Barcelos and Penteado *Journal of Software Engineering Research and Development* (2017) 5:3
DOI 10.1186/s40411-017-0038-9

Journal of Software Engineering Research and Development

RESEARCH Open Access

Elaboration of software requirements documents by means of patterns instantiation

Leonardo Vieira Barcelos[1] and Rosângela Dellosso Penteado[2*]

* Correspondence: rosangela@dc.ufscar.br
[2]Department of Computing, Universidade Federal de São Carlos, UFSCar, São Carlos, SP, Brazil
Full list of author information is available at the end of the article

Abstract

Studies show that problems associated with the requirements specifications are widely recognized for affecting software quality and impacting effectiveness of its development process. The reuse of knowledge obtained from previous projects can facilitate the identification and writing of the requirements to reach the elaboration of a complete and consistent requirements document. Software patterns are a solution to capture and reuse knowledge from different contexts for the software development. On the information system domain, it is common to find a set of requirements that have similar characteristics and repeat themselves in different systems of this domain, indicating a possibility of becoming a requirements pattern. By reusing knowledge obtained in previous projects, requirements patterns can be elaborated to facilitate a more complete and consistent specification of system requirements in that domain. The advantage of using requirements patterns is to provide the software engineer a starting point for the elaboration of requirements documents, instead of starting from scratch to solve known problems. The requirements patterns can contribute to the improvement of the software quality, minimize the development time and costs, because previously defined and tested solutions are reused. This paper shows a set of functional requirements patterns and business rules elaborated to help the software engineer in the writing of the requirements document. The non-functional requirements were not considered in this project. A computational support was developed, based on that set of patterns, to facilitate the organized and complete writing of the requirements document. Case studies are presented with the use of the computational support.

Keywords: Requirements document, Requirements patterns, System information domain, Computational support based on requirements pattern

1 Background

1.1 Contextualization

Problems related to Requirements Engineering (RE) are widely recognized for affecting software quality and impacting effectiveness in its development process (Niazi and Shastry 2003). It is estimated that finding and fixing a problem after software delivery can be 100 times more expensive than doing this during the early stages of development (Boehm and Basili 2001). In this sense, there are reports stating that complete

understanding and specification of requirements are among the most difficult tasks faced by a software engineer (Pressman 2009).

Currently, there is greater concern in the elaboration of a Requirements Document (RD) that not only meets the requirements of stakeholders, but can also clearly identify omissions and mistakes. There is an increasing need for effective stakeholder participation in the preparation and checking of this document.

Chernak (2012) reports that there are three main forms of requirements documentation in the IT industry: the traditional RE (48%), the one directed by use cases (28%), the agile methods (11%) and other approaches that represent (13%). The data shows that the traditional RE corresponds to the principal form of requirements documentation. So, the requirements are usually written in natural language, which provides a common and intuitive understanding in the communication between stakeholders (Paldês et al. 2016). However, despite the flexibility of the natural language, there may be ambiguity, inconsistency, and incompleteness in requirements specifications (Fatwanto 2013).

There are common problems in requirements elicitation such as the difficulty of stakeholders to express their needs and of software engineers, without the client domain experience, in understanding them completely (Sommerville 2011). When the software product developed does not meet the real needs of the stakeholders, there might be rework, cost increase, dissatisfaction or even cancellation of the project.

1.2 Motivation

Reuse allows the usage of acquired knowledge or parts developed in previous projects, which increases the chances of success in the elaboration of new projects (Palomares et al. 2013). Moreover, it can contribute to increased productivity since it leads to: a) improvement of product time to market; b) improvement of product quality; and c) reduction of development costs (Chernak 2012). Application of reuse approaches from the requirements phase can assist software engineers in the elicitation, validation, and documentation of software requirements, resulting in a more complete, consistent and unambiguous specification (Ketabchi et al. 2011; Palomares et al. 2011).

Software patterns have been used in various areas to capture and reuse the previously acquired knowledge. Their best known uses are architecture, component design, and implementation. In this work, the focus is on the use of patterns in the requirements engineering.

A Software Requirements Pattern (SRP) is an artifact that provides guidance on requirements specification, functional, non-functional, and business rules so that they can be reused in well-defined contexts and problems (Withall 2007).

In software development, there are requirements that are similar in nature or that appear frequently in most softwares, indicating a possible pattern (Withall 2007). Other sources for identifying patterns include knowledge gained from field surveys, individual knowledge of software engineers, standards and best practices or any other software artifacts (Roher and Richardson 2013).

Usually the non-functional requirements can be reused independently of the domain, i.e., they appear in the same way in different projects. On the contrary, the reuse of

functional requirements, in most cases, is only possible for a given software domain (Palomares et al. 2013).

A pattern has a presentation structure that corresponds to formalization and documentation to capture and reuse knowledge. There are several ways to present a pattern. In general, four elements are essential: the name of the pattern (general description of applicability), the problem (what it intends to solve), the solution (description of how to obtain the desired result) and the consequences (implications on the use of the pattern).

The use of patterns helps software engineers in reusing successful solutions to develop new projects (Gamma et al. 1994), eliminating the redundancy of defining a set of requirements for each software project, which in turn reduces the workload and skill required for communication among stakeholders.

1.3 Problem

Due to the evident importance and problems faced in the RE process, there are some research gaps regarding the production of a complete, consistent and unambiguous RD. Reusing requirements with the use of patterns can be a viable alternative to increase the quality of specification requirements in software development.

When the software engineer has a computational support for elicitation of requirements and writing of the RD with the use of patterns, the homogeneity of the information can also be guaranteed, besides the increase in productivity since there is a guide to be followed.

This work aims to present a set of elaborated patterns for the specification of functional requirements and business rules in the domain of Information Systems (IS), as well as a computational support elaborated on the basis of these patterns for the production of an RD. Patterns for non-functional requirements are not within the scope of this project, some of which can be found in Franch et al. (Franch et al. 2010).

1.4 Contributions

The main contributions of this paper are: a) a catalog of patterns for the elicitation and writing of functional requirements and business rules in the IS domain; b) a computational support based on the set of patterns that favors a more complete and consistent writing of the requirements obtained from the stakeholders, generating a standardized RD, with specific statements of the system being elaborated; c) the possibility of writing new patterns from an elaborated structure.

Currently, commercial information processing corresponds to the main area of software application (Rezende 2005), which justifies the selection of the IS domain for the elaboration of patterns proposed in this work. The relevance in the elaboration of these patterns is justified by the similarity found when describing different IS.

