Journal of Universal Computer Science, vol. 19, no. 2 (2013), 254-280
submitted: 7/5/12, accepted: 31/12/12, appeared: 28/1/13 © J.UCS

Evaluation of a Systematic Approach to Requirements Reuse

Fabiane Barreto Vavassori Benitti
(Universidade do Vale do Itajaí, Itajaí, Brazil
fabiane.benitti@univali.br)

Rodrigo Cezario da Silva
(Universidade do Vale do Itajaí, Itajaí, Brazil
rodrigocezario@msn.com)

Abstract: The benefits of reusing artifacts in the software development process are well-known in the software engineering community, and the earlier in the system development life-cycle reuse is attempted, the more benefit can be expected. Thus, we highlight the reuse of requirement specifications, leading to greater reuse of other artifacts such as models, code and tests. This paper presents an approach to the requirements reuse, supported by a tool that gives suggestions for reuse from requirement patterns, a patterns catalog and traceability between requirements. The efficiency and effectiveness of the approach were evaluated using a quasi-experiment in a university. We conducted a quantitative evaluation of the approach, and an assessment of participants' perceptions regarding the use of the approach and the computational tool. Finally, we performed a qualitative assessment using the GQM method, from the point of view of experts in the area of requirements engineering, in order to obtain more indicators of the feasibility of applying the approach in companies. The results of the quasi-experiment indicate that the approach presented makes the activities of requirement elicitation and specification about 40% more efficient and effective in terms of the way they are conducted, without the support of the approach. Regarding the perceptions on the use, the experimental group positively evaluated the proposed approach and the developed tool. Based on the evaluation by the GQM method, indicators were obtained that the approach assists in activities of requirement elicitation and specification, from the point of view of experts.

Keywords: Requirements reuse; Requirement patterns; Requirements engineering
Categories: D.2.1, D.2.13, M.8

1 Introduction

In recent years, organizations in the area of software development have been searching for best practices in requirements engineering [Young, 04] [Wiegers, 06] [Robertson, 06] [Davey, 08] [Tamai, 09] [Liu, 10]. This search for new practices occurs because organizations have realized that the success of the project is increasingly related to a better understanding of the requirements [Wiegers, 06] [Robertson, 06].

The requirements engineering process is complex and involves a great deal of work, from the requirement elicitation through to the requirement documentation. However, several studies [Griss, 98] [Barber, 99] [Kuloor, 02] [Perednikas, 08] point out that much effort using reuse approaches can be saved in this process.

The reuse of requirement specifications is an attractive alternative because it leads to greater reuse of other artifacts such as models, code, and test plans, thereby extending the benefits provided by the reuse [Keepence, 95] [López, 02]. Other benefits are described in [Robertson, 06], indicating that reuse of requirement specifications can help in the activities of elicitation, analysis, validation and documentation, in addition to providing the highest quality specifications, both in terms of content and syntax.

However, several investigations in this field show that reuse should be made formally, maintaining the integrity of the component and obtaining the benefits of reuse. The use of the "copy / paste" method is rarely beneficial, because writing this new requirement in this way consists of a new specification, therefore it needs to go through a validation process, thus losing one of the main benefits provided by reuse approaches [Keepence, 95]. Moreover, inconsistent information and lack of control of the impact analysis are other problems caused by the "copy / paste" method [Monzon, 08].

The problems with the use of the "copy / paste" method highlights the need for an approach that supports reuse efficiently and deals with the difficulties encountered in the reuse process (reported in [Pooley, 08]): (i) the existence of artifacts for reuse, (ii) the availability of artifacts, (iii) the location of artifacts, (iv) the understanding of the artifacts, and (v) the validity of the artifacts and their integration with the project.

Thus, this paper describes an approach for implementing the reuse of requirements specifications based on three pillars: (i) requirement writing patterns for structuring knowledge in order to reuse requirements, and provide guidelines for writing and selection of the requirement, (ii) Patterns catalog providing a mechanism to facilitate the selection of a pattern, as well as suggesting other related patterns, and (iii) traceability, to identify links between requirements and maximize reuse.

This approach to the application of reuse is intended to assist the analyst in the stages of requirement elicitation and specification. The elicitation phase is supported by suggestions of requirements that the approach provides, using mechanisms based on patterns and traceability. The specification phase takes advantage of the approach the templates provided by the requirements patterns. The approach is presented in detail in section 3.

In section 2 we begin describing concepts and strategies for the reuse of requirements, and we present a comparison between different approaches. Section 3 presents the approach for the reuse of requirements proposed in this paper. Section 4 describes the procedure for evaluating the approach and results. Finally, we present our conclusions and suggest future research directions for this line of work.

2 Background to requirements reuse and related work

The advantages of software reuse are apparent: increased software productivity, higher software quality, shorter software development time, reduced software costs, less personnel, a competitive advantage, and improved software system interoperability [Mussbacher, 99]. Reusable software artifacts include requirements, designs, source code, test plans, test cases, test data, and documentation. The earlier in the software development life cycle reuse is attempted, the greater the potential savings in terms of effort and cost. Sommerville suggests that reuse of already

implemented requirements decreases the risk of writing low quality requirements, and may lead to the reuse of design, code, and test artifacts [Sommerville, 97].

Requirements reuse has been researched since the late 1980s and early 1990s [Finkelsteinm 88] [Reubenstein, 91]. Lam, McDermid and Vickers [Lam, 97a], however, suggest that "there is little evidence ... that requirements reuse is widely practised" and present various steps towards systematic requirements reuse based on their work for Rolls-Smiths Engine Control Ltd. The authors state that these steps have made a considerable contribution to successful reuse in the domain of aero-engine control systems. This work [Lam, 97a] highlighted the need for a strategy for effective reuse of requirements, and since then, some strategies have been found in the literature.

The requirement patterns are viewed by the community as an approach to strong support for reuse [Franch, 10]. According [Fredj, 99], patterns are an efficient solution for reuse for three reasons: (i) the pair problem / solution serves to guide and motivate designers seeking to solve a given problem, (ii) the small size of a pattern improves the usability, as well as its understanding, selection and adaptation, and (iii) the generality of a pattern increases its scope.

Another way of reusing requirements is through traceability between requirements [Dick, 02] [von Knethen, 02]. Traceability, in addition to its main focus which is to support the impact analysis and integration of the changes that occur in the software process, also serves as support for the reuse of artifacts [Dick, 02] [Spanoudakis, 04]. Requirements is an approach that uses traceability to reuse the requirements specifications [von Knethen, 02].

2.1 State of the art

We used a systematic mapping study as a tool for identifying the state of the art in terms of approaches to requirements reuse. Konda and Mandava [Konda, 10] present a systematic mapping study that aimed to find reusable assets other than source code. As we can see, the scope of this research is broader than just the reuse of requirements, however, from this research [Konda, 10] we have established some criteria to determine the state of the art in the group of studies that refer to the reuse of requirements. The criteria used were studies classified as belonging to the area of software requirements and with sufficient information to reproduce/apply the approach and/or studies that show some form of evaluation of such.

Applying these criteria, nine (9) studies were selected. However, one study was discarded (even though it was of interest to the research) because we did not have access it, and did not receive any reply to our request to the authors. We have highlighted that the mapping used as the reference for the studies selection has taken into account models till first half of 2009 [Konda, 10].

Keepence, Mannion and Smith [Keepence, 95] present a guide to help maximize the level of reuse requirements. This guide consists of steps to categorize requirements as: (i) not reusable; (ii) directly reusable; or (iii) based on parameters. As a next step, the authors recommend an analysis of the requirements that cannot be reused to remove specific references and standardize terms. The authors also suggest subdividing the generic specification in more specific specifications.

Lam [Lam, 97b] describes the attempts to promote the reuse of requirements specifications in the aircraft industry using the perspective of case-based reasoning.

The approach consists of two main phases: domain analysis and domain engineering. To this end, the author adapted an existing method for analyzing domain for use, focusing on the reuse of requirements. The phase of domain engineering is supported by a computational tool, which aims to build requirements retrieved from a library of generic requirements, which can be instantiated with the information provided in the forms, and placed in a library project.

The research reported in [Gotzhein, 98] proposes a reuse approach to formally specify system requirements. The researchers show how a requirement specified in natural language for a real-time system has been transformed into a precise and concise specification in a formal way. They demonstrate the potential for reuse of requirements across several examples. Thus, the main contribution is a non-trivial pattern of requirements for real-time systems, considering the appropriate pattern for reuse.

