# Definition and Use of Software Requirement Patterns in Requirements Engineering Activities

Cristina Palomares

GESSI Research Group, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
cpalomares@essi.upc.edu

**Abstract.** The final quality of software products and services depends on the requirements stated in the Software Requirements Specification (SRS). However, some problems like ambiguity, incompleteness and inconsistency, have been reported in the writing of SRS, especially when natural language is used. Requirements reuse has been proposed as a key asset for requirement engineers to efficiently elicit, validate and document software requirements, and as a consequence obtain SRS of better quality through more effective engineering processes. Among all the possible techniques to achieve reuse, patterns hold a prominent position. Although there have been several techniques proposed to reuse requirements, it may be observed that no concrete proposal has achieved a wide acceptance. Due to that, this research proposes the PABRE framework, which uses Software Requirement Patterns (SRP) as a means to capture and reuse requirements knowledge in the context of IT projects.

**Keywords:** Requirement engineering; Software requirements; Requirement reuse; Requirement Patterns.

## 1 Problem and Relevance

Requirements Engineering (RE) is the process in which the system to be built is defined, getting as result the requirements that will act as a guideline for the software development team. Requirements elicitation is the process of acquiring these requirements from system stakeholders. The quality of this process is critical to make software projects a success. However, evidence exists that the current state of the practice is still far from being satisfactory; for instance, a study about the current state of RE involving over 300 participants [Swi12], reveals that 70% of them are not, or only somewhat, satisfied with their requirements elicitation, being considered the maturity level of their RE very weak or weak in the case of 31% of the participants.

Without a proper set of requirements the project will fail, no matter how well the rest of the project is executed. A study performed in 1995 by the Standish group [Sta95] showed that only 9% of large companies and 16% of small companies delivered projects on time and within budget. When the participants were asked to report the causes of failed projects, on the top eight factors accounting for failed projects in the report, five of them were related to requirements and their elicitation, being two of them the top two factors: Incomplete requirements (13.1%), and Lack of user involvement during RE (12.4%). Besides the Standish report, more recent studies have also identified requirements as an important risk factor in project failures [Arn11] [PMS11]. It has been further reported that the cost of fixing requirements-based problems increases rapidly the farther into the software development they are discovered.

Eliciting the suitable requirements produces benefits such as preventing errors, improving quality, and reducing risk on software projects [Pro02]. However, there are some problems that often exist in Software Requirements Specifications (SRS), especially when written with natural language: usually requirements are stated in an ambiguous, incomplete and inconsistent manner, and generally they are expressed in an unsystematic way [Yu97]. The SwissQ study cited above [Swi12] corroborates this fact: 74’5% of participants had problems related to ambiguousness, 73’6% problems related to incompleteness, and 61’1% problems related to inconsistency. The main challenge therefore is obtaining unambiguous and consistent requirements that state clearly the complete needs of the system [Hul05].

Requirements reuse has been proposed as a key asset to efficiently elicit, validate and document requirements, obtaining SRS of better quality through more effective engineering processes. In a nutshell, requirement reuse is the concept of taking requirements that have been written for previous projects and then using them in a new project. Due to its proved benefits [Gol13], requirements reuse, its success factors, barriers, and processes are attracting the interest of practitioners and researchers, and it has been examined from a number of different perspectives (e.g., analogy [Mai93], case-based reasoning [Lai94] and generic modelling [Bol94]).

One of the techniques to achieve reuse is the adoption of patterns. In their most classical form, patterns describe problems that occur over and over again, and then describe the core of the solution to these problems [Ale77]. Software engineers have adopted the notion of pattern in several contexts, in particular related to software design (e.g., design patterns [Gam95] and software architectural patterns [Sha95]), but also in other development phases. Following this strategy, some works have focused on the use of patterns for reusing knowledge during RE, such as analysis patterns [Fow97], requirement pattern [Wit07], and product family variability pattern [Kee99]. Unfortunately, as far as we know these ideas have been restricted to small-scale academic examples, and remain largely untested in a genuine commercial capacity.

Based on the previous observations we formulate our research objective (RO) in terms of the following design problem: Define the concept of SRP and provide a framework to facilitate its use, to ultimately encapsulate reusable knowledge (in this case, textural requirements) during RE stage and make its use viable in real requirements elicitation processes. Concrete research questions (RQ) are presented below, following the design science approach to classify them in knowledge problems (KP) and practical problems (PP) [Wie09]:

**RQ1 (KP)** Which are the existent approaches to the notion of pattern in the context of RE knowledge reuse?

**RQ2 (PP)** What is the best structure and semantics software requirement patterns (SRP) should have to be applied over functional (F), non-functional (NF) and non-technical[1] (NT) requirements and to improve the quality of SRS?