1.5 Organization of work

This work is divided into six sections, besides the introduction. In Section 2, the work related to the use of patterns in the requirements specification is presented. In Section 3, considerations are made about concepts in IS. In Section 4, the patterns developed for the IS domain are presented and commented. In Section 5, the

computational support developed for the elaboration of requirements documents is presented. In Section 6, the results of the evaluation of the use of the computational support and elaborated patterns are presented. In Section 7, the conclusions and suggestions for future work are discussed. In Section 8, competing interests are presented. In Section 9, the authors' contributions are presented. In Section 10, the list of abbreviations is presented. In Section 11, the availability of data and materials are presented.

2 Discussion of related work

Withall (2007) presents a catalog with thirty-seven requirements patterns organized in eight domains (fundamental, information, data entity, user function, performance, flexibility, access control and business). Each pattern addresses a specific functionality of these domains, but some patterns share and integrate information. Most patterns meet the specification of non-functional and technical requirements, some of which are also for functional requirements.

Roher and Richardson (2013) suggest the incorporation of sustainability requirement patterns into the requirements engineering process in order to facilitate RD writing for certain types of sustainability requirements. These patterns are not always addressed in IS.

Wei et al. (2013) provide a template in order to write patterns for software security requirements based on the design patterns of Gamma et al. (Gamma et al. 1994) and the requirements patterns of Withall (2007). The pattern “hazard control command”, for example, is applicable to the specification of critical safety requirements at A-C levels, and these levels correspond to the risk severity degree, with level A - catastrophic, B - critical and C - significant.

The study by Li et al. (2012) consists of the presentation of two requirements patterns of typical projects for computational seismology: the simulation pattern and the data access pattern. The authors report that these patterns can benefit the specification of future projects and reduce development costs.

Franch et al. (2013) present the PABRE (Pattern-Based Requirements Elicitation) framework, which is designed to support the reuse of requirements by means of patterns. This framework consists of a metamodel that describes the main concepts using the notion of pattern, a method to conduct the requirements elicitation and documentation process, a catalog of patterns and a tool to support the management and use of the catalog. These patterns primarily serve non-functional and document-management requirements that are outside the scope of this project.

The study by Palomares et al. (2012) presents two patterns of non-technical requirements, elaborated through the PABRE framework. In addition, the authors affirm that these patterns are part of a larger catalog, which currently contains 37 non-technical requirements patterns, 29 non-functional requirements patterns, and 47 functional requirements patterns that apply to the document management domain and which were built from requirements documents of previous projects. These functional requirements patterns are not applicable to the commercial IS domain.

The patterns found in these related papers are mostly for the specification of non-functional requirements, thus allowing the reuse of the pattern in different domains without any change. For functional requirements this does not occur, since there is a

need to specify details for the developing system (Palomares et al. 2013). However, in the IS domain, a similarity in the specification of functional requirements, regardless of the type of system, can be observed. This similarity is found, for example, in data maintenance (CRUD), in management reports, in business transactions, among other functions.

3 Concepts in Information Systems

In IS requirements documents, it is common to find functional requirements that specify operations for data processing, maintenance, and query, such as creating, reading, updating, and deleting information.

Due to the existence of different interests, specialties and levels in an organization, systems are classified into types, the most common of which are: Transaction Processing Systems (TPS), Management Information Systems (MIS) and Decision Support Systems (DSS) (Stair and Reynolds 2005). TPSs are used to record basic transactions of the organization, such as sales orders records, hotel reservations, sales, client record maintenance, etc. MISs use the data stored by the TPSs to provide information to managers and decision makers, e.g., a weekly order report or the list of hotel reservations during a given period. DSSs are used to support decision-making on specific problems by helping to answer questions such as: What is the impact on production scheduling in case sales double in December? DSSs use internal information from TPSs, MISs, and often use information from external sources (Laudon and Laudon 2007).

The functional requirements of these types of systems, in most cases, require the definition of specific business rules. In IS perspective, a business rule is a statement that defines or restricts some business circumstance (Wiegers and Beatty 2013).

Wiegers and Beatty (2013) identify five main types of business rule classification, which present a typical form of writing, as shown below:

- **Facts**: are significant statements about how the business works. A fact describes associations or relationships between important business terms. Example: Every order has a delivery fee.
- **Constraints**: are statements that restrict the actions that the system or its users can perform. Example: A student can only borrow, concurrently, from one to three books.
- **Action enablers**: are rules that trigger some action if specific conditions are true. Example: If the withdrawal of the book does not occur within the stipulated period, then the reservation is canceled.
- **Inferences**: are rules that derive new facts from other facts. Usually written in the "if / then" pattern, however, the "then" clause simply provides a fact or knowledge and not an action to be undertaken. Example: If the user does not give back the loaned book before the deadline, then he becomes a pending user.
- **Computations**: are business rules that define calculations that turn existing data into new data using specific mathematical algorithms or formulas. They can be expressed as mathematical formulas, textual descriptions, tables, etc. Example: A progressive discount applies if more than 10 units are purchased. From 6 to 10 units the discount is 10%. The purchase of 11 or more units has a discount of 20%.

These concepts were used in this paper for the elaboration of business rule patterns. However a new nomenclature has been defined in order to facilitate the use by software engineers during the RD writing.

The following section presents the methodology used for the elaboration of requirements patterns.

4 Method

The methodology used for the elaboration of the patterns consisted of the following steps:

1. **Getting different requirements documents**: were obtained seven fictitious requirements documents from used in academies, such as courseware disciplines in educational institutions (UFSCar, USP and UEMG).
2. **Organization of Requirements**: the requirements of different IS domain documents were organized by type of operation. Examples: data maintenance (inclusion, removal, change, query), transaction processing (purchase, sale, refund, etc.), printing or listing of managerial reports, etc. The requirements that perform the same operation were those selected to become a pattern.
3. **Requirements Analysis**: for each type of operation, the requirements were analyzed to identify whether they were complete and unambiguous, as well as identify the parts of the text that should be fixed and the variable parts.
4. **Pattern Specification**: the applicability and the solution of the pattern were defined to meet the specification of a set of similar requirements (Additional file 1). Moreover, in order to support the software engineer in the use of the pattern, for each template parameter (variable part of the requirement), suggestions of fill-in values were established and the multiplicity, to guide the amount of fill-in values, was indicated (e.g., the multiplicity 1..*, indicates that the parameter may have one or many values).
5. **Establishment of Relationship between Patterns**: definition of the relationship of the pattern in question with other patterns. This relationship can guide and/or alert the software engineer in the use of other patterns that may complement the specification when applying a certain pattern, thus improving completeness in the RD.

To meet the IS domain, a structure for the presentation of patterns was established (Additional file 1) based on the structure of Gamma et al. (1994) and Withall (2007).Some elements have been added such as domain, type, and related patterns.

