	The notice should inform about NI.
		Param	**Metric**
		NI: is a non-empty set of notice information	NT: Set(NoticeInformation) NoticeInformation: {maeeting date, meeting time, meeting place, guest list, agenda, etc.}

23. Source Code Documented

Source code documented			
Goal: State the necessity of documenting the source code			
Requirement Form *Source code documented*	**Fixed Part**	**Form Text**	The source code elements should be documented.
	Extended Part *Different levels of abstraction*	**Form Text**	The documentation of the source code can use several levels of abstraction if the complexity of the
	Extended Part *Extra information*	**Form Text**	The documentation of the source code should include PA properly
		Param	**Metric**
		PA: is a non-empty set of programming aspects	PA: Set(ProgrammingAspect) ProgrammingAspect: {configuration files, installation procedures, etc.}
	Extended Part *Uniformity*	**Form Text**	The style and level of documentation in the source code should uniform thorough the code.
	Extended Part *Coding rules*	**Form Text**	The source code should include the coding rules that define the essential style conventions for obtaining a uniform coding style.

24. Settlement of Disputes

Settlement of Disputes			
Goal: State how the disputes will be solved			
Requirement Form *Settlement of Disputes*	**Fixed Part**	**Form Text**	Any dispute that may arise during the project and which could not be settled amicably should be competence of
		Param	**Metric**
		CO: Competence Organization	CompetenceOrganization: String. (eg. "the courts of Luxembourg city")
	Extended Part *Null contratual clauses*	**Form Text**	Any contractual clause that is unilateral for the supplier or that is outside the requirement books will

25. Planning

Planning			
Goal: Provide information about the palnning schedule			
Requirement Form *General Planning*	**Fixed Part**	**Form Text**	Detailed information about the schedule of the development of the project should be provided.
	Extended Part *Planning focus*	**Form Text**	The schedule should be focused on PT phases of the project.
		Param	**Metric**
		PT: is a non-empty set of phases types	PT: Set(PhaseType) PhaseType: {specification, implementation, etc.}
	Extended Part *Planning contents*	**Form Text**	The schedule should identify precisely Con.
		Param	**Metric**
		Con: Content	Content: Set(String) (eg, "different artifacts to deliver", "delivery times of the artifacts", "milestones corresponding to the supply of modules, sub-modules and sub-
	Extended Part *Planning coverage*	**Form Text**	The schedule should describe all steps necessary to complete the project from STP to FTP.
		Param	**Metric**
		STP: StartTime Point	StartTimePoint: {project's inception, specification stage, etc.}
		FTP: FinalTime Point	FinalTimePoint: {project's final acceptance, text stage, etc.}
	Extended Part *Planning dates close*	**Form Text**	The schedule should include dates relatively close (every NAT TT).
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {weeks, months, etc.}
	Extended Part *Planning Functionalities*	**Form Text**	The schedule should be the baseline for SO.
		Param	**Metric**
		SO: Schedule Objective	ScheduleObjective: Set(String). (eg. "the validation of various artifacts", "assessing project progress at sterring
	Extended Part *Planning considerations*	**Form Text**	The planning should consider
		Param	**Metric**
		CA: Consideration Aspects	ConsiderationAspects: Set(String). (eg. "any data migration must be done", "installing and configuring test
	Extended Part *Planning reescheduling*	**Form Text**	The schedule will be adjusted in case of ET events.
		Param	**Metric**
		ET: EventType	EventType: {unanticipated, staff problems, etc.}

Requirement Form *Fixed planning*	**Fixed Part**	**Form Text**	The schedule of the development of the project should follow specific
	Extended Part *Planning focus*	**Form Text**	The schedule should be focused on PT phases of the project.
		Param	**Metric**
		PT: is a non-empty set of phases types	PT: Set(PhaseType) PhaseType: {specification, implementation, etc.}
	Extended Part *Fixed starting milestone*	**Form Text**	The PT stage should start on DA.
		Param	**Metric**
		PT: is a non-empty set of phases types	PT: Set(PhaseType) PhaseType: {specification, implementation, etc.}
		DA: date	DA: date
	Extended Part *Fixed ending milestone*	**Form Text**	The DL should be ready at DA.
		Param	**Metric**
		DL: Deliverable	Deliverable: {functional solution, implementation stage deliverables,
		DA: date	DA: date
	Extended Part *Planning duration*	**Form Text**	The DL should be ready within a maximum of NAT TT after TP.
		Param	**Metric**
		DL: Deliverable	Deliverable: {functional solution, final solution, etc.}
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {weeks, months, etc.}
		TP: TimePoint	TimePoint: String (e.g. "the date of notification of final acceptance")