**RQ3 (PP)** How SRP can be integrated in the RE stage techniques and processes in order their application gives benefits that justify the cost of their adoption?

**RQ4 (PP)** Does the proposed framework give benefits and drive to higher quality SRS when applied into RE activities?

[1] NT requirements are those ones not referring directly to the intrinsic quality of software, but to the context of the system under analysis (e.g. economic, political and managerial issues)

## 2 Solution

The PABRE framework [Pal14-1] proposes the use of SRP to capture and use requirements knowledge in the context of IT projects. Following the typical context-problem-solution structure of patterns, an SRP basically consists of: a template (solution) that may generate one or more requirements when applied in a certain project, and some information (context-problem) to identify its applicability in that project. Patterns are classified by means of classification schemas, i.e. hierarchies of classifiers that facilitate the identification of relevant SRP during the elicitation process and the organization of requirements in the SRS.

The framework currently embraces: 1) A metamodel that describes the structure of patterns and their organization [Fra10] [Pal11-1] [Fra13]; 2) An SRP catalogue composed by 29 NF-SRP, 37 NT-SRP and 45 F-SRP [Pal11-1] [Pal12] [Fra13] [Pal13-1]; 3) A preliminary method for constructing and evolving SRP, as well as another one for guiding the use of the catalogue in the elicitation [Pal11-1] [Fra13] [Pal13-1]; 4) The PABRE system as technological support [Pal11-2] [Pal13-3]; and 5) A preliminary version of an economic model to perform cost-benefit analysis on the adoption of SRP based on Return on Investment (ROI) [Pal13-4].

The potential benefits of the PABRE framework are:

- Less time required in the elicitation of recurrent requirements (e.g., requirements for different projects addressing the same domain, requirements addressing functionalities shared by different products, requirements addressing certain regulations). Consequently, more time available for the definition of creative requirements (requirements that may change the typical behavior of a product and that provide them an added value).
- Improved quality of SRS (consistency, non-ambiguity, completeness, etc.), thanks to the uniformity of SRP, the dependencies and relationships defined among them and the checklist character that the catalogue may take.

More information of the PABRE assets can be found in their references. In the next subsections, we present an example of SRP and a brief introduction to its metamodel.

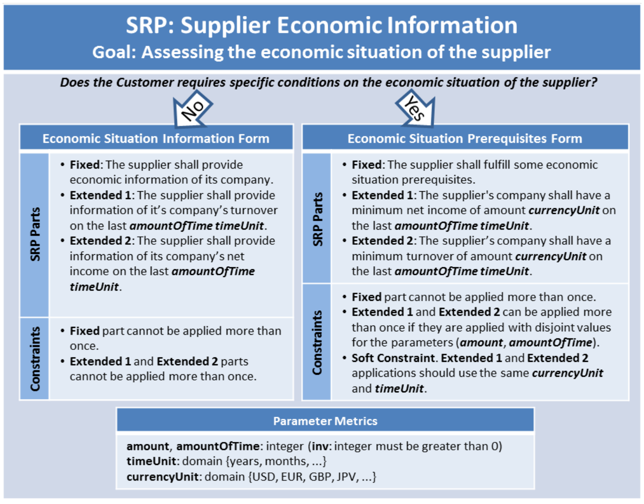
### 2.1 SRP example

An SRP is a pattern that, when applied, produces software requirements related to the objective (goal) of that pattern. The application of the *Supplier Economic Information* SRP (Figure 1) may produce requirements related to the goal of *Assessing the economic situation of the supplier* that procures a software system, as could be the supplier company's turnover or net income.

A goal can be achieved in different ways. An SRP consists of several Forms, each one representing a different *solution* for achieving the goal. In our example, the goal can be attained by asking the supplier the relevant economic information (*Economic Situation Information Form*), or by setting conditions or prerequisites on the economic situation that the supplier should have (*Economic Situation Prerequisites Form*).

We organize Forms into Parts, each of them being a template. Each Form is characterized by a Fixed Part which states the minimal requirement that always holds when applying that form, and some Extended Parts which may be applied or not. The fixed part always becomes a requirement when an SRP is applied with this form. Extended Parts are only used if more precise information is required in the specification.

Due to this nature, the Fixed Part is usually quite generic and hardly measurable. For instance, in our example, the fixed part of the first form is *The supplier shall provide economic information of its company*, whilst the two extended parts identify the type of information required (company's turnover or net income) and the period of time.