The research described in [Montabert, 05] [Montabert, 09] discusses the reuse of requirements in the usability area. The authors propose a process for reuse of requirements for notification systems, involving claims in a model of hierarchical task analysis (a technique commonly used in the field of usability).

Study [Mikyeong, 05] suggests a method of producing requirements that will be a core asset in the product line. The authors describe a process for developing domain requirements in which commonality and variability in a domain are explicitly considered. They introduce various matrices to ensure objectivity when identifying commonalities and variabilities, and make the processes concrete by defining specification atoms (primitive requirement) and composition rules.

As in [Mikyeong, 05], Monzon's work [Monzon, 08] deals with the systematic reuse of requirements in a family of products. The first step of the approach, each time a new project of the product family starts, is to identify in the traceability trees of related projects those requirements (or requirements sets) that shall be applicable to the new one. These requirements are marked as “parent requirements” in the family. The specifications of each project inherit these requirements, keeping track of where they come from (in the case of strong and weak reuse) and maintaining synchronization in the changes (only in the case of strong reuse).

Another contribution to requirement reuse is presented in [Perednikas, 08]. In this paper the author proposes the reuse of requirements specifications based on the idea of forecasting user needs. The approach consists of a reuse process based on factual knowledge about the source of requirements.

2.2 Comparison between approaches to requirements reuse

Considering the existence of different approaches to requirements reuse, it is pertinent to conduct a comparison between the approaches, seeking to define the differences from the approach proposed in this paper. Thus, our proposal in Table 1 is compared with the other approaches in the literature considering the following criteria:

- The scope of reuse, which seeks to identify the domain of reuse discussed in the research.
- Characteristics of the approach: this criterion seeks to identify and classify the type of method or technique used to support the reuse.
- The support of some type of computational tool: checks whether the approach is supported by a computational tool.

- The use of the knowledge repository for storing software requirements: a repository is a database that stores the collection of components for reuse, and promotes mechanisms for the efficient recovery of these elements. This criterion identifies the existence of a repository in the architectural approach. We understand that an approach aimed at the reuse of requirements should be based on a repository; not adopting a repository for the storage of knowledge that you wish to reuse raises doubts about its effectiveness, and questions such as: Where and how is the knowledge intended for reuse to be found?
- The method of evaluation of the approach (type of assessment): agreeing with the position of Basili [Basili, 07], empirical study plays an important role in the evolution of software engineering, contributing to building a body of knowledge in software engineering, supported by observations and empirical evidence. Thus, this criterion identifies the evaluation method to which the approach has been subjected.

Studies	Scope / Approach	Tool /Repository	Evaluation
[Keepence, 95]	Scope: Software Product Line (SPL) (domain of spacecraft mission planning systems) Approach: This paper proposes a classification of reusable intra-domain requirements components.	Tool: No tool mentioned. Repository: No repository mentioned.	Case Studies. Two user requirements specification from the same domain were analyzed and split into the three classes described.
[Lam, 97b]	Scope: SPL (domain-specific approach from avionics) Approach: Reuse of requirements from abstract forms based on generic collection requirements.	Tool: Tool-based form for instantiation of the generic requirements. Repository: It has a library of generic requirements. The search is performed manually; however, it is aided by the tool support through instantiation of requirements based on the information provided on the forms.	The tool was evaluated using a framework for evaluating domain-specific kits based on kit elements and pre-defined attributes of each kit element.
[Gotzhein, 98]	Scope: Real-time system Approach: The approach is based on requirement patterns. The focus is on the application-driven discovery of an interesting, non-trivial real-time requirement pattern.	Tool: No tool mentioned. Repository: It stores the requirement patterns based on templates. The search is performed manually in file systems, without specific computational aid.	The reuse potential of requirement specifications is presented through a wide range of examples.

[Montabert, 05] [Montabert, 09]	Scope: Domain of notification systems Approach: The methodology proposed by the authors proposes integrates a critical-parameter-based approach to task modeling within a user-centric design framework.	Tool: LINK-UP system (Leveraging Integrated Notification Knowledge through Usability Parameters) is a web-based computer-aided design tool suite intended to improve support for the design of notification systems through reuse. Repository: Claims associated with the task model are stored in a repository and will be recommended through the subsequent reuse of the instance.	Feasibility study (Usability Study) with seven Computer Science students enrolled in an Introduction to Human-Computer Interaction class involving trans-test and post-test questionnaire [28] [29].
[Mikyeong, 05]	Scope: SPL (article discusses newspaper industry and online Travel booking Systems). Approach: Domain requirements are collected and generalized, and are then analyzed and modeled while explicitly managing the commonalities and variabilities.	Tool: A CASE environment, called DREAM, for managing commonality and variability analysis of domain requirements is described. Repository: Domain requirements are stored in the core asset repository.	Case study in the e-Travel Systems (e-TS) domain.
[Monzon, 08]	Scope: Product Family (specifications in military aircraft on-board systems for tactical missions). Approach: The methodology proposes a standard traceability tree with different requirement document types at different abstraction levels, from the contractual specification to the detailed SW specification documents.	Tool: Add-in to a commercial tool (Doors) to provide reuse of product family requirements. Repository: Repository of the commercial tool.	No evaluation mentioned.

[Perednikas, 08]	Scope: No specific area was mentioned. Approach: The reuse the requirements specifications is based on idea of forecasting user needs. The most significant difference from other known approaches is that the information used as input for the requirements reuse process is factual knowledge about users (not only requirements documents)	Tool: No tool mentioned. Repository: No repository mentioned.	No evaluation mentioned.
Proposed approach	Scope: Approach can be applied to any area. Currently used for information systems. Approach: Approach involving recommendations based on requirements of a catalog of patterns (using templates) and the traceability between requirements.	Tool: Web tool that potentializes the reuse from suggestions of new requirements. Repository: Repository that stores the requirements and patterns. The search for reusable requirements is through the support tool when the user receives suggestions for reuse based on patterns and traceability.	Case Study (quasi-experimental evaluation). Evaluation of perception of the tool and approach. Specialist evaluation (GQM method).

Table 1: Comparative analysis of approaches to reuse

In summary, we can affirm that the studies selected demonstrate the efforts made by the software engineering community to reuse software requirement specifications. We can observe that the focus of studies comes from the engineering product line [Keepence, 95] [Monzon, 08] [Lam, 97b] [Montabert, 05] [Montabert, 09] [Mikyeong, 05] highlighting the area related to the aeronautical sector. When not applied to the product line, reuse occurs in a very specific scope, as in case [Gotzhein, 98], which discusses the reuse of non-trivial requirements for real-time systems. Study [Perednikas, 08] does not define a specific scope, i.e. it is assumed that it may be appropriate in any context.

The approaches use different methods/techniques, such as: classification, patterns, task modelling, traceability, forecasting of user needs and commonalities and variabilities of the requirements. There was no prominent method or technique used (discussed in several of the selected studies).

Authors [Young, 04] [Wiegers, 06] [Johnson, 91] [Tracz, 94] recommend the use of tools to support the reuse process, and Wahono [Wahono, 02] highlights the importance of having a repository. In this regard, four of the seven approaches describe the development of a tool, and five studies pointed to the use of a repository.

The approach proposed in this paper brings together some of the methods used (patterns and traceability) and uses them to automatically display suggestions for reuse requirements (mechanism not explored by the works studied). Initially, the approach is focused on the field of information systems (because of the patterns used). The approach is supported by a computational tool and a repository.

An important aspect to be considered concerns the form of evaluations [Alves, 10]. Evaluations are not described in three of the studies considered. When they are considered, the most widely used form is the case study. However, the studies [Montabert, 05] [Montabert, 09] included quantitative assessments (including a trans-test and post-test questionnaire).

3 Proposed Approach

The proposed approach is based on three techniques for enabling reuse suggestions: (i) requirement writing patterns [Toro, 99] [Withall, 07] for assisting in the structuring of knowledge for reuse, (ii) patterns catalog [Withall, 07] [Renault, 09] providing a mechanism to facilitate the selection of a pattern; and (iii) traceability [Dick, 02] [von Knethen, 02] to suggest new requirements from a reused requirement. Using these elements, the possibility was explored of automatically providing suggestions for reuse of requirements, an aspect that is not explored by existing approaches.