26. Release

Release			
Goal: Stablish the time when the solution should be released			
Requirement Form *Release*	**Fixed Part**	**Form Text**	The supplier should release the solution TP.
		Param	**Metric**
		TP: TimePoint	TimePoint: String (e.g. "after provisional acceptance")
	Extended Part *Release Notification*	**Form Text**	The supplier should notify the release NAT TT before.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {weeks, months, etc.}

27. Job Properties and Intellectual Rights

Job Properties and Intellectual Rights			
Goal: Stablish the rights of using assets involved in the project			
Requirement Form *Job properties and intellectual rights*	**Fixed Part**	**Form Text**	AT will become the PT property.
		Param	**Metric**
		AT: AssetType	AssetType: Set(String). (eg. "hardware", "software", "data and documents provided or paid by client as deliverables")
		PT: PropertyType	PropertyType: {supplier's, client's, etc.}
	Extended Part *Use of assets*	**Form Text**	PT can use freely and without restriction AT.
		Param	**Metric**
		PT: PepopleType	PeopleType: {client, supplier, etc.}
		AT: AssetType	AssetType: Set(String). (eg. "hardware", "software", "data and documents provided or paid by client as deliverables")
	Extended Part *Assets Return*	**Form Text**	The supplier should return AT that client provide related to the project.
		Param	**Metric**
		AT: AssetType	AssetType: Set(String). (eg. "hardware", "software data", "documents")
	Extended Part *Right of copies*	**Form Text**	The supplier will be allowed to keep copies of documents for which client has given consent.

28. Data Migration

Data migration			
Goal: State the necessity of migrate data			
Requirement Form *General Data migration*	**Fixed Part**	**Form Text**	The need of some data migration identified during the analysis stage should be one of the deployment steps proposed by the supplier.
Requirement Form *General Data migration*	**Fixed Part**	**Form Text**	Data migration should be considered one of the deployment steps proposed by the supplier.
	Extended Part *Data to migrate*	**Form Text**	The data to migrate are DT.
		Param	**Metric**
		DT: DataType	DataType: SetString (eg. "all databases", "underway projects", etc.}
	Extended Part *Confidetiality of migrated data*	**Form Text**	The provided should ensure the confidentiality of the migrated data into the solution.
	Extended Part *Data migration source*	**Form Text**	The DT should be recovered from FT.
		Param	**Metric**
		DT: DataType	DataType: Set(String) (eg. "the data and data structure of companies"}
		FT: FileType	FileType: {excel file, text file, database, etc.}

29. Privacy

Privacy			
Goal: State the privacy rules among client and supplier			
Requirement Form *Privacy*	**Fixed Part**	**Form Text**	The clientand the supplier should accept to preserve the confidentiality of any information collected as part of the project and not disclose anything to any third party without the written permission of the other party.
	Extended Part *Parties Privacy Extension*	**Form Text**	The confidentiality should extent to PaT.
		Param	**Metric**
		PaT: PartiesType	PartiesType: Set(String). (eg. "employees of each party", "parties subcontractors")
	Extended Part *Affairs Privacy Extension*	**Form Text**	The supplier should not disclose at any time information concerning client's affairs or any of client's representatives that can be known during the project.
	Extended Part *Exemptions of the privacy in legal affairs*	**Form Text**	In case it would be needed, the supplier could disclose information know during the project in legal proceedings.
	Extended Part *No Benefit over the Private Information*	**Form Text**	At any time (whether during or after project), the supplier should not use for his own interest or for the benefit of any person, firm, corporation, association or other business, our the trade secrets, programs of development of business plans owned by the client or relating to client's business including information related to client's employees.

30. Installation

Installation			
Goal: Stablish the time when the solution should be installed			
Requirement Form *Installation*	**Fixed Part**	**Form Text**	The supplier should install the solution TP.
		Param	**Metric**
		TP: TimePoint	TimePoint: String (e.g. "after final acceptance")

31. Payment Method

Payment Method			
Goal: State the payment schedule			
Requirement Form *Agreed Payment*	**Fixed Part**	**Form Text**	The payment should follow an schedule that will be agreed by th supplier and the client.
	Extended Part *No automatic payments*	**Form Text**	No payment will be automatically established.
Requirement Form *Stablished Payment*	**Fixed Part**	**Form Text**	The payment should follow an schedule provided by the client.
	Extended Part *Payment by Stages*	**Form Text**	PER% of the payment will be done ST.
		Param	**Metric**
		PER: Percentage	PER: Float > 0 and <= 100
		ST: Stage	Stage: String (eg, "after all deliveries", "during development")
	Extended Part *Dependent Payments*	**Form Text**	The payments done ST will depend on DP.
		Param	**Metric**
		ST: Stage	Stage: String (eg, "after all deliveries", "during development")
		DP: Dependency	Dependency: String (eg, "the progress of the project", "the approval of the minutes")
	Extended Part *No automatic payments*	**Form Text**	No payment will be automatically established.
	Extended Part *Milestone payment schedule*	**Form Text**	The supplier can propose a payment schedule based on project milestones.