**SRP: Supplier Economic Information**
**Goal: Assessing the economic situation of the supplier**

***Does the Customer requires specific conditions on the economic situation of the supplier?***

**Economic Situation Information Form**

SRP Parts

- **Fixed**: The supplier shall provide economic information of its company.
- **Extended 1**: The supplier shall provide information of it's company's turnover on the last ***amountOfTime timeUnit***.
- **Extended 2**: The supplier shall provide information of its company's net income on the last ***amountOfTime timeUnit***.

Constraints

- **Fixed** part cannot be applied more than once.
- **Extended 1** and **Extended 2** parts cannot be applied more than once.

**Economic Situation Prerequisites Form**

SRP Parts

- **Fixed**: The supplier shall fulfill some economic situation prerequisites.
- **Extended 1**: The supplier's company shall have a minimum net income of amount ***currencyUnit*** on the last ***amountOfTime timeUnit***.
- **Extended 2**: The supplier's company shall have a minimum turnover of amount ***currencyUnit*** on the last ***amountOfTime timeUnit***.

Constraints

- **Fixed** part cannot be applied more than once.
- **Extended 1** and **Extended 2** can be applied more than once if they are applied with disjoint values for the parameters (***amount***, ***amountOfTime***).
- **Soft Constraint**. **Extended 1** and **Extended 2** applications should use the same ***currencyUnit*** and ***timeUnit***.

**Parameter Metrics**

**amount**, **amountOfTime**: integer (**inv**: integer must be greater than 0)
**timeUnit**: domain {years, months, ...}
**currencyUnit**: domain {USD, EUR, GBP, JPV, ...}

**Figure 1: Supplier economic Information SRP**

Usually, fixed and extended parts must conform to some part Constraints for declaring multiplicities or dependencies among parts. In the example SRP, aside of constraints on the possible number of appearances of each part in a specific SRS, there exist constraints on the parameters values in each application (see next paragraph).

Both fixed and extended parts are similar from a syntactic point of view. They are composed by the text to be used as a requirement and optionally some parameters to be instantiated when applying the pattern. Parameters establish their Metric, and eventually a correctness condition *inv* (see these definitions on the bottom of Figure 1). The constraints in the use of parts, may also state restrictions on the values of parameters during their application. For instance, in the second form of the example SRP, the application more than once of the *Extended 1* part in an SRS can be done as long as the values of the parameters *amount* and *amountOfTime* in the different applications are different. This allows to state restrictions on the net incomes of the supplier's company for the last 2 years and also for the last 10 years. There is also another type of Soft Constraint that allows giving recommendations in order to maintain the consistency of the SRS document. One example of such constraint is using the same *currencyUnit* in each application of the extended parts of the example SRP.

## 2.2 Metamodel

The aim of a PABRE metamodel (Figure 2) is to define the structure and semantics of an SRP as well as their organization inside a catalogue. In this metamodel we can distinguish four independent, but interrelated, parts.

1. *SRP Core*. It defines the exact structure of a SRP.

2. *Application part*. This part allows adapting an SRP in every project, using parameters and metrics associated to them.
3. *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.
4. *Classification part*. This part defines how SRP can be organized to facilitate its access in a repository.

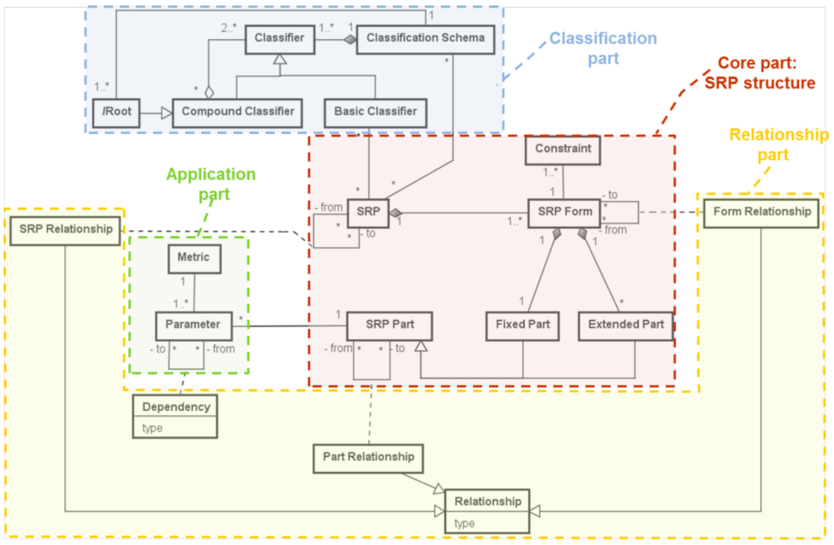


Figure 2: PABRE Metamodel Overview

## 3 Progress, Research Method and Novelty

This PhD thesis is half way to its four year duration. It was started on January 2012 and is expected to be presented on February 2016.