Franch et al. [Franch, 10] define software requirement patterns as an artifact that can be used during the requirements elicitation activity and, according to [Tsumaki, 04], their use creates good software artifacts. A systematic mapping of the state of the art concerning requirement writing patterns was performed (protocol is described in [Silva, 11a]), and from this mapping, the patterns proposed by [Withall, 07] were selected as the basis for the proposed approach, because they were more complete and detailed. Based on the pattern, a catalog was developed considering the 37 patterns proposed by [Withall, 07], organized into six distinct domains (seeking to reconcile the proposed areas [Withall, 07] and IEEE Std 830-1998 [IEEE, 98]), as follows: Logical database, external interfaces, system attributes, functional, design constraints and performance.

Also, to facilitate the use of patterns, the catalog is organized according to the following guidelines [Meszaros, 96]:

- Pattern Name: a name by which the pattern can be identified. This should serve as a reference for the problem / solution [Meszaros, 96] and also represent the context of the pattern [Tsumaki, 04].
- Objective: this element should indicate the purpose of the requirement to solve a problem. The objective is similar to the force and must indicate the individual target [Tsumaki, 04].
- Context: describes a situation in the project in which the problems occur [Tsumaki, 04]. Also serves to show the relative importance of forces [Meszaros, 96].
- Problem: The problem field serves to remind the reader of the problem they are trying to solve. According [Tsumaki, 04], the problem has several causes, so a description of the problem consists of two parts: a general description of the problem, and the causes of the problem.

- Forces: This element provides information that should be considered when choosing a solution to a problem.
- Template: The template serves as a starting point for writing a requirement, and can also be seen as the element "solution" that is usually found in a pattern language. The content of this element is an adaptation of what was presented in [Withall, 07]. The main difference is that in our approach, the patterns are strengthened by a framework that helps in the selection and guidance on how to use them.
- Examples: the purpose of this element is to guide with one or more samples using the pattern. The examples start completing the template element.

We emphasize that the base of the catalog is the [Withall, 07] study, including all patterns. However, we have proposed some changes in order to easy the selection and understanding. The main changes were related to the pattern structure adding the elements “Forces” and “Problem”. Also the sections “basic details”, “application” and “discussion” from [Withall, 07] were relocated at the elements “Objective” and “Context”. The Figure 1 illustrates an example of a requirement pattern for "Calculation Formula Requirement", focusing on the Objective, Template and Example fields.

Objective: Use this requirement pattern to specify how to calculate a particular kind of value, or how to determine a value via a process of logical steps.	
Templates:	
Summary	Definition
«Value name» calculation	«Value description» shall be calculated as follows: «Value name» = «Formula» where «Variable 1 name» is «Variable 1 description»; «Variable 2 name» is «Variable 2 description»; … [«Calculation refinements».] [«Applicability limitations».] [«Reference».] [For example, «Example».]
«Value name» determination	«Value description» shall be determined as follows: 1. «Step 1 description». 2. «Step 2 description». 3. … [«Applicability limitations».] [«Reference».] [For example, «Example».]
Examples	
Summary	Definition
Simple interest calculation	Simple interest for a period of not more than one year shall be calculated as follows: Interest = Principal * Interest rate * Period in days / Days in year * 100 Where **Principal** is the monetary amount upon which interest is earned; **Interest rate** is the percentage rate applicable; Period in days is the number of days’ worth of interest, calculated as per the next requirement; **Days in year** is the number of days in the year for which the calculation is being performed, calculated as per the next requirement but one.

Figure 1: Example of requirement pattern [Withall, 07]

3.1 Requirements reuse process - detailing the approach

How are these concepts (requirements patterns, the patterns catalog and traceability) linked in order to enable reuse? Figure 2 shows the proposed process demonstrating the possibilities of reuse and the mechanisms for suggesting requirements for reuse.

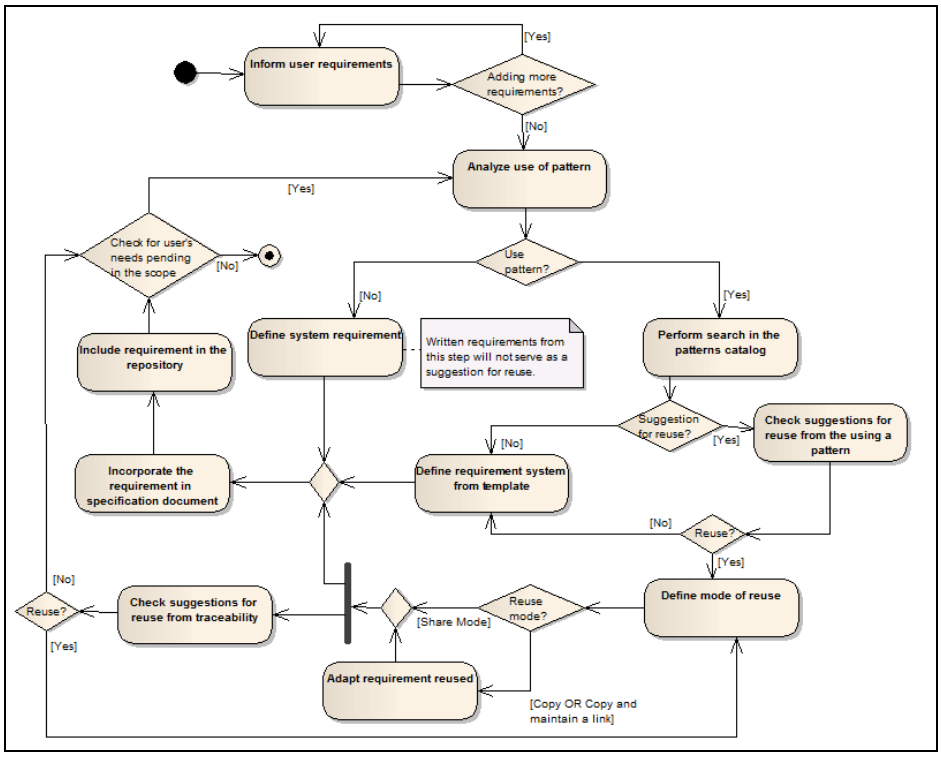

Figure 2: Proposed approach process

We must consider that the process occurs in an iterative way, i.e. the system requirements should be specified in multiple cycles, until all the project requirements are linked to the end of the process generating the artifact requirements specification document.

The process starts from the elicitation of user requirements and may be performed using any conventional technique (suggestions are given in [Zhang, 07]). Subsequently, the user's requirements are "turned into" system requirements - elements that can be reused in the proposed approach. At this point, we can choose to use, or not use a pattern to define the system requirements, but we should note that not using a pattern leads to the traditional process without reuse. In other words, in the proposed approach, the suggestions for reuse will occur from the use of a pattern (we suggest initiating with the patterns previously mentioned, however, this is an aspect of the approach where the user may include new patterns as new projects are being developed and complementary patterns identified).

Thus, the analyst should seek the catalog a pattern that is most appropriate his need. After identifying the pattern, the analyst can create a new requirement from the pattern template proposed, or he can check the suggestions for reuse - in this case, the suggestions are based on requirements of other shared projects, and were written based on the same pattern selected by the analyst. Once the analyst has identified a requirement for re-use, the chosen the reuse mode:

(1) Share Mode: this type of reference links the new requirement with the requirement of the original project (in which the requirement was originally specified). When the original requirement undergoes any kind of alteration, it will be propagated to the related requirements. Thus, this method makes the reuse of the requirement specification entirely and the adaptation of the content of the new requirement is prohibited.

(2) Copy and maintain a link: in this kind of reference, the requirement specification of origin is copied to the new project, maintaining the link to the origin requirement, but allowing the content of the new requirement to be adapted. The objective of this mode of reuse is to keep the link with the original requirement, allowing a comparison between the specifications, so that we can accept or reject the propagation of the changes in the original requirement to the new requirement.

(3) Copy: in this type of reference the requirement specification of origin is copied, without keeping any link between the requirements. In this case, is also possible to adapt the content of the requirement.

Once a new requirement has been defined, it is added to the specification document. Based on the reuse of a requirement, regardless of the mode of reuse, new requirements are suggested for reuse from the traceability links with the origin requirement. And in this case, if the analyst choosing to reuse a requirement must choose one of the modes provided for reuse (for which the procedure is the same as that already described).