To meet the requirements of more common systems of the IS domain, a catalog was elaborated with ten functional requirements patterns, divided into four groups of operations:

- **Basic Operations**: the requirements patterns elaborated for these operations are those that describe data manipulation, such as creating, reading, updating, and deleting information. To meet each of these operations, the Include Information, Retrieve Information, Change Information and Delete Information patterns were respectively elaborated;

- **Transaction Processing**: most organizations need systems that perform and record routine transactions for the company's operations, such as processing sales and reservation requests, dealing with payable and receivable accounts, among others (Laudon and Laudon 2007). This type of system, called transaction processing, supports daily activities and helps organizations add value to their products and services (Stair and Reynolds 2005). In order to contribute to the identification and writing of requirements related to the processing of transactions, the pattern Process Transaction was elaborated and is presented in Additional file 2.
- **Management information**: in organizations, administrators and managers need systems that assist in monitoring, control, decision-making and administrative activities. In this sense, Management Information Systems (MIS) provide reports on the performance of the organization (Laudon and Laudon 2007). The main purpose of MIS is to help an organization achieve its goals, using reports to provide detailed managerial information on daily operations of the organization (Stair and Reynolds 2005). In this context, the "Management Query" pattern was elaborated. The management information required by the companies usually refers to the transactions processed or information about basic business records, such as a listing of products, clients, vendors, etc. In both cases, the data is stored in a database.
- **Business rules**: functional requirements often entail the definition of specific business rules. These rules establish restrictions on system operations and may vary in different organizations. As an example, a business rule can establish the policy of sales discount, the commercial conditions for credit sales, the establishment of a minimum value of an order, among others. To comply with the IS domain, four patterns of business rules were elaborated, namely: Execution Condition (Additional file 3), Value Calculation, Required Information and Execution Limit.

Two of the patterns elaborated from the requirements obtained are presented below: Process Transaction (transaction processing) in Additional file 2. and Execution Condition (business rules) in Additional file 3.

In Additional file 4 a summary of the elaborated patterns for the IS domain is presented. The first column contains the type of the pattern, the second one has the name of the pattern, and the consequence is shown in the third column.

No patterns have been generated to meet the requirements for DSSs because of their specificity.

5 Computational Support for the Elaboration of Requirements Documents with the Use of Patterns

Palomares et al. (2014) state that it is difficult to use patterns without computational support to guide the software engineer. In particular, they also report participants' opinions on reuse based on software requirements patterns. One of the evaluated features involved the critical factors for the introduction of a requirements patterns catalog. Two main points were cited: the need for a well-defined method of use and a support tool.

In this project, after creating the patterns of functional requirements and the business rules, considering the observations of Palomares et al. (2014), the process of the

approach and the computational support were also developed to guide the software engineers in the activity of requirements elicitation using the elaborated patterns. It is once again emphasized that non-functional requirements are not part of the scope of this paper.

In Section 5.1, the construction of computational support is discussed. In Section 5.2, the process of using the computational support with the requirements patterns for the elaboration of an RD is presented. In Section 5.3, the functions of the computational support are described. An example of application of computational support is presented in Section 5.4.

5.1 Computational Support Construction

The construction of the computational support used the iterative and incremental process model, allowing the improvement of the usability, of the functionalities as well as the inclusion of new ideas.

The Java language was chosen together with the Java SE (Standard Edition) platform (Java) with the MySQL Server Community database (MySQL), the JDBC (Java Database Connectivity) driver for MySQL library (MySQL et al. 2017) to allow connection to the database and the iText library (iText) for creating an RD in PDF format.

The computational support for the elaboration of an RD in the IS domain has three modules: i) the specification and management of the patterns, usually under the responsibility of a software engineer with more experience; ii) the instantiation of the patterns during the elicitation of requirements; and iii) the basic functions inherent to the maintenance of user (software engineer), client and project.

In Fig. 1 the conceptual model of computational support is presented using an UML class model. The elements of this model are presented in detail in the following topics:

- A pattern (Pattern) may have related patterns (Related Pattern), which contribute to the complete elaboration of the RD. Each pattern may or may not have parameters (Parameter) corresponding to the variable part of the solution proposed by the pattern. If it exists, it must be associated with a type (Type of Parameter) which in turn provides suggestions of filling values for the parameters (Parameter Value Suggestion) to assist the software engineer while writing the variable part of the requirement. Example: The "Process Transaction" pattern has the "Execution Condition" pattern as related. When instantiating the "Process Transaction" pattern, the instantiation of the "Execution Condition" pattern is suggested in order to complement the elaborated requirement specification. Each pattern provides, as a solution, a template that can have parameters. In the case of "Process Transaction", the parameters are: <<transaction>>, <<entity> > and < <attributes>>. These parameters must be filled with values according to the needs of stakeholders. When elaborating the solution of this pattern, these parameters were associated to a type of parameter contained in the repository of the computational support so as to provide filling suggestions. The parameter "transaction", for example, is associated with a type called (transactionEntity) that provides suggestions for values to be filled in such as purchase, sale, rental, etc.