32. Final Acceptance

Final acceptance			
Goal: Stablish the time and conditions when the solution should be finally accepted			
Requirement Form *Final acceptance*	**Fixed Part**	**Form Text**	The conditions for the final acceptance of the solution will be AC.
		Param	**Metric**
		AC: Acceptance Conditions	AcceptanceConditions: Set(String) (eg. "having the solution working effectively in our site", "training users")
	Extended Part *Final acceptance period*	**Form Text**	The final acceptance will be done after NAT TT of TP.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TT: time type	TT: {weeks, months, etc.}
		TP: TimePoint	TimePoint: String (e.g. "the date of notification of proval acceptance")

33. Analysis

Analysis			
Goal: State the actions to take during analysis stage			
Requirement Form *Analysis*	**Fixed Part**	**Form Text**	Analysis stage should be one of the deployment steps proposed by the supplier.
	Extended Part *Requirement analysis*	**Form Text**	The supplier should do the requirements analysis needed to achieve the knowledge necessary about KT.
		Param	**Metric**
		KT: is a non-empty set of knowledge type	KT: Set(KnowledgeType) KnowledgeType: {technical design features, technical solution, functional adaptation, etc.}
	Extended Part *Analysis Extra Objectives*	**Form Text**	The suppleir also should In the course of this analysis AC.
		Param	**Metric**
		AC: Analysis Action	Analysis Action: Set(String) (eg. "analyze existing client's applications that the supplier will use during development", "indentify key deficiencies in the current implementation")
	Extended Part *Analysis Validation*	**Form Text**	The analysis stage will be validated by AD.
		Param	**Metric**
		AD: AssetDocument	AssetDocument: {a specification document, an analysis document, etc.}

34. Start of the Solution

Start of the Solution			
Goal: Stablish the conditions to start the solution working in the client			
Requirement Form *Start of the Solution*	**Fixed Part**	**Form Text**	The start of the solution will be done by SA.
		Param	**Metric**
		SA: StartActions	StartActions: Set(String) (eg. "migrating client current data without modification")

35. Development

Development			
Goal: State the actions to take during development stage			
Requirement Form *Development with parametrization*	**Fixed Part**	**Form Text**	Development, and its included parametrization, should be considered one of the deployment steps proposed by the supplier.
	Extended Part *Parametrization Goal*	**Form Text**	The parametrization sholud be done to PA.
		Param	**Metric**
		PA: Parameteri-zation Goal	ParameterizationGoal: String (eg. "meet the requirements stated in the specifications")
	Extended Part *Parametrization Focus*	**Form Text**	The parametrization should be focus on AT adaptation.
		Param	**Metric**
		AT: Adaption Type	AdaptationType: {functional, etc.}
	Extended Part *Development and parametrization documentation*	**Form Text**	The development and parametrization of the solution should be recorded in documents by the supplier .
	Extended Part *Documentation deliverement*	**Form Text**	The development and parametrization documentation should be delivered to the client.
	Extended Part *Minimum development*	**Form Text**	The coverage of requirements with minimum development will be an advantage.
	Extended Part *Development method*	**Form Text**	The developmen of SP will follow the DM method.
		Param	**Metric**
		SP: SystemPart	SystemPart: String (eg. "the solution", "each module of the solution")
		DM: Developme-ntMethodolgy	DevelopmentMethodology: {peer programming, etc.}
	Extended Part *Development Errors*	**Form Text**	The supplier should identify any error and the ways to adress them during the development.

Requirement Form *Development without parametrization*	**Fixed Part**	**Form Text**	Development should be considered one of the deployment steps proposed by the supplier.
	Extended Part *Development Basis*	**Form Text**	The development sholud be made on the basis of AsD.
		Param	**Metric**
		AsD: is a non-empty set of AssetDocument	AsD: Set(AssetDocument) AssetDocument: {a specification document, an analysis document,
	Extended Part *Development with community*	**Form Text**	The devolopment should be made with communities of free software developers for DO.
		Param	**Metric**
		DO: Develop-mentObjective	DevelopmentObjective: String. (eg. "integration of generic capabilities with existing free software components")
	Extended Part *Development method*	**Form Text**	The developmen of SP will follow the DM method.
		Param	**Metric**
		SP: SystemPart	SystemPart: String (eg. "the solution", "each module of the solution")
		DM: Developme-ntMethodolgy	DevelopmentMethodology: {peer programming, etc.}
	Extended Part *Development Errors*	**Form Text**	The supplier should identify any error and the ways to adress them during the development.