It should be emphasized that there is a need to establish traceability between requirements, because this is one of the pillars of the approach. We understand that the lack of these pillars would lead to a considerable reduction in suggested requirements for reuse.

In order to exemplify the dynamic approach, we can assume that it is necessary to define a requirement that specifies the calculation of the freight for the shipment of goods. Using the approach, the analyst performs a search in order to select the appropriate pattern for writing, in this case, "Calculation Formula Requirement Pattern" (as illustrated in Figure 1).

At this point, the analyst may choose to use the pattern as a starting point for writing the requirement, or may select any of the suggested requirements from the selected pattern, the final requirement should be set as shown in Figure 3.

If the analyst chooses to select a requirement from the options suggested for reuse (which uses the Calculation Formula Requirement Pattern and is linked to the calculation of freight), then the mechanisms for supporting the reuse-based traceability link (considering the requirement reused in their source project) may suggest requirements, as listed in Figure 4.

Summary	Definition
Value of freight determination	value of freight shall be determined as follows: 1. calculate the value of the total weight of purchase. 2. calculate the total purchase price (declared value). 3. based on the total weight calculated, the corresponding "weight/destination" value is selected in the price list provided by the mail for express shipment. Upper limit for declaration of value: $ 10.000,00. Reference calculation: table of values for express shipment. For example, total weight of purchase is 1kg, the purchase price is $450.00, the destination is Miami. For this purchase, the freight value is $30.80.

Figure 3: Requirement set from the Calculation Formula Requirement Pattern.

Summary	Definition
Data type for weight of goods	The weight of the goods, which are used for calculation of freight, must be of the type “double”.
Response time for transactions	Each function of customer service (search, shopping cart, shipping calculation, customer records, and so on) should have a response time not exceeding 4 seconds from the correct input (when using minimal configuration required the system). This value is based on indications that users start to lose patience after this time.

Figure 4: Suggestions for reuse from the traceability

3.2 Tool

Based on this, we highlight that the following are essential to enable the implementation of the approach:

- the use of the key elements to promote reuse suggestions - in this case, there is a set of patterns organized in the form of a catalog, and traceability is created between requirements;
- automated support for the proposed process, in the form of a computational tool.

The use of tools that support reuse has many more benefits than informal reuse (i.e. reuse that is conducted on an ad-hoc basis) [Tracz, 94][Wahono, 02] because a computational tool provides mechanisms that facilitate access, identification, selection and understanding of the reusable element stored in the repository. A repository is a database structure that stores a collection of components for reuse, and promote mechanisms for the efficient recovery of these elements.

Thus, this approach is supported by a tool called SERS [Silva, 11b]. This tool is designed to provide specific functionality to implement the mechanisms for reuse of the approach. The functionalities can be divided into two groups:

- Basic Features - these are inherent any tool of software requirements area, including: (a) user account, (b) registration of the project, (c) a register of stakeholders, (d) registration of the sections of the specification document requirements, (e) registration of user requirements, (f) registration of system requirements, (g) traceability between requirements, and (h) printing of the specification requirements.
- Features to support reuse – the differential of the tool: (a) forms for search, selection and application of requirements patterns, (b) Suggested requirements for reuse based on a requirement pattern, and (c) Suggested requirements for reuse based on the traceability links.

Briefly, the suggestions of reuse occur when a requirement is registered. When we select a pattern, the system presents an interface to the registration requirement, allowing the template pattern to be applied (Figure 5a). In this same interface, if the tool finds any requirements for reuse (based on the selected pattern), an alert is displayed to the user (Figure 5b). Accessing the tab "Suggestions for reuse" brings up a list of requirements for reuse (Figure 5c), which can be incorporated into the project in one of the ways described in section 3.1.

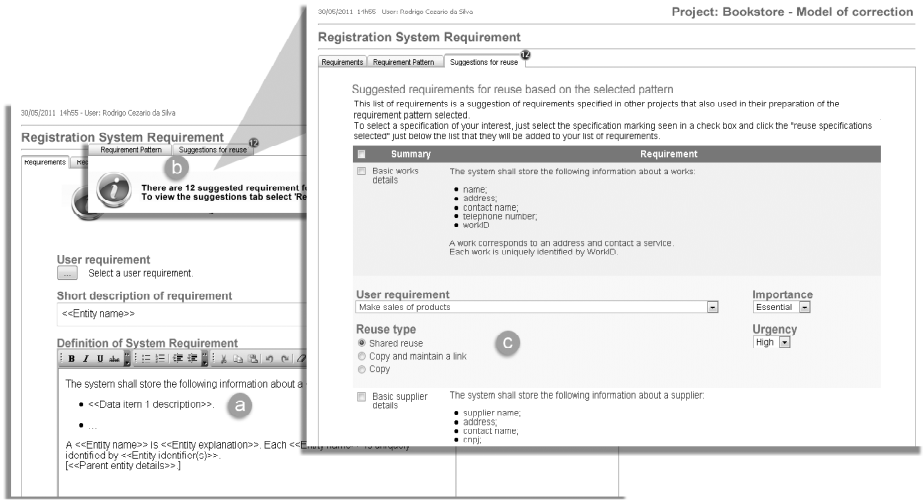

Figure 5: Suggestions for reuse based on the selected pattern (tool interfaces).

By adding to the project requirements reused (Figure 6a), there may be suggestions for reuse from the traceability link (warning as shown in Figure 6b), showing a longer list of candidates for reuse requirements (Figure 6c).

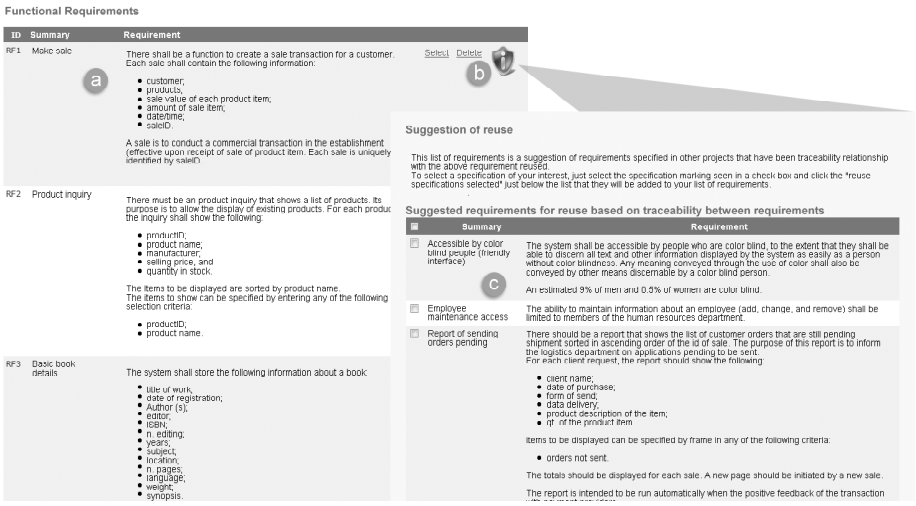

Figure 6: Suggestions for reuse based on the traceability link presented by the tool support.

Thus, the SERS tool plays an important role in the approach, providing interfaces for searching, selecting and applying the requirements patterns. It also assists with interfaces containing suggestions for reuse automatically, these suggestions are patterns-based and traceability links.

Additional information about the tool can be found at [Silva, 11b] including a comparison of SERS with other 7 different tools that implement reuse of requirements. The study has revealed that no tool that gathers the three sharing modes of SERS has been found (as described at section 3.1), as well as no mechanisms for suggesting the reuse of requirements and its patterns as proposed.

4 Evaluation

Some experiments were performed to evaluate the approach and tool. Through the development of case studies, we attempted to identify whether the approach, and the tool, were able to achieve the goal of assisting in the activities of requirements elicitation and specification. The evaluations attempted to measure the effective contribution of the approach and tool in the Requirements Engineering area. In this context the following research questions and hypotheses were established:

Q1 – Does the approach proposed for reuse of requirements make the activities of requirements elicitation and specification more **efficient** compared to	H01: There is no difference in the efficiency of activities of requirements elicitation and specification when using the reuse approach compared with not using it. HA1: The efficiency of activities of requirements elicitation and specification based on the approach

not using the approach?	using reuse is greater than when this approach is not used.
Q2 – Does the approach proposed for reuse of requirements make the activities of requirements elicitation and specification more **effective** compared to not using the approach?	H01: There is no difference in the effectiveness of activities of requirements elicitation and specification when using the reuse approach compared with not using it. HA1: The effectiveness of activities of requirements elicitation and specification based on the approach using reuse is greater than when this approach is not used.