Apart from the design and implementation of the PABRE framework (introduced in Section 2), on the one hand we have been working on a Systematic Literature Review (SLR) [Kit04] to answer the question "What is the State of the Art to reuse knowledge during RE using patterns?". The SLR is finished and the main results can be found in [Pal11-1] [Pal13-4]. On the other hand, in order to provide evidence of the correctness of the framework, some efforts to validate it have already been done. A preliminary version of the framework was advertised at REFSQ'11 [Fra11]. We are also carrying out a survey [Pal13-2] to: firstly, know the current state of RE in relation to reuse in IT organizations as well as the main problems in RE processes; secondly, validate the acceptance of the overall framework. A preliminary analysis of its results can be found in [Pal14-2]. We are currently beginning a study of the current state of requirements reuse in requirements management tools existent in the marketplace.

The future work of this thesis is fully described in [Pal13-4]. Apart from improving the finished assets embraced in the PABRE framework, it will mainly focus on:

- Enrich the economic model by adding more metrics.
- Validate the correctness and suitability of the framework, by testing and evaluating the use of SRP in real elicitation processes through a case study.

This PhD project started as a response to the RE needs of a partner (the Public Research Centre Henri Tudor at Luxembourg). The research approach adopted for this thesis is based in the experimental software engineering paradigm [Bas93], and more specifically in the scientific paradigm. Firstly, the problem was identified with the help of our partner, and complemented the motivation identified from the literature (briefly discussed in Section 1) and the SLR carried out. Then, the scientific problem was defined in the format of RQs. After that, a solution idea was formed, by studying the structure and content of real SRS, and improved afterwards with the SLR. Finally, the solution idea has already been validated using online questionnaires and testing each one of the framework's assets separately, and in the future it will be validated with semi-structured interviews for theoretical evaluation, and case studies for practical evaluation.

As for novelty, although there have been several techniques proposed to reuse knowledge during RE, we are not aware of a concrete proposal that has achieved a wide acceptance. In particular, this thesis proposes a requirements reuse approach that deals with what appears to be missing for reuse being achieved in requirements elicitation as part of regular projects: 1) a concrete proposal of a reusable requirements catalogue; 2) how the requirements reuse process shall be implemented and integrated in organizations; and 3) how to calculate the ROI of the requirements reuse approach.

## Acknowledgements

This work has been supported by the Spanish project TIN2010-19130-C02-01.

## References

[Ale77] Alexander C., Ishikawa S., Silverstein M., Jacobson M., Fiksdahl-King I., Angel S., "A Pattern Language". Oxford University Press, 1977.

[Arn11] Arnuphaptrairong T., "Top Ten Lists of Software Project Risks: Evidence from the Literature Survey". Lecture Notes in Engineering and Computer Science, 2011.

[Bas93] Basili V.R., "The Experimental Paradigm in Software Engineering". International Workshop on ESE Issues: Critical Assessment and Future Directions. Springer London (1993).

[Bol94] Bolton D., Jones S., Til D., Furber D., Green S., "Using domain knowledge in requirements capture and formal specification construction". Requirements Engineering: Social and Technical Issues, Academic Press, London, 1994.

[Fow97] Fowler M., "Analysis Patterns: Reusable Object Models". Addison-Wesley, 1997.

[Fra10] Franch X., Palomares C., Quer C., Renault S., De Lazzer F., "A Metamodel for Software Requirement Patterns". 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'10), 2010. (Work-in-Progress Paper)

[Fra11] Franch X., Guerlain C., Palomares C., Quer C., Renault S., "Interested in Improving Your Requirements Engineering Process? Try Requirement Patterns!" 17th Int. Working Conference on Req Engineering: Foundation for Software Quality (REFSQ'11), 2011. (Short Paper)

[Fra13] Franch X, Quer C, Renault S, Guerlain C, Palomares C, "Constructing and Using Software Requirements Patterns". Chapter in book Managing Requirements Knowledge, Springer, 2013.

[Gam95] Gamma E., Helm R., Johnson R., Vlissides J., "Design Patterns: Elements of Reusable Object-Oriented Software". Addison-Wesley, 1995.