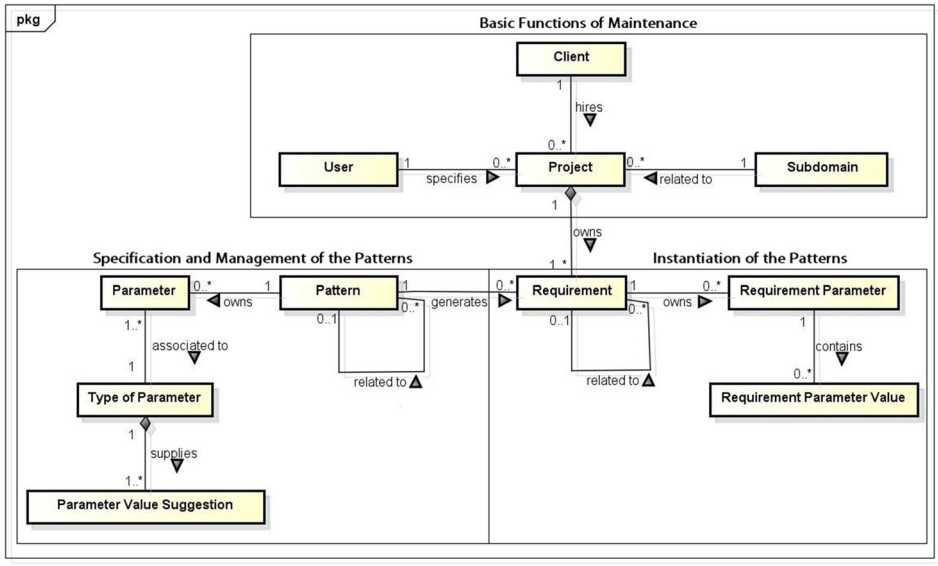

Fig. 1 Conceptual model of computational support

- For the writing of requirements (Requirement), a project must be created (Project), which requires the selection of the client (Client), the user (User), and the subdomain (Subdomain). The client corresponds to the one who hired the project. The user is responsible for the specification of the RD. The subdomain guides the user towards the reuse of specified requirements in future projects. Example: The "Sales Order System" project of client "X", has "Y" as the responsible user (software engineer). This project belongs to the subdomain of "Sale of Products and Services". A project can be specified by more than one user, however, it must have a responsible user.
- By means of the pattern (Pattern) a template, which has a text with fixed and variable parts, is provided for the specification of the requirement (Requirement). The variable part is called the parameter of the requirement (Requirement Parameter), which is used to receive the values that the user has supplied (Requirement Parameter Value). Example: The "Execution Condition" pattern provides the template " The < operation > should only be allowed if < condition > " to guide the software engineer in writing the requirement. After filling in the parameters of the pattern, the requirement is generated.

5.2 Process of Using the Computational Support for the Patterns Instantiation

To use the computational support for the instantiation of the patterns in order to elaborate IS requirements documents, the following steps must be followed:

1. **Selection of the Pattern**: according to the client needs, the software engineer selects the appropriate pattern for this type of problem from the tree structure organized by the type of pattern that is currently represented by functional requirements and business rules Fig. 2. The information about the selected pattern

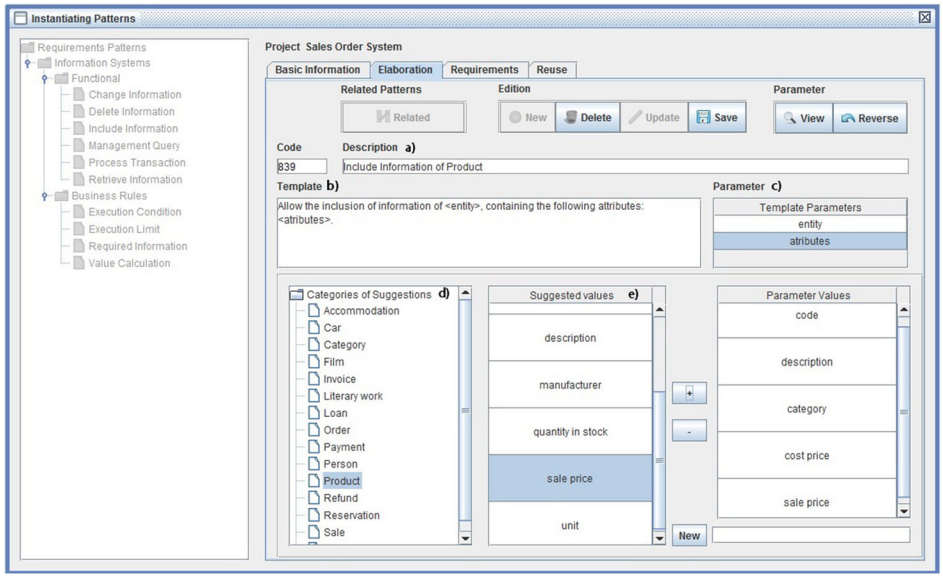

Fig. 2 Pattern instantiation module

is displayed in the "Basic Information" tab. If the pattern meets the requested requirement, the software engineer proceeds to the next step.

2. **Specification of the variable parts**: the software engineer informs the client of the variabilities available for the use of this pattern to obtain the desired solution. The client can choose the options that will satisfy the needs of his system from the set of suggestions registered for the pattern. However, the client may inform what is desired and this can be incorporated as variabilities later on.
3. **Additional specifications**: the software engineer identifies with the client the existence of some business rule and/or some complementary functional requirement in relation to the elaborated requirement. In both cases, a pattern can be used to complement the specification. Thus, we turn to steps 1 and 2 for the specification of the complementary requirements. The software engineer may also propose some complementary requirements through the suggestions of related patterns provided by the computational support.

If none of the patterns available in the computational support meet the client's need, there is also the possibility of writing a requirement without the use of a pattern. To do this, it is necessary to use the [New] button of the "Elaboration" tab that will leave the Template field (item b. Fig. 2) blank to write the requirement.

In Fig. 3, the process of using the computational support for the patterns instantiation is presented.

5.3 Implemented Functions

With the module of specification and management of patterns (Fig. 4), it is possible to do maintenance on the patterns. For the specification of a pattern, it is necessary to fill in the elements: name, domain, purpose, problem, consequence, type and solution. In the definition of the solution, for each parameter of the template there is: a) a type,

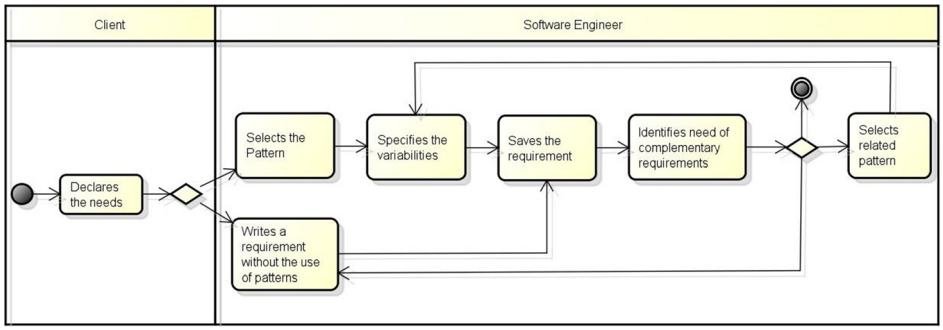

Fig. 3 Process of using the computational support for the patterns instantiation

which is associated to a set of fill-in suggestions that can be used in the pattern instantiation. This set of fill-in suggestions was obtained from the requirements documents used for the elaboration of the patterns; b) a description that guides the correct completion of the parameter; c) the multiplicity that guides the amount of allowed values for the parameter.

To specify a pattern, the "Basic Information" and "Related Patterns" tabs are used. The patterns related to the pattern in question are described by means of the tab with the respective name.

The pattern instantiation module (Fig. 2) has four tabs. These are presented in detail in the following topics:

- **Basic information**: presents the applicability and solution information of the pattern.