Efficiency was calculated as the ratio between the number of requirements described correctly and the time spent performing the activities of requirements elicitation and specification. Effectiveness was measured as the ratio between the number of correctly described requirements and the total number of requirements that exist. For this purpose, we used a requirements specification document, reviewed and approved by a specialist, as correction model of the case studies used in the evaluation.

In addition to measuring the efficiency and effectiveness of the approach, we also evaluated the perception in the use of the tool/approach and performed a GQM (Goal-Question-Metric) evaluation [Basili, 94] with specialists/professionals in the area of requirements.

4.1 Design of experiment

The experiment involved twenty-four students in the discipline of Software Engineering of an undergraduate degree in Computer Science, the content of which included Requirements Engineering. The process followed in the evaluation is shown in Figure 7.

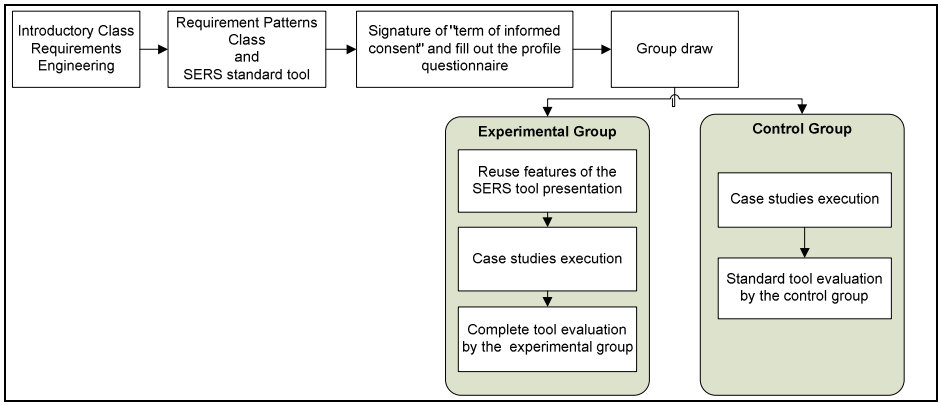

Figure 7: Process adopted in the evaluation

Initially, we gave an introductory lesson (3h) on requirements engineering, covering the basic steps of the process (elicitation, analysis, specification and

validation) as well as requirements definition and types (user and system - functional and non-functional). Subsequently, we gave a lecture (2h) on requirement patterns, then we presented the SERS tool in standard version[1] of the tool (described in section 3.2). Later, we explained to the class the context of research, and asked to the students to sign the consent form and complete a profile questionnaire.

Having overcome the initial stages, the participants were randomly assigned to two groups (experimental and control) and separated into different rooms. In the next moment, the functionality of the tool for reuse was presented to the experimental group.

The case studies used in the quasi-experiment[2] were the development of an online bookstore and a car rental system. Based on the case studies, the participants were asked to identify and specify the functional and nonfunctional requirements. As the experiment was carried out in pairs, there were six pairs in the control group and six pairs in the experimental group. In each group, three pairs performed the case study of the bookstore and the other three pairs performed the case study of the rental cars.

Thus, the experiment consisted of producing a requirements specification for the case study with the support of the SERS tool. In the case of the control group, the application of the templates of the patterns was performed on an ad-hoc, looking for them directly in the pattern catalog available on the web. The experimental group used all the features of the tool, and benefited from suggestions of requirements reused from projects previously registered in the repository tool. In this case, there were three shared projects, the first in the field of office automation (waterproof products and services company), the second in the field of video rental, and third in the field of automobile sales. It is important to note that the projects available for reuse were distinct from the case studies, but they provided requirements suitable for reuse.

After carrying out the proposed activity in the case studies, the participants answered a perceptual questionnaire to identify their impressions on the use of the tool and the approach to reuse.

4.1.2 Results of the experiment

Results were obtained for three aspects: (i) the time spent by each group in order to identify requirements, (ii) the outcome of the case study of each group (obtained directly from the tool repository), and (iii) the perception questionnaire.

4.1.2.1 Evaluation of the Approach

After performing the quasi-experiment, all the data were collected and arranged in a table for analysis, seeking to answer research questions. The correction model for case study 1 (online bookstore) provided 9 functional requirements and 9 non-functional requirements, totaling 18 requirements. The correction model case study 2

---

[1] Standard refers to the basic functionality of the tool being suppressed features regarding the selection/application of patterns and all the mechanisms suggested for reuse.

[2] A quasi-experimental design is one that looks a bit like an experimental design but lacks the key ingredient -- random assignment [Trochim, 06] Trochim, W.M.K., J.P. Donnelly, Research methods knowledge base, in, 2006. Because the selection of participants was restricted to the class indicated, it is considered a quasi-experiment.

(car rental) provided a total of 22 requirements, of which 15 were functional and 7 non-functional. Table 2 summarizes the results, for the experimental group (A) and the control group (B).

Group	Q.Enc	Q.Esp	Q.Reuse	Time	Efficiency	Effectiveness
			CASE STUDY 1			
A1	8	9	1	100	0.080	0.444
A4	7	9	4	99	0.070	0.388
A6	4	5	2	101	0.039	0.222
B3	5	24	-	105	0.047	0.277
B4	2	10	-	105	0.019	0.111
B5	2	5	-	105	0.019	0.111
			CASE STUDY 2			
A2	7	10	7	110	0.063	0.318
A3	6	9	4	102	0.058	0.272
A5	7	20	4	94	0.074	0.318
B1	4	10	-	105	0.038	0.181
B2	0	11	-	105	0	0
B6	4	6	-	105	0.038	0.181

Q.Enc is the number of correctly specified requirements (in relation to the correction model). Q.Esp matches the total number of requirements specified by the group. Q.Reuse is the amount of requirements reused. Time corresponds to minutes used in the experiment, and finally the result of calculating the efficiency and effectiveness.

Table 2: Results of Case Studies

The results of case study 1 show that only one pair from the control group (B3) had a better result in relation to efficiency and effectiveness, and their results were only better than those of one of the experimental groups (A6). Even so, we can note that group B3 specified 24 requirements (far more than the other groups) in order to obtain only 5 requirements.

Analysing the results of case study 2, we note that groups supported by the approach obtained better results for both the efficiency and effectiveness of the activity. Figure 8 (efficiency analysis) and Figure 9 (effectiveness analysis) help to illustrate the observations in the two groups.

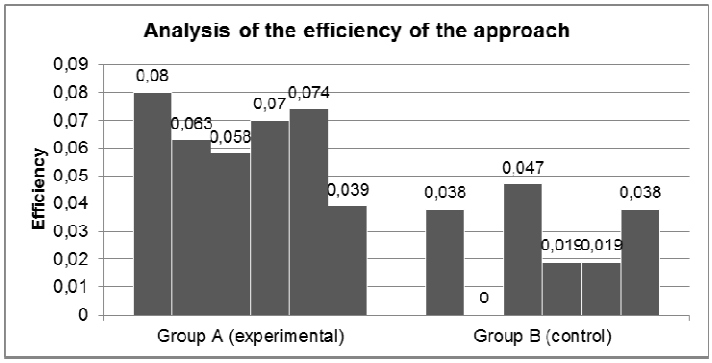

Figure 8: Analysis of the efficiency of the approach

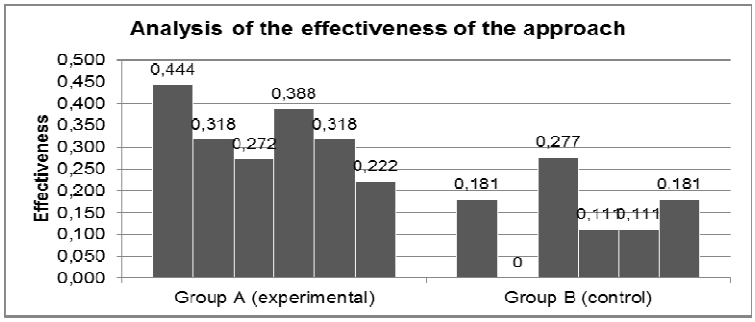

Figure 9: Analysis of the effectiveness of the approach

Analysing only those projects that resulted in suggestions for reuse, through the percentage of reuse (% Reuse) compared to the total requirements found (Table 3), we perceive that in both groups (A4 and A6), the reuse rate was at least 50% in relation to requirements found in case study 1, and the mean was of 46.82% reuse in these projects. Regarding reuse in case study 2, one group (A2) has achieved 100% reuse; the average was 74.60% reuse.