36. Acceptance Tests

Acceptance tests			
Goal: Stablish the type of tests to accept the solution			
Requirement Form *Acceptance tests*	**Fixed Part**	**Form Text**	The acceptance of the solution should involve the following tests: TT.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
	Extended Part *Tests based on recipes*	**Form Text**	The TT acceptance tests should be based on recipes.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
	Extended Part *Tests date*	**Form Text**	The TT aceptance test should be TP.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
		TP: TimePoint	TimePoint: String (e.g. "the date of notification of final acceptance")
	Extended Part *Testing time*	**Form Text**	The supplier should have NAT TiT to perform the TT test.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		TiT: time type	TiT: {weeks, months, etc.}
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
	Extended Part *Content of the recipes*	**Form Text**	The recipes for TT acceptance test should be based on <param2>.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
		TB: TestBasis	TestBasis: Set(String) (eg. "test cases", "at least four typical scenarios of client work")
	Extended Part *Simulated testing conditions*	**Form Text**	The TT acceptance tests should be on a temporary test platform reproducing similar conditions to the production environment.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}

	Extended Part *Writter of the recipes*	**Form Text**	The content of the recipes for TT acceptance tests should be written by PaT.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
		PaT: PartiesType	PartiesType: Set(String). (eg. "the suppleir", "the client")
	Extended Part *Report for validation of recipes*	**Form Text**	The validation of recipes for TT acceptance tests should create an official report.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}
	Extended Part *Validation Privacy*	**Form Text**	The client will issue a OT opinion on each item of the recipe depending on the testing results.
		Param	**Metric**
		OT: is a non-empty exclusive set of opinion types	OT: ExclusiveSet (OpinionType) OpinionType: {favourable, reserved, etc.}
	Extended Part *Positive validation foundations*	**Form Text**	A positive validation of the tests will be issued if AC.
		Param	**Metric**
		AC: Acceptance Conditions	AcceptanceCondition: String. (eg. "if the solution does not have a large amount of non-blocking errors")
	Extended Part *Correction of errors for positive validation*	**Form Text**	The errors detected during the tests should be corrected by the supplier in order to pass the tests.
	Extended Part *New version after correction*	**Form Text**	The supplier should provide a new version of the solution for TT acceptance once all detected correctins have been done.
		Param	**Metric**
		TT: is a non-empty set of test types	TT: Set(TestType) TestType: {final, provisional, etc.}

37. Training

Training			
Goal: Stablish the solution training			
Requirement Form *General Training*	**Fixed Part**	**Form Text**	The supplier should provide trainning about the solution.
	Extended Part *Trainning location*	**Form Text**	The trainning will take place at Place.
		Param	**Metric**
		Place	Place: {PCF, PCB, company offices, client office, etc.}
	Extended Part *People to train*	**Form Text**	The supplier should train NAT UT users of the solution.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		UT: is a non-empty set of user types	UT: Set(UserType) UserType: {administrator, main, normal, etc.}
	Extended Part *Trainning materials*	**Form Text**	The trainning materials should be provided by PaT.
		Param	**Metric**
		PaT: PartiesType	PartiesType: Set(String). (eg. "the suppleir", "the client")
Requirement Form *Training for specific users*	**Fixed Part**	**Form Text**	The supplier should provide training about the solution for each group of users.
	Extended Part *Different groups to train*	**Form Text**	The different groups of users to train will be differentiated by CT.
		Param	**Metric**
		CT: Criteria	Criteria: Set(String) (eg. "the different patterns of use of the solution", "the differences in the configuration of the solution")
	Extended Part *Trainning location*	**Form Text**	The trainning will take place at Place.
		Param	**Metric**
		Place	Place: {PCF, PCB, company offices, client office, etc.}
	Extended Part *People to train*	**Form Text**	The supplier should train NAT UT users of the solution.
		Param	**Metric**
		NAT: natural	NAT: integer > 0
		UT: is a non-empty set of user types	UT: Set(UserType) UserType: {administrator, main, normal, etc.}
	Extended Part *Trainning materials*	**Form Text**	The trainning materials should be provided by PaT.
		Param	**Metric**
		PaT: PartiesType	PartiesType: Set(String). (eg. "the suppleir", "the client")