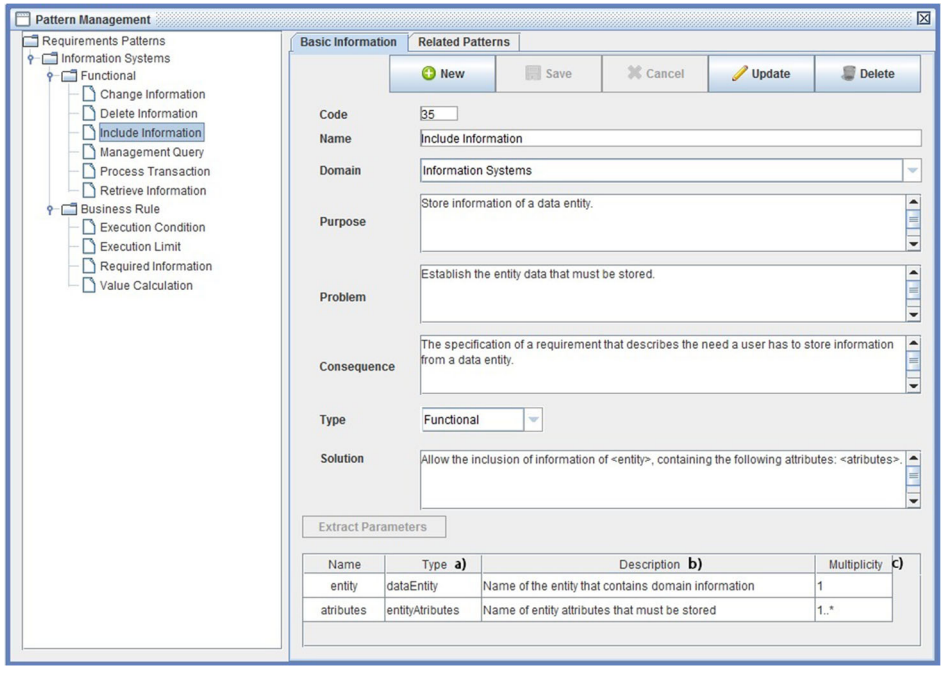

Fig. 4 Pattern specification and management Module

- **Elaboration**: used for the specification of a requirement by means of the chosen pattern. The letters placed in Fig. 2 are intended to guide the reader, so they are the same as those used below. In this tab there is: a) a field to describe the requirement; b) the solution template; c) a list with parameters; d) a list of suggestions organized into categories of suggestions (per parameter); e) when you select the category of suggestions, a list of values is displayed. In the example in Fig. 2, the "Include Information" pattern is being instantiated for writing a requirement that describes the need of store product information.
- **Requirements**: presents all project requirements; allows selection of the requirement for editing or deleting; allows the generation of an RD in PDF format; allows the establishment of relationship between the requirements; and provides suggestions of patterns related to the instantiated requirement.
- **Reuse**: presents the requirements already instantiated in other projects, which were elaborated by the selected pattern. These requirements can be reused in the current project instead of being specified by using a pattern. The requirements generated in concluded projects are stored in a local repository that allows the reuse of requirements in future projects, with the possibility of changing those requirements if necessary.

5.4 Exemplifying the Writing of an RD by means of Instantiating Patterns

To exemplify the use of computational support, the specification of a Sales Order system will be used. The System Overview is displayed in Additional file 5.

The elaboration of the RD through the computational support begins with the registration of the project. For this you need to provide: a name for the project, its overview (it will be part of the RD), the selection of the responsible client and software engineer (previously registered), the project start date, the status (initially it must be "open" to indicate that the project is in progress) and the subdomain selection, to guide the software engineer in future reuse of requirements. In Fig. 5, the project registration interface populated with the Sales Order System data is shown.

After registering the project, through the pattern instantiation module, it is necessary to select the pattern according to the needs declared by the stakeholders. The information of the selected pattern is displayed in the "Basic Information" tab as can be seen in Fig. 6.

According to the system overview, it is indispensable to store clients in the database. Therefore, the “Include Information” pattern must be instantiated using the [Instantiate] button.

In Fig. 7, the "Elaboration" tab used to write a requirement for client inclusion is shown. To do so, one must: a) define a description for the requirement, to facilitate its location; b) fill in the parameters with the information provided by the stakeholders; c) select the parameter and choose one of two options: manually write the corresponding data which is done by clicking the [New] button (c1), or select the data from the suggestions provided by the computational support repository (c2). In this example, the < attributes > parameter is populated with values that match the attributes of the client. Selecting this parameter provides some categories of suggestions, such as sales, product, person, etc. When one of these suggestions is selected, others are presented

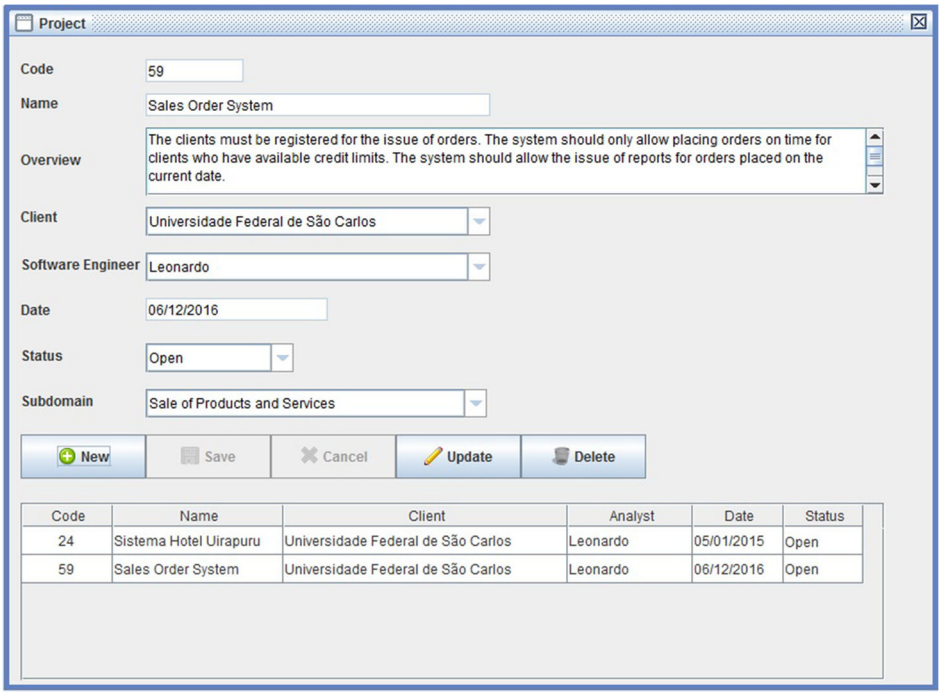

Fig. 5 Project maintenance module

for filling the parameter. After these selections, the [+] button must be pressed so that the selected values are assigned to the parameter; d) with the selection of the [View] button the substitution of the parameter in the template occurs with the data assigned to the parameter. This step is not required and replacement will occur when the [Save] button is pressed to wrap up the writing of the requirement; e) finally, press the [Save] button to finish writing the requirement.