Another aspect that can be analyzed is about the amount of reuse by requirement type. We can observe that the amount of functional requirements reused was higher, except for group A5 (which was equal in both categories). This fact contradicts some studies that claim non-functional requirements are more suitable for reuse [13]. However, this is still an open issue for further investigation.

Group	Q. ERF	Q. ERNF	QR. RF	QR. RNF	% Reuse
A1	2	1	1	0	33.33%
A2	5	2	5	2	100%
A3	4	2	3	1	66.66%
A4	3	4	3	1	57.14%
A5	5	2	2	2	57.14%
A6	2	2	2	0	50%

Q.ERF is the number of functional requirements found. Q.ERNF corresponds to the number of non-functional requirements found. QR.RF is the number of functional requirements reused. QR.RNF the amount of non-functional requirements reused. Finally, % Reuse is the percentage of reuse in the project.

Table 3: Analysis of the percentage of reuse

We also analyze the use of mechanisms to support reuse (shown in Figure 10). This analysis can be performed based on the suggestions from the requirements patterns (at the time of selection of a pattern) and traceability (being carried out after the reuse of a requirement).

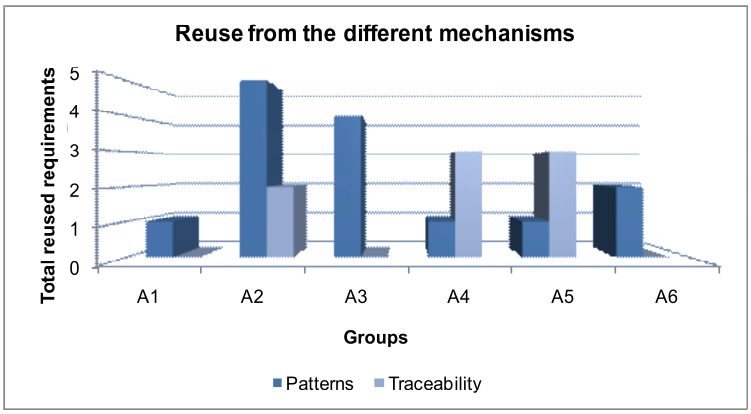

Figure 10: Reuse of requirements from the mechanisms

The results show that the majority of the reuse acceptance was promoted by suggestions based on patterns, this aspect was expected, since the suggestions for reuse from traceability are a result of the use of a pattern. However, we highlight groups A4 and A5, which found the greatest benefit from the suggestions of traceability. These data provide indications that support mechanisms favoring reuse.

Finally, Figure 11 shows the results for the comparison of the two groups regarding efficiency and effectiveness. The average efficiency of the experimental group was 0.064 and that of the control group was 0.026. For effectiveness, the result for the experimental group was 0.327, while that of the control group was 0.143.

Based on the results shown in Figure 11, wee see that the mechanisms for reuse brought 40.62% greater efficiency in the experimental group. We can see that the conduct of the proposed activity in the case study which used the support of the approach had 43.73% greater efficiency than the conduct without the use of the approach.

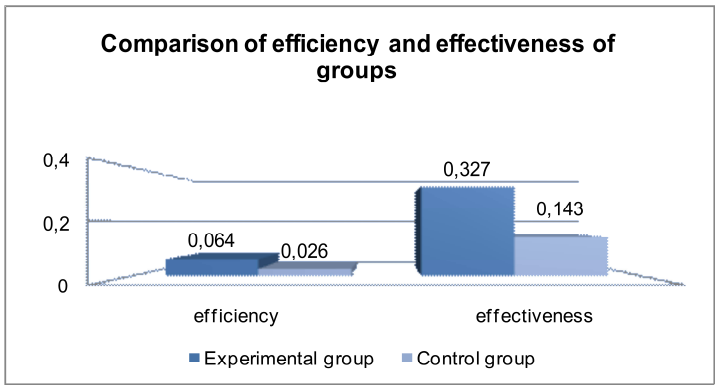

Figure 11: Comparative analysis of the results obtained by the two groups

Additionally we performed a statistical analysis to accept or reject the hypotheses. Thus, we applied the Mann-Whitney statistical test, which is appropriate because the sample does not follow a normal distribution.

We selected the lowest value of U for efficiency, to obtain a value of 1. Taking the lowest value of U for effectiveness, the result was 2. Thus, according to the statistical test, with the value of the intersection between n1 = 6 and n2 = 6 with α = 0.05, we obtain a value of Uc = 5. Based on these results, we can reject the hypotheses H01 (There is no difference in the efficiency of activities of requirements elicitation and specification when using the reuse approach compared with not using it) and H02 (There is no difference in the effectiveness of activities of requirements elicitation and specification when using the reuse approach compared with not using it).

Thus, our results indicate that the proposed approach assists in the activities of requirements elicitation and specification, making them more efficient and effective. However, we believe these results are merely indicative, and further case studies are needed to provide stronger evidence of the benefits of the approach.

4.1.2.2 Perception evaluation of the use of the tool

Through the perception questionnaire which the participants filled out at the end of the experiment, their impressions of the SERS tool were obtained. For the questionnaire, we used a 5-point Likert scale [Ferguson, 41] to indicate the participants' level of agreement. Figure 12 shows the results of the questionnaires for the experimental group participants.

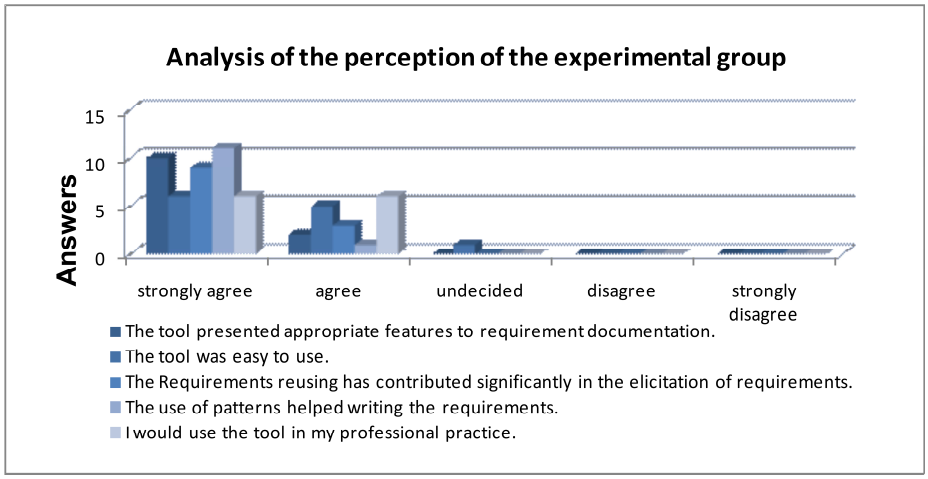

Figure 12: Results of the perception questionnaire in the experimental group.

Based on participants' perceptions, we highlighted the very positive result for the contribution of reuse in the activities of requirements elicitation and specification, as well as the perception that the patterns helped in writing the requirements.

4.2 Evaluation of specialists through the GQM method

This evaluation was conducted in an attempt to reduce the bias that emerges from the analysis of the approach in an academic environment, where participants probably do not have the desired level of experience to work in the industry. We therefore decided to conduct an evaluation by experts in the field of requirements engineering, in order to obtain more indicators of the feasibility of implementing the approach. That this evaluation is exploratory in nature, and some points need to be improved in order to obtain more reliable results. In this evaluation, the GQM method was used [Basili, 94] to obtain an assessment of the approach from the point of view of six (6) professionals in requirements engineering.