38. Packaging the solution

Packaging of the solution			
Goal: Stablish the package in whoch the price should be divided			
Requirement Form *Packaging of the solution*	**Fixed Part**	**Form Text**	The offer (price) made by the supplier should be composed of the PT packages.
		Param	**Metric**
		PT: is a non-empty set of packgage types	PT: Set(PackageType) PackageType: {licensing, installation, data migration, training, maintenance, etc.}
	Extended Part *Offer inclusion*	**Form Text**	The offer (price) should include the costs of OI.
		Param	**Metric**
		OI: Offer Inclusion	OfferInclusion: SetString (eg. "travelling", "training due to client specific training method")
	Extended Part *Offer exclusion*	**Form Text**	The offer (price) should not include the costs of OE.
		Param	**Metric**
		OI: Offer Exclusion	OfferExclusion: SetString (eg. "delaying in the schedule")
	Extended Part *Contents of the packages*	**Form Text**	The PT package should include PC.
		Param	**Metric**
		PT: is a non-empty set of packgage types	PT: Set(PackageType) PackageType: {licensing, installation, data migration, training, maintenance, etc.}
		PC: Package Content	PC: Set(String) (eg. "license price of the solution and the server database", "price licensing scheudles if necessary")

Appendix B: Study of the Utility of a Patterns Catalogue during the requirement engineering Stage of Software Projects

1. About the interviewee

****** To be answered before the interview (if possible) ******

With the following questions we want to know your personal and experience aspects in order to better understand your answers.

We will not use this information to any other finality than data analysis of the current study; therefore it will not be published by any means. Mandatory fields are indicated with (*)

Personal Information

First and last names:	
Contact e-mail:	

Studies

Main academic degree (*):	
Related studies:	
Related studies:	
Related studies:	

Professional experience in the organization

Position (*):	
Years in this position (*):	
Years in the organization (*):	

About the organization

Name:	
Number of employees (*):	
Main business line (*):	

2. Questions about the context

****** Questions to be answered during the interview ******

The following questions are about the Requirements Engineering phase as conducted in your organization and your experience in this stage. With these questions we can better understand the answers you provide in the rest of the interview.

Organization context

In this first part of the interview we want to obtain a profile of the projects that your organization develops, as well as the type of requirements

engineering carried out in these projects and what are the biggest difficulties found in this stage. We are also interested in knowing what your role in the requirements engineering process during these projects is.

	(Organization Context)
Q1	What are the most frequent domains of software projects in which your organization works?
Q2	Describe briefly the requirements engineering process that is carried out in your organization.
Q2b	In particular, does it exist a documented methodology related with this process?

Q3	What are the most common problems related with requirements engineering that your organization faces up?
	• Internal problems related with the management of requirements
	  ○ Ambiguities in the writing of requirements.
	  ○ Inconsistencies between different requirements.
	  ○ Lack of traceability.
	• External problems impacting customers and end users
	  ○ Misunderstandings with the clients in the definition of terms.
	  ○ Lack of requirements (incompleteness) in the final result.
	  ○ Divergences between several people from the client organization.
	  ○ Changes in the opinion of the client through the project.
	• Others (specify which ones).

	(Role during the Requirements Engineering)
Q4	How many software projects have you worked in during the process of requirements elicitation? In these projects, were you in charge of all type of requirements (functional, non-functional, non-technical)? If not, specify the concrete types.
Q4b	How many requirement books do you write per year? a) None b) Between 1 and 3 c) More than 3

Non-Functional Requirements (NFRs)

In this second part of the interview you will be asked about your conception of Non-Functional Requirements (NFRs) and what is your experience in their

management. We also want to know how the NFRs are treated in your projects (what are the most prioritary ones, the most usual ones, if some of them are recurrent or the problems that this type of requirement entails). Below you can find a possible classification schema for NFRs that can be helpful in further questions (extracted from the ISO/IEC 9126-1 quality standard):

1. Functionality	
	1.1 Suitability
	1.2 Accuracy
	1.3 Interoperability
	1.4 Security
	1.5 Functionality Compliance
2. Usability	
	2.1 Understandability
	2.2 Learnability
	2.3 Operability
	2.4 Attractiveness
	2.5 Usability Compliance
3. Reliability	
	3.1 Fault tolerance
	3.2 Recoverability
	3.3 Maturity
	3.4 Reliability Compliance

4. Maintainability	
	4.1 Analisability
	4.2 Changeability
	4.3 Stability
	4.5 Testability
	4.6 Maintainability Compliance
5. Portability	
	5.1 Adaptability
	5.2 Installability
	5.3 Coexistence
	5.4 Replaceability
	5.5 Portability Compliance
6. Efficiency	
	6.1 Time Behaviour
	6.2 Resource Utilisation
	6.3 Efficiency Compliance