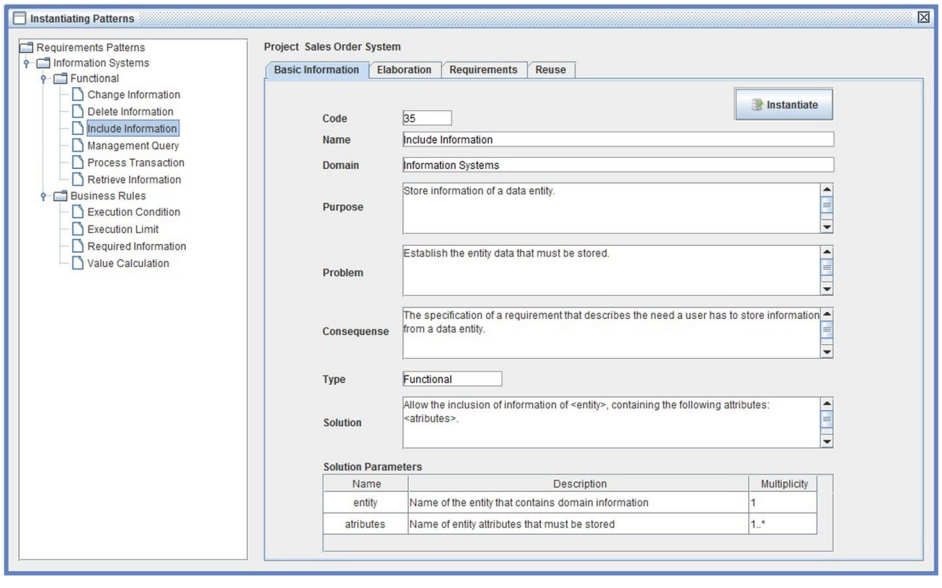

Fig. 6 Instantiation of include information pattern

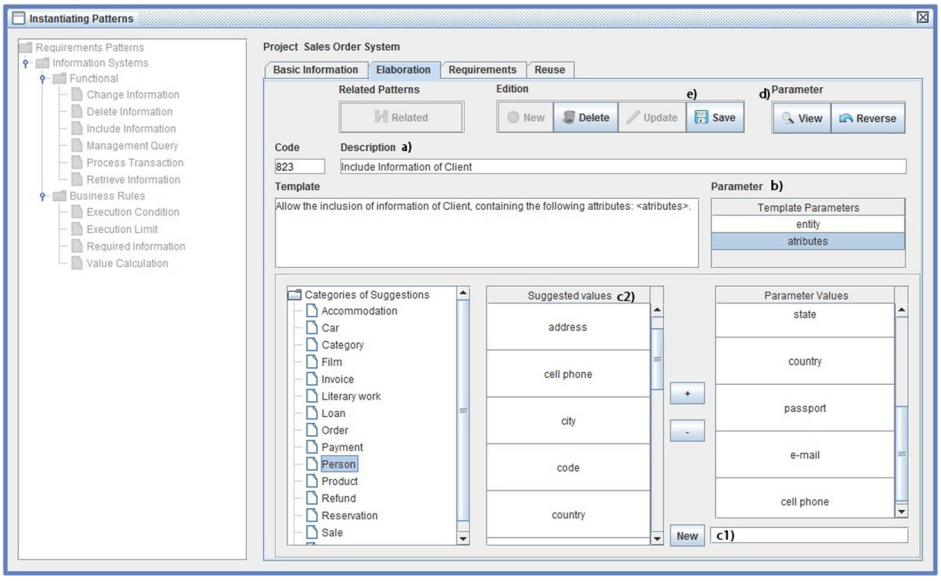

Fig. 7 Writing the requirement for client registration

In the example, it is assumed that a sales order must be processed, i.e., the registration of the sales order with all relevant information must be carried out. To meet this functionality the "Process Transaction" pattern should be used. Its instantiation occurs in the same way as the "Include Information" pattern, described above.

After saving the requirement, the computational support may suggest related patterns to complement the specification of the requirement. This is done using the [Related] button which displays a number that corresponds to the number of patterns related to the one instantiated. In Fig. 8, there are five patterns related to the pattern "Process Transaction". In this case, the suggested patterns are business rules commonly required when specifying a transaction, which establish for example: a) conditions or restrictions for the execution of an operation; b) the description of a mathematical calculation required by an operation and the condition for performing the calculation; c) a description of the information required in a given operation; and d) the limits on the execution of an operation. These patterns can guide the software engineer to complete the requirement specification. The instantiation of these patterns also occurs in the "Elaboration" tab.

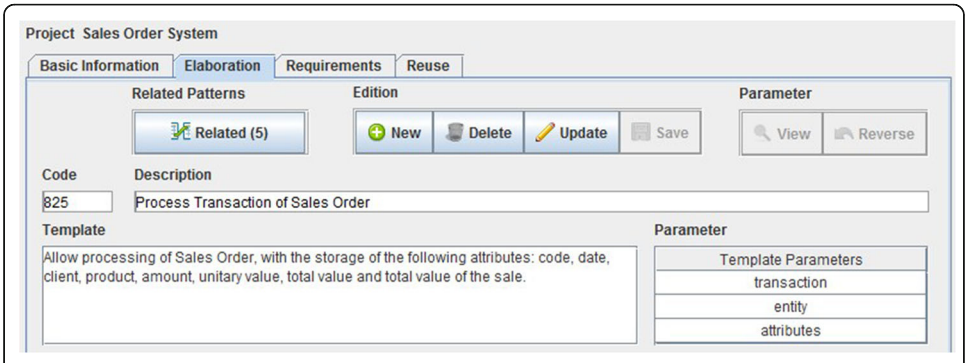

Fig. 8 Patterns related to the pattern process transaction

To meet, for example, the business rule that allows you to place orders on time only for clients that have available credit limits, the pattern “Process Transaction” has the related pattern “Execution Condition” that must be used for this purpose. Finally, to meet the requirement to issue a report, the “Management Query” pattern must be instantiated.

In the Pattern Instantiation module, the "Requirements" tab (Fig. 9) presents the following items: a) all project requirements. When selecting a requirement by means of a click, requirements related to it are also presented, if they exist. In the example, the “Process Transaction of Sales Order” requirement has a dependency relationship with the “Execution Condition of the Order” requirement; b) allow the editing, visualization or exclusion of the selected requirement, as well as provide suggestions of patterns related to the selected requirement, as previously presented; c) allow the establishment of relationship between the requirements. The relationship between the requirements elaborated by the related patterns occurs automatically, but if the establishment of relationships between other requirements is necessary, this can be done manually. To exemplify the manual relationship between requirements, a dependency relationship was established for the requirement "Process Transaction of Sales Order" with the requirement "Include Information of Client", this relationship can be observed in the RD shown in Fig. 10; d) allow the generation of the document with all the instantiated requirements, in PDF format, as shown in Fig. 10.