Goal	To evaluate an approach for the reuse of requirements related to the efficiency and effectiveness under requirements engineering experts point of view at the context of especification and documention of software requirements.
Questions	**Metrics**
Q1. What is your subjective impression of efficiency (gain in time) on the percentage of improvement in the activity of elicitation and documentation requirements obtained when using the approach, compared to not using it?	MQ1. Subjective impression on the percentual variation of time applying the approach and comparing to not using it.
Q2. The software artifact (requirements specification document) produced with the approach is more complete and accurate than the one produced in another approach?	MQ2.1. Subjective impression on the percentual variation of the completeness of the document with the use of the approach compared to not using. (Understanding "completeness" as the amount of requirements correctly identified) MQ2.2. Subjective impression of the percentage variation of the correctness of the document with the use of the approach compared with no use. (Understanding "correctness" as the amount of requirements correctly described)
Q3. Has the use of patterns/catalog helped guiding the written of requirements?	MQ3.1. Subjective impression on the percentage of help in writting the requirements when applying the patterns proposed by the approach. MQ3.2. Grade of subjective acceptance on the easiness to find an appropriate pattern on the catalog.
Q4. Were the reuse suggestions adequate?	MQ4.1. Grade of subjective acceptance of the suggestions made for the pattern.

	MQ4.2. Grade of subjective acceptance of the suggestions made for the traceability.
Q5. Has the support tool contributed with any improvement on the activity execution?	MQ5.1. Subjective impression on the percentage of the benefits provided by the tool. MQ5.2. Grade of subjective acceptance about the benefits provided by the tool.

Figures 13 and 14 show the result of the evaluation, using the GQM method, for the questions answered after using the approach.

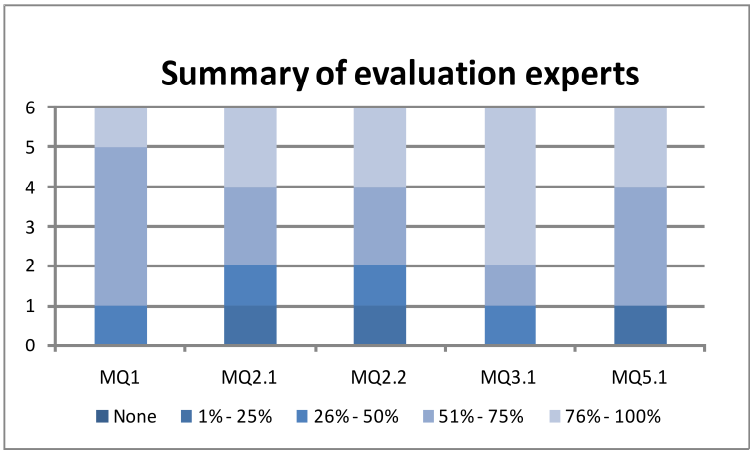

Figure 13: Results of the evaluation with the GQM method as percentages

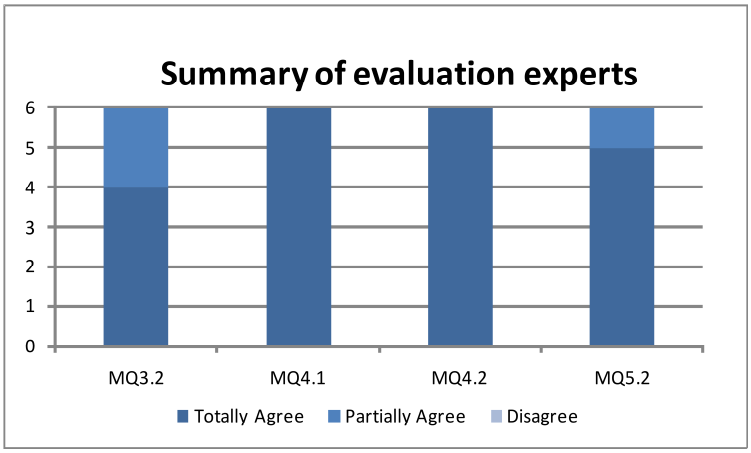

Figure 14: Results of the evaluation with the GQM method as acceptance grade

According to the data collected, we observed, in general, that there are indications that the proposed approach assists in the activities of requirements elicitation and specification, according to the views of the experts in the field of requirements engineering.

5 Conclusions

Reuse is considered as an alternative for assisting in requirements engineering activities. In general, researchers [Spanoudakis, 96] [Massonet, 97] [Cybulski, 00] [Robertson, 06] argue that requirements reuse provides a reduction in development times and improves the quality of the developed product, thereby leading to quality improvement in the requirements engineering process. However, a reuse approach rarely presents benefits without a systematization or a process [Perednikas, 08]. Considering the approaches found in the literature, we did not find any empirical evaluations to determine the efficiency and effectiveness of the proposed approaches.

This article therefore describes an approach to requirements reuse, seeking to determine whether this approach would make the activities of requirements elicitation and specification more efficient and effective. The approach is based on three pillars (already explored separately in other approaches): (i) requirement writing patterns for structuring knowledge in order to assist reuse; (ii) Patterns catalog providing a mechanism to facilitate the selection of a pattern; and (iii) traceability to identify new requirements from a requirement reused (this aspect has not been explored by existing approaches).

The results of this quasi-experiment indicated that the efficiency and effectiveness of the approach were higher in the experimental group compared to the control group. However, we emphasize that the quasi-experiment was restricted to 2 hours, so we cannot affirm that the results would be the same after more prolonged use of the approach. Another aspect to consider relates to the repository for reuse; we understand the challenges involved in the initial use of the approach are higher, because we do not have a repository of artifacts available for reuse. We stress that the similarity among the shared projects at repository and the project being specified impacts at the result, having a tendency of the reuse achieving better results in product lines (as observed at section 2.2). In this sense, the repository used and the case studies applied represent a threat to the results of the quasi-experiment, thus being recommended to repeat the experiment with different repositories and case studies.

The use of the questionnaire to evaluate perceptions showed that the experimental group evaluated the tool positively in relation to its features, and considered that the patterns and suggestions for requirements reuse helped in writing the document specification requirements.

In attempting to reduce the bias that arises from the evaluation of the approach in an academic environment, where participants probably do not have the desired experience to work in the industry, we decided to conduct an evaluation with experts in the area of requirements engineering, to obtain better indicators of the feasibility of implementing the approach. The results of the evaluation by the GQM method [Basili, 94] indicated that the approach assists in the activities of requirements elicitation and specification, in the views of six experts (with experience in activities related to Requirements Engineering of between three and seven years). However, we also

recommend performing the evaluation study in real scenarios with expert developers as future work.

In another proposal for future work, we can explore aspects of requirements variability to refine the suggestions for reuse. Furthermore, the development of a plugin that implements the approach in an existing tool, such as Eclipse, DOORS, RequisitePro or Enterprise Architect, would enable further evaluations to be obtained in a commercial context. We emphasize that the patterns used in the experiment are from Withall's catalog [Withall, 07] in its full version (without excluding any pattern). However, we encourage investigations for other patterns to complement this catalog. Another interesting point would be to add more templates to the current patterns. We note that even with a template as a starting point for specifying a requirement, one may find other solutions (templates) that can be added to the pattern, in order to supplement it. Also, to extend the approach and perform new experiments with other artifacts, such as models, test plans, and code, from requirements is a viable path for future projects.

References

[Alves, 10] Alves, V., Niu, N., Alves, C., Valen, G.: Requirements engineering for software product lines: A systematic literature review, Inf. Softw. Technol., 52, 2010, pp. 806-820.

[Barber, 99] Barber, K.S., Graser, T.J., Jernigan, S.R., Silva, J.: The Systems Engineering Process Activities (SEPA) - supporting early requirements analysis and integration prior to implementation design, in: Software Technology and Engineering Practice, 1999. Proceedings, 1999, pp. 50-59.

[Basili, 07] Basili, V.R.: The role of controlled experiments in software engineering research, in: Proceedings of the 2006 international conference on Empirical software engineering issues: critical assessment and future directions, Springer-Verlag, Dagstuhl Castle, Germany, 2007, pp. 33-37.

[Basili, 94] Basili, V., G. Caldiera, H.D. Rombach, Goal Question Metric Approach, in: Encyclopedia of Software Engineering, John Wiley & Sons, 1994, pp. 528-532.

[Cybulski, 00] Cybulski, J.L., K. Reed, Requirements Classification and Reuse: Crossing Domain Boundaries, in: Proceedings of the 6th International Conerence on Software Reuse: Advances in Software Reusability, Springer-Verlag, 2000, pp. 190-210.

[Davey, 08] Davey, B., Cope, C.: Requirements Elicitation – What's Missing?, Issues in Informing Science and Information Technology, 5, 2008.