	(Knowledge about NFRs in the projects)
Q5	Provide 3 examples of NFRs that show what your conception of this type of requirement is.
Q6	In a scale from 1 to 5 (1=very low; 2=low; 3=medium; 4=high; 5=very high), how do you rate your experience in NFR management?
Q6b	In particular, how do you rate it during the elicitation stage?
Q7	Do you use a NFR classification schema? If so please mention if it is a standard, a company classification schema, etc.
Q7b	What type of NFRs do you usually deal with in your projects? You may use the previous classification if necessary.
Q8	What are the types of NFRs that use to be more prioritary in your projects? You may use the previous classification if necessary.
Q9	What percentage of NFRs do you estimate that are recurrent in your projects? Answer following this scale: almost none (~0%); few (~25%); about 50%; many (~75%); almost all (~100%).
Q10	What are the most recurrent NFRs or types of NFRs in your projects? You may use the previous classification if necessary.

Q11	What are the NFRs that raise more problems during the requirements engineering process? You may use the previous classification if necessary.
Q11b	What type of problems (ambiguity, etc.)?

Non-Technical Requirements (NTRs)

Non-Technical Requirements (NTRs) are those ones that set restrictions such as: 1) Characteristics that the software producer organization has to comply (for instance, its organizational structure, its reputation or the support offered); 2) Business characteristics that the software product has to fulfil (licensing schemas, property rights, guarantee and costs); 3) Other characteristics of the software product (related, for instance, with deliverables or customization of the product). Below you can find a possible classification schema for NTRs that can be helpful in further questions:

1. Supplier		
	1.1 Organizational Structure	Description of the organizational structure of the supplier company.
	1.2 Positioning and Strength	Description of the position and orientation of the supplier company in the market.
	1.3 Reputation	Recognition of the capability of the supplier to perform similar projects based on past experiences and certifications.
	1.4 Services Offered	Description of the services offered by the supplier.
	1.5 Support	Description of the support mechanisms offered by the supplier company.
2. Business		
	2.1 Licensing Schema	Description of the licensing options.
	2.2 Ownership	Description of the aspects in relation to the intellectual property rights.
	2.3 Guarantees	Detail of the guarantees provided over the product.
	2.4 Licensing Cost	Description of the total costs of ownership for the different licensing options available.
	2.5 Platform Cost	Estimation of the costs for the required production platform.
	2.6 Implementation Cost	Estimation of implementation costs based on similar past experiences.
	2.7 Network Cost	Estimation of additional costs for network operation.
3. Product		
	3.1 History	Evolution of the product since it has been offered to the clients.
	3.2 Deliverables	Detail of the out-of-the-box and expected post-implementation deliverables.
	3.3 Parameterization and Customization	Description of the initial effort required for the product to operate.

In this third part of the interview you will be asked about how your organization treats this type of requirements (as a part of non-functional requirements or as a separate type) and, in case of treating them in a separate way, what is your experience managing them and how they appear in your projects.

	(Knowledge about NTRs in the projects)
Q12	Does it exist in your organization the concept of NTR or this class of requirements is considered as a part of NFRs? In case of considering NTRs as part of NFRs move to the next section *Questions about patterns*.
Q13	Do you use a different name for NTRs? Which one?
Q14	In a scale from 1 to 5 (1=very low; 2=low; 3=medium; 4=high; 5=very high), how would you rate your experience in NTRs management?
Q14b	In particular, how do you rate it during the elicitation stage of this type of requirements?
Q15	Do you use a NTR classification schema? If so, please mention if it is a standard, a company classification schema, etc.
Q15b	What type of NTRs do you usually deal with in your projects? You may use the previous classification if necessary.
Q16	What are the types of NTRs that use to be more prioritary in your projects? You may use the previous classification if necessary.
Q17	What percentage of NTRs do you estimate that are recurrent in your projects? Answer following this scale: almost none (~0%); few (~25%); about 50%; many (~75%); almost all (~100%).
Q18	What are the most recurrent NTR or types of NTRs in your projects? You may use the previous classification if necessary.
Q19	What are the NTRs that raise more problems during the requirements engineering process? You may use the previous classification if necessary.
Q19b	What type of problems (ambiguity, etc.)?