The resulting RD after the use of computational support is shown in Fig. 10. Functional requirements are identified by the abbreviation "FR" and the business rules by the abbreviation "BR". The "Depends on" column shows the dependency between the requirements. In the example in question, the FR2 requirement depends on the FR1 requirement.

The "Reuse" tab presents the requirements of the repository that were instantiated in other projects, referring to the selected pattern. In the example that is shown in Fig. 11, as the “Include Information” pattern is selected, the suggestions for requirements that

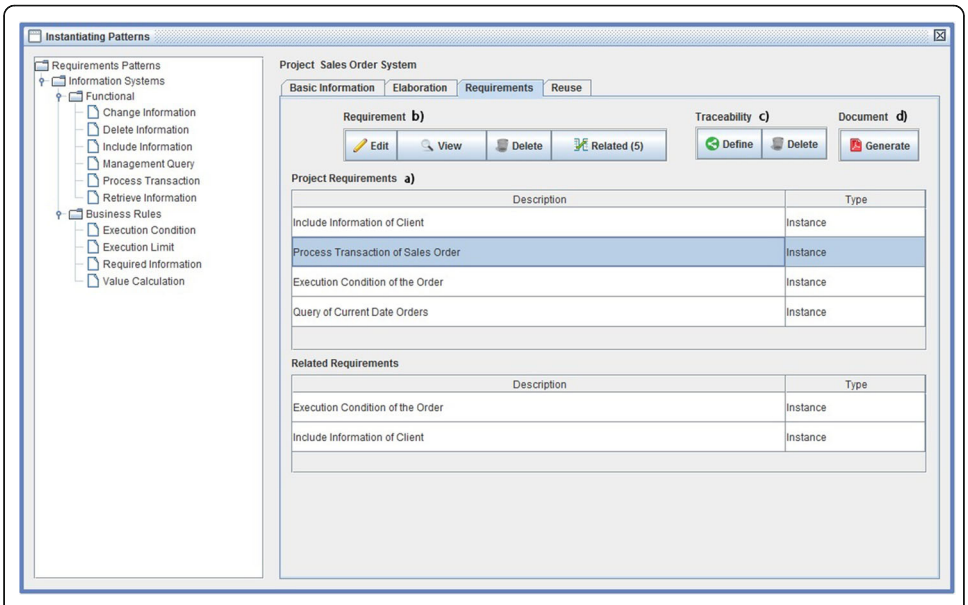

Fig. 9 Project requirement list

Sales Order System

Requirements Document

Overview

The clients must be registered for the issue of orders. The system should only allow placing orders on time for clients who have available credit limits. The system should allow the issue of reports for orders placed on the current date.

Functional Requirements

Identifier:	Description	Depends on:
FR1	Allow the inclusion of information of Client, containing the following attributes: code, name, address, city, state, country, passport, e-mail and cell phone.	
FR2	Allow processing of Sales Order, with the storage of the following attributes: code, date, client, product, amount, unitary value, total value and total value of the sale. BR2.1. The Order should only be allowed if the customer has an available credit limit.	FR1.
FR3	Allow printing of a report of orders placed on the current date, containing the following attributes: client code, name of client, social security and total value of order.	

Fig. 10 Sample requirements document

have been instantiated with this pattern in "reservation of goods and services" and "restaurant" projects are being presented. These requirements can be fully reused in the current project or have the possibility of changes. There are two possibilities for reuse: simply reuse the requirement by selecting the [Reuse] button or reuse with dependent requirements (related) by selecting the [Reuse with Dependents] button. Before reusing a requirement, the software engineer can view its specification by selecting the [View] button.

This work differs from the others presented in Section 2, since: a) it allows the writing of an RD with functional requirements and business rules for specific systems in the IS domain. For example, the pattern that specifies a new registration of an entity can be used for the elaboration of a hotel system, a pharmacy system or any other, and

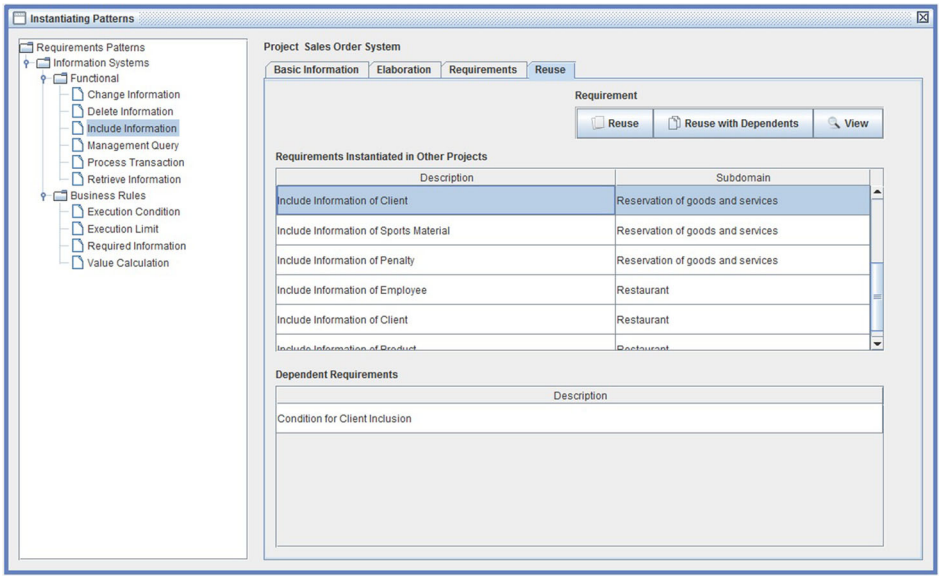

Fig. 11 Reusing repository requirements

it will have the complete specification of that requirement in the RD; b) with the use of computational support the RD is elaborated in PDF format; c) the software engineer can reuse requirements specifications built on previous systems modifying some attributes or not. For example, a specification for the reservation in a hotel, can be retrieved from the repository and reused in another hotel system. There is the possibility of reusing this function in any other system, for example, a reservation system of books, cars, products, etc.; d) new requirements patterns can be elaborated using the presentation structure.

6 Results and discussion

To evaluate the proposed approach as well as the completeness of the patterns and usability of the computational support, case studies were carried out with undergraduate students from the bachelor's courses in Computer Science and Electrical Engineering at UFSCar, São Carlos/SP and from the Information Systems course at UEMG, Frutal/MG unit.