[Dick, 02] Dick, J. Rick Traceability, in: 1st International Workshop on Traceability in Emerging Forms of Software Engineering, Edinburgh, Scotland, 2002.

[Ferguson, 41] Ferguson, L.W., A Study of the Likert Technique of Attitude Scale Construction, Journal of Social Psychology, 1941, pp. 51-57.

[Finkelsteinm 88] Finkelstein, A.: Re-use of formatted requirements specifications, Software Engineering Journal, 3, 1988, pp. 186-197.

[Franch, 10] Franch, X., Palomares, C., Quer, C., Renault, S., Lazzer, F.: A Metamodel for Software Requirement Patterns, Lecture Notes in Computer Science, 6182 (2010) 85-90.

[Fredj, 99] Fredj, M., Roudies, O.: A pattern based approach for requirements engineering, in: Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on, 1999, pp. 310-314.

[Gotzhein, 98] Gotzhein, R., Kronenburg, M., Peper, C.: Reuse in Requirements Engineering: Discovery and Application of a Real-Time Requirement Pattern, in: Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, Springer-Verlag, 1998.

[Griss, 98] Griss, M.L., Favaro, J., d'Alessandro, M.: Integrating feature modeling with the RSEB, in: Software Reuse, 1998. Proceedings. Fifth International Conference on, 1998, pp. 76-85.

[IEEE, 98] IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, 1998.

[Johnson, 91] Johnson, W.L., Harris, D.R.: Sharing And Reuse Of Requirements Knowledge, in: Knowledge-Based Software Engineering Conference, 1991. Proceedings., 6th Annual, 1991, pp. 57-66.

[Keepence, 95] Keepence, B., Mannion, M., Smith, S.: SMARTRe requirements: writing reusable requirements, in: Systems Engineering of Computer Based Systems, 1995., Proceedings of the 1995 International Symposium and Workshop on, 1995, pp. 27-34.

[Konda, 10] Konda, B.M., Mandava, K.K.: A Systematic Mapping Study on Software Reuse, in: Department of System and Software Engineering, Blekinge Institute of Technology, Sweden, 2010.

[Kuloor, 02] Kuloor, C.: Requirements Engineering for Software Product Lines, in: Dept. of Electrical and Computer Engineering, University of Calgary, Calgary, Alberta, 2002.

[Lam, 97a] Lam, W., McDermid, T.A., Vickers, A.J.: Ten steps towards systematic requirements reuse, in: Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on, 1997, pp. 6-15.

[Lam, 97b] Lam W.: Achieving requirements reuse: a domain-specific approach from avionics, J. Syst. Softw., 38, 1997, pp. 197-209.

[Liu, 10] Liu, L., Li, T., Peng, F.: Why Requirements Engineering Fails: A Survey Report from China, in: 8th IEEE International Requirements Engineering Conference, Sydney, NSW, 2010, pp. 317-322.

[López, 02] López, O., Laguna, M.A., García, F.J.: Metamodeling for Requirements Reuse, in: Workshop em Engenharia de Requisitos, Valencia, Spain, 2002, pp. 76-90.

[Massonet, 97] Massonet, P., A. van Lamsweerde, Analogical reuse of requirements frameworks, in: Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on, 1997, pp. 26-37.

[Meszaros, 96] Meszaros , G., J. Doble, MetaPatterns: A Pattern Language for Pattern Writing, in: 3rd Pattern Languages of Programming conference, Allerton Park, Illinois 1996, pp. 529-574.

[Mikyeong, 05] Mikyeong, M., Keunhyuk, Y., Heung Seok, C.: An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line, IEEE Transactions on Software Engineering, 3, 2005, pp. 551-569.

[Montabert, 05] Montabert, C., Bussert, D., Gifford, S.S., Chewar, C.M., Mccrickard, S.: Supporting Requirements Reuse in Notification Systems Design through Task Modeling, in: 11th International Conference on Human-Computer Interaction Las Vegas, NV, 2005.

[Montabert, 09] Montabert, C., McCrickard, D.S., Winchester, W.W., Pérez-Quiñones, M.A.: An integrative approach to requirements analysis: How task models support requirements reuse in a user-centric design framework, Interact. Comput., 21, 2009, 304-315.

[Monzon, 08] Monzon, A. A Practical Approach to Requirements Reuse in Product Families of On-Board Systems, in: International Requirements Engineering, 2008. RE '08. 16th IEEE, 2008, pp. 223-228.

[Mussbacher, 99] Mussbacher, G.: Combining Case Based Reasoning and Commonality Analysis for Software Requirements Reuse, in: School of Computing Science, Simon Fraser University, Canada, 1999.

[Perednikas, 08] Perednikas, E.: Requirements Reuse Based on Forecast of User Needs, in: International Conference 20th EURO Mini Conference "Continuous Optimization and Knowledge-Based Technologies" Neringa, Lithuania, 2008, pp. 450–455.

[Pooley, 08] Pooley, R., Warren, C.: Reuse through Requirements Traceability, in: Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on, 2008, pp. 65-70.

[Renault, 09] Renault , S., Ó. Méndez-Bonilla, X. Franch, C. Quer, A Pattern-based Method for Building Requirements Documents in call-for-tender Processes, International Journal of Computer Science and Applications, 6 (2009) 175-202.

[Reubenstein, 91] Reubenstein, H.B., Waters, R.C.: The Requirements Apprentice: automated assistance for requirements acquisition, Software Engineering, IEEE Transactions on, 17, 1991, pp. 226-240.

[Robertson, 06] Robertson, S.R.J.: Mastering the Requirements Process, 2 ed., Addision Wesley Professional, 2006.

[Silva, 11a] Silva, R.C., F.B.V. Benitti, Padrões de Escrita de Requisitos: um mapeamento sistemático da literatura, in: 14th Workshop on Requirements Engineering Rio de Janeiro, 2011, pp. 259-271.

[Silva, 11b] Silva, R.C., F.B.V. Benitti, SERS: Uma Ferramenta de Apoio ao Reuso de Requisitos, in: II Congresso Brasileiro de Software: Teoria e Prática (CBSoft) - XVIII Sessão de Ferramentas, São Paulo, 2011.

[Sommerville, 97] Sommerville, I. Sawyer, P.: Requirements Engineering: A good practice guide, John Wiley & Sons, England, 1997.

[Spanoudakis, 04] Spanoudakis, G., Zisman, A.: Software Traceability: A Roadmap, in: Handbook of Software Engineering and Knowledge Engineering, World Scientific Publishing, 2004, pp. 395-428.

[Spanoudakis, 96] Spanoudakis, G., P. Constantopoulos, Analogical Reuse of Requirements Specifications: A Computational Model, Applied Artificial Intelligence, 10 (1996).

[Tamai, 09] Tamai, T., Kamata, M.I.: Impact of Requirements Quality on Project Success or Failure, 14, 2009, pp. 258-275.

[Toro, 99] Toro, A.D., B. Bernárdez, A. Ruíz, M.A. Toro, Requirements Elicitation Approach Based in Templates and Patterns, in: Workshop on Requirements Engineering, Buenos Aires, Argentina 1999.

[Tracz, 94] Tracz, W. Software reuse myths revisited, in: Proceedings of the 16th international conference on Software engineering, IEEE Computer Society Press, Sorrento, Italy, 1994, pp. 271-272.

[Trochim, 06] Trochim, W.M.K., J.P. Donnelly, Research methods knowledge base, in, 2006.

[Tsumaki, 04] Tsumaki, T., Requirements engineering pattern structure, in: Software Engineering Conference, 2004. 11th Asia-Pacific, 2004, pp. 502-509.

[von Knethen, 02] von Knethen, A., Paech, B., Kiedaisch, F., Houdek, F.: Systematic requirements recycling through abstraction and traceability, in: Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on, 2002, pp. 273-281.

[Wahono, 02] Wahono, R.S.: On the Requirements Pattern of Software Engineering, in: Temu Ilmiah XI 2002.

[Wiegers, 06] Wiegers, K.E.: More About Software Requirements: Thorny Issues and Practical Advice, Microsoft Press, 2006.

[Withall, 07] Withall, S., Software Requirement Patterns, Microsoft Press, Washington, 2007.

[Young, 04] Young, R.R.: The Requirements Engineering Handbook, Artech House, 2004.

[Zhang, 07] Zhang, Z., Effective Requirements Development - A Comparison of Requirements Elicitation techniques, in: SQM2007 conference, Staffordshire, UK, 2007.