3. Questions about patterns

****** Questions to be answered during the interview ******

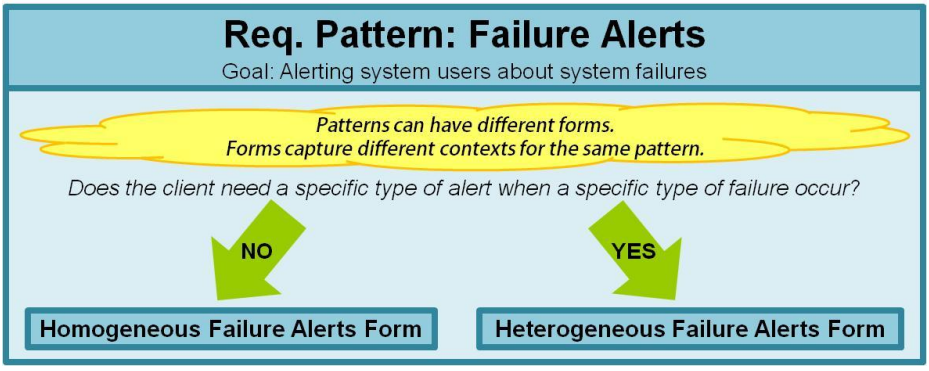
The following questions deal with the specific aspects of our investigation. We aim at finding out what value a pattern catalogue may provide to the requirements engineering process and what role the catalogue would play during the development of software projects.

Introduction to the concept of Requirement Pattern

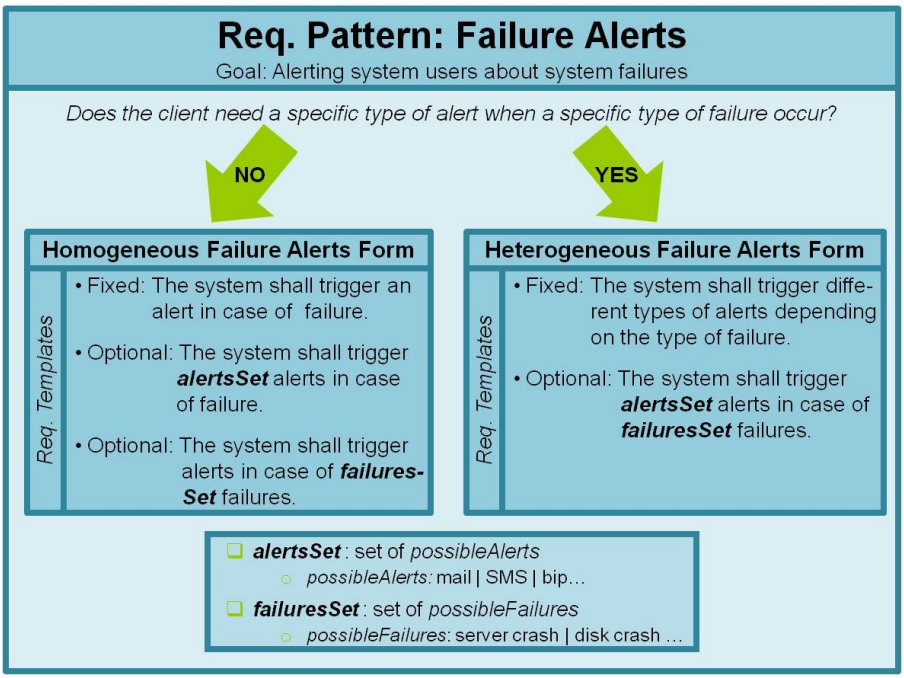
To make it simple, we consider in this study that a requirement pattern contains the text that describes some interrelated and recurrent requirements in a generic way. For instance, the pattern Failure Alerts contains the text of the requirements that are related to the fact of wanting

to have a system that warns of failures that occur (which types of failures should be monitored, what types of alerts must be launched, etc.).

A pattern consists of several Forms, which represent different contexts for the same pattern. For each project, the most suitable Form has to be chosen. Forms are mutually exclusive: in a project you can only use one of them. For example, the pattern Failure Alerts has two different Forms, as illustrated in the figure:



In each Form we always find a Fixed Template and some Optional Templates. The Fixed Template always becomes a requirement when using the Form. Optional Templates, as its name suggests, are only used if you want more precise information about the requirement contained in the Fixed Template. Each Template is formed by the text associated with this requirement and optionally some parameters (in italics in the example), which are those parts that vary in using the template in different software projects.



Structure of a Requirement Pattern

The questions in this fourth part of the interview try to figure out whether the presented structure of a requirement pattern would fit the needs of your organization. Answer the following questions based on one type of NFRs that is recurrent in your projects.