[Gol13] Goldin, L., Berry, D.M., "Reuse of requirements reduced time to market at one industrial shop: a case study". In: Requirements Engineering Journal, 2013.

[Hul05] Hull E., Jackson K., Dick J., “Requirements Engineering” 2nd edition. Springer, 2005.

[Kee99] Keepence B., Mannion M., “Using Patterns to Model Variability in Product Families”. IEEE Software, 16(4), 1999.

[Kit04] Kitchenham B., "Procedures for performing systematic reviews". Keele University TR/SE-0401/NICTA Technical Report 0400011T, vol. 1, 2004.

[Lai94] Lain W., “Reasoning about requirements from past cases”. PhD thesis, Kings College, University of London, 1994.

[Mai93] Maiden N., Sutcliffe A., “Exploiting reusable specification through analogy”. Communications of the ACM, 35(4), 1993.

[Pal11-1] Palomares C., “Including Functional and Non-Technical Requirements in a Software Requirement Patterns Catalogue”. MSc on Computing Thesis, 2011. Available at: http://upcommons.upc.edu/pfc/handle/2099.1/12689. Last access: January 2014.

[Pal11-2] Palomares C., Quer C., Franch X., “PABRE-Man: Management of a Requirement Patterns Catalogue”. IEEE International Req. Engineering Conference (RE'11), 2011. (Tool Demo)

[Pal12] Palomares C., Quer C., Franch X., Guerlain C., Renault S., “A Catalogue of Non-Technical Requirement Patterns”. 2nd International Workshop of Requirements Patterns (REPA'12), at 20th IEEE International Requirement Engineering Conference (RE'12), 2012.

[Pal13-1] Palomares C., Quer C., Franch X., Guerlain C., Renault S., “A Catalogue of Functional Software Requirement Patterns for the Domain of Content Management Systems”. Requirements Engineering Track at 28th ACM Symposium On Applied Computing (SAC’13), 2013.

[Pal13-2] Palomares C., Quer C., Franch X., “Using a Pattern Catalogue in Requirements Engineering Activities”. 19th Int. Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ’13), 2013. (Short Paper)

[Pal13-3] Palomares C., Quer C., Franch X., “PABRE-Proj: Applying Patterns in Requirements Elicitation”. IEEE International Req. Engineering Conference (RE'13), 2013. (Tool Demo)

[Pal13-4] Palomares C., “Definition and Use of Software Requirement Patterns in Req Engineering Activities”, Thesis Proposal, June 2013. Available at: http://www.essi.upc.edu/~cpalomares/publicacions/2013/CPalomares_ThesisProposal_2013.pdf Last access: January 2014.

[Pal14-1] Palomares C., Quer C., Franch X., “Requirements Reuse with the PABRE Framework”. Requirements Engineering Magazine, vol.2014-01, 2014.

[Pal14-2] Palomares C., Quer C., Franch X., “Requirements reuse and patterns: A Survey”. *Accepted in* 20th Int. Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ’14), 2014. (Short Paper)

[PMS11] PM Solution research, “Strategies for Project Recovery – A PM Solutions Research Report”, 2011. Available at: http://www.pmsolutions.com/collateral/research/Strategies%20for%20Project%20Recovery%202011.pdf. Last access: June 2013.

[Pro02] Procaccino J., Verner J., Wvermyer S., Darter M., “Case Study: Factors for Early Prediction of Software Development Success”. Information and Software Technology, vol. 44, 2002.

[Sha95] Shaw M., “Patterns for Software Architectures”. In: Coplien J., Schmidt D., “Pattern Languages of Program Design”. Addison-Wesley, 1995.

[Sta95] The Standish Group, “The Standish Group Report - Chaos”, 1995. Available at; http://www.projectsmart.co.uk/docs/chaos-report.pdf. Last access: June 2013.

[Swi12] SwissQ, “SwissQ Requirements Trends & Benchmarks Switzerland 2012”, 2012. Available at: http://www.swissq.it/wp-content/uploads/2013/03/ SwissQ_Req_Trends_2012_Web_EN.pdf. Last access: June 2013.

[Wit07] Withall S., “Software requirement patterns”. Microsoft Press, 2007.

[Wie09] Wieringa, R., “Design science as nested problem solving”. In Vijay K. Vaishnavi and Sandeep Purao, editors, DESRIST. ACM, 2009.

[Yu97] Yu, E., “Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering”. 3rd IEEE Int. Symp. on Requirements Engineering (RE), 1997.