	(Structure of a Requirement Pattern)
	Consider again your answer to Q10 and select only one type of NFR that is recurrent in your projects.
Q20	Write up to 5 requirements of this type of NFR that have appeared in a requirement book of some of your projects.
Q21	Considering how this type of NFR appears in your projects documentation, does it appear as a single requirement or as several requirements?
Q22	From the requirements related to the type of NFR chosen, do you think they could be grouped according to the context in which they apply? For the examples given in Q25, what are these contexts?
Q23	From the contexts given as examples in the previous question, do you think they are mutually exclusive or on the contrary you could need several of these contexts in the same project?
Q24	From the requirements that may appear within a same context for the type of NFR chosen, is there some that appears on most (or all) of the projects where you apply this context? Give an example of what this requirement would be in a particular context.
Q25	From the requirements related to the type of NFR chosen, are there any of them that represent the same restriction but have some small part that varies due to the project? Give 3 examples of the same requirement applied to different projects where this happens.

Usage of a pattern catalogue

In this fifth part of the interview we want to discover what would be the value of a requirement pattern catalogue in the software projects of your organization, as well as knowing what would be needed to incorporate it and what would be the pros and cons.

	(Use of a Requirement Pattern Catalogue)
Q26	Do you think it would be useful to use a catalogue of patterns for NFRs during the elicitation process? Please use the following scale (from 1=useless; to 5=useful)
Q26b	And for their documentation?

Q26c	And for their validation?
Q26d	What might be the advantages obtained in each of these stages?
Q27	In case that your organization considers NTRs as an independent type from NFRs, do you think it would be useful to use a catalogue of patterns for NTRs during the elicitation process?
Q27b	And for their documentation?
Q27c	And for their validation?
Q27d	What might be the advantages obtained in each of these stages?
Q28	Do you think it would be useful to use a catalogue of patterns for Functional Requirements (FRs) during the elicitation process in one of the most common domains in your organization?
Q28b	And for their documentation?
Q28c	And for their validation?
Q28d	What might be the advantages obtained in each of these stages?
Q28e	Do you foresee any other benefit?
	(Incorporation of a Requirement Pattern Catalogue)
Q29	If you have considered that a pattern catalogue may be useful during the requirements engineering stage for some kind of requirement, what would be needed in order to add the catalogue to the usual requirements engineering activities as currently undertaken by your company? You can choose more than one. • A training • A software tool • A guide / procedure • Other (please explain)
Q30	What would be the major constraint limiting the adoption / incorporation of such catalogue to your RE activities?

Appendix C: Glossary

1. **Conceptual model.** A conceptual model, also known as domain model, represents 'concepts' (entities) and relationships between them.
2. **Functional requirement.** Functional requirements establish the observable behaviour that must exhibit the system (calculations, manipulations, listings, evolution aspects, etc.), as well as the data types specification.
3. **i* framework.** The i* framework proposes an agent-oriented approach to requirements engineering centering on the intentional characteristics of the agent. Agents attribute intentional properties (such as goals, beliefs, abilities, commitments) to each other and reason about strategic relationships. Dependencies between agents give rise to opportunities as well as vulnerabilities. Networks of dependencies are analyzed using a qualitative reasoning approach. Agents consider alternative configurations of dependencies to assess their strategic positioning in a social context [62].
4. **Metamodel.** Description or definition of a well-defined language in the form of a model.
5. **Model.** A description of (part of) a system written in a well-defined language.
6. **Noise.** In the bibliographic context, noise is defined as the set of results which are not related with the area or topic the user is interested in obtaining.
7. **Non-functional requirement.** Non-functional requirements establish the criteria or global qualities of the software system and set restrictions (internal and external) on the software and the development process. Common types of non-functional requirements are: usability, efficiency and portability.
8. **Non-technical requirement.** Non-technical requirements are those that do not refer directly to the intrinsic quality of software, but to the context of the system under analysis. They include economic, political and managerial issues [45].
9. **Pattern.** Consistent and recurring characteristic or trait that helps in the identification of a phenomenon or problem, and serves as an indicator or model for predicting its future behaviour or solution. They clearly state the original phenomenon or problem (to know when to use it) and the future behaviour or solution.
10. **Requirements Book.** See *Software Requirement Specification (SRS)*.
11. **Requirements Elicitation.** Requirements Elicitation is the process of discovering the requirements for a system by communication with customers, system users and others who have a stake in the system development [63].
12. **Requirements Engineering.** Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families.

13. **Software Requirement Specification (SRS).** A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for a software system. An SRS fully describes what the software will do and how it will be expected to perform.
14. **Template.** Design, mold, or pattern of an item (or a group of items) that serves as a basis or guide for designing or constructing similar items. Templates may allow certain degree of freedom in aiming alterations or modifications. They only state a guide, but anything about when to use them.
15. **Unified Modeling Language (UML).** UML is a language to specify, visualize, and document models of software systems, including their structure and design. UML can be used for business modeling and modeling of other non-software systems.
16. **Use case.** A Use Case is the smallest unit of activity that is meaningful to the user. It must be self-contained, and leave the business of the application in a consistent state.