6.1 Case study 1

forty-two students divided into eight groups. Firstly, the students elaborated a document of ad hoc requirements, related to some services offered by UFSCar such as: library, university restaurant, field rental, registration in discipline, etc., involving few requirements. This stage occurred after the students were presented with the concepts of requirements and techniques to obtain them, which aimed the elaboration of an RD.

The software requirements patterns and computational support were presented to students, besides having training sessions using a sample system with requirements different from those developed by the students.

Next, the elaboration of the RD of the system specified of ad hoc mode was requested to each group, using the tool. After this activity, the students answered a questionnaire in order to identify the percentage of requirements specified with the use of the computational support, if the RD became more complete and if the computational support was easy to use. The result was: on average, 93% of requirements were specified. Because of the lack of in-depth knowledge of existing patterns, some requirements were not specified using them; 87% of respondents stated that the RD became more complete, with new requirements and business rules; and 75% evaluated the computational support as easy to use.

Regarding the completeness of the RD, the factors evidenced by the students were: a) the possibility of defining dependency relations between the requirements and business rules; b) the creation of one requirement suggests the creation of another; c) the identification of requirements that had not been identified in the first RD due to the patterns provided; and d) the patterns supplied a more complete description for the requirements.

Regarding usability, several suggestions were made to improve the computational support, such as: a) increasing the number of characters in the project overview field; b) updating the parameters with the values provided by the software engineer after their completion (in the first version the update occurred only when saving the requirement); c) presenting the suggestions of parameter values in alphabetical

order; d) improving the dimensions of the frames and windows; e) making it possible to view a requirement of the project under development through the "Requirements" tab. At first it was only possible to visualize the requirements through the elaborated RD.

All of these suggestions were implemented in the computational support before doing Case study 2.

6.2 Case study 2

This one was carried out in two disciplines of UFSCar, one being: i) composed of computer science students with more advanced knowledge about requirements and development of systems for mobile devices; and ii) with electrical engineering students with initial knowledge about requirements, who have developed simple systems. Students were divided into groups and each one developed systems in the IS domain. In this study, the RDs were elaborated after the knowledge and training with the requirements patterns and computational support. The students did not previously write an RD in ad hoc mode.

This case study, besides having the objective of identifying the comprehensiveness of the patterns for the elaboration of an RD and the usability of the computational support, also aimed to understand the students' perception regarding the proposed reuse approach for the elaboration of an RD (by means of patterns and requirements contained in the repository).

The results obtained were as follows:

i). Systems developed for mobile devices; 15 groups with four students in each group. In relation to the students' perception about the reuse in the elaboration of RDs: 46% of the groups noted that it is possible to completely use patterns and reuse requirements in the elaboration of RDs and 26% mentioned that the reuse is partial. Regarding the comprehensiveness of the patterns: 20% of groups specified more than 90% of requirements; 27% specified 50% to 90%. In addition, 80% agreed that it is easy to use computational support.

ii). Simple information systems; 7 groups with a total of 24 students. In relation to the students' perception about the reuse in the elaboration of RDs: 70% of the groups noted that it is possible to completely use patterns and reuse requirements in the elaboration of RDs and 10% mentioned that the reuse is partial. Regarding the comprehensiveness of the patterns: 20% of groups specified more than 90% of requirements; 30% specified 50% to 90%. In addition, 60% agreed that it is easy to use computational support.

Regarding the usability of the computational support, the students were asked to describe possible improvements. The responses of this questionnaire contributed to the identification of the following improvements in computational support: a) the possibility of reusing specific details of a requirement, in the instantiation of related patterns. For example: the values used to fill a requirement instantiated by the pattern "Include Information", are suggested when instantiating a related pattern, facilitating the filling; b) one of the groups stated the importance of creating models (new patterns) for other types

of requirements. Regarding this item, it is important to highlight that, in the case studies, the students only had access to the instantiation module of the requirements patterns for the elaboration of an RD, not having access to the management module and specification of patterns. This request was due to the need to elaborate non-functional requirements patterns that are very common in mobile systems and are not yet available in the computational support; c) one of the operations was not performing the expected function when clicking the mouse, it took two clicks; and d) the need to make the interface more intuitive.

The identified improvements were implemented after the completion of the case study, in order to carry out a new case study and measure the results again.

6.3 Case study 3

Thirty-four students from the Information Systems course at the UEMG, divided into thirteen groups, participated. The projects were related to real systems developed in companies where the students work. As in case study 1, the RDs were first developed in ad hoc mode and after the presentation of the patterns and the computational support, the RDs were redone.

In this case study, the groups were also asked, through a questionnaire, about their opinion on the productivity of the computational support.

The results obtained were: In relation to the students' perception about the reuse in the elaboration of RDs: 54% of the groups noted that it is possible to completely use patterns and reuse requirements in the elaboration of RDs and 38% mentioned that the reuse is partial. Regarding the comprehensiveness of the patterns: 38% of groups specified more than 90% of requirements; 46% specified 50% to 90%. In addition, 61% agreed that it is easy to use computational support.

Regarding productivity, students answered based on their personal experiences if there is a reduction in time when using computational support for elaborating RDs. The results obtained were: 8% of the groups considered that the computational support improves productivity between 21% and 50% (reduction of time by up to 50%), 46% consider that improvement to be between 51% and 89% (reduction of time by up to 89%) and 38% more than 90% (reduction of time by up to 90%).

The students described the following facts about productivity: a) there is a reduction in the time spent in drafting an RD; b) it provides solution to a problem, with the need to stipulate only the specific details; c) the possibility of reusing already defined requirements makes the task simpler and more dynamic; d) it improves the understanding of the requirements; and e) one of the groups stated the improvement in productivity, however, realized the need for non-functional requirements patterns.

Tables 1, 2 and 3 summarize the results obtained in the case studies.

In carrying out the case studies, in addition to the guidelines for using computer support and the exercise of specifying a simple system, the students also had an sample project in the repository of the computational support in the hotel domain, containing 25 requirements that could be reused. However, it is worth mentioning that the projects developed by the students are different from the project contained in the repository.

Table 1 Summary of Case Study 1

Number of participants	Requirements specified with the patterns	Respondents that considered the RD more complete	Respondents that considered the computational support easy to use
42	93%	87%	75%

There are indications, through the results obtained after the accomplishment of the case studies and with the answers obtained in the questionnaires, that the elaborated patterns helped to make a more complete RD. Other studies should be performed, especially with real systems in companies, to obtain more conclusive results. Due to space limitations, the questionnaire applied is not presented in this paper.

Possible threats to the validity of the proposal have been identified and are presented below:

- The students' lack of experience in RD elaboration and also in the use of patterns resulted in the partial specification of various requirements;
- There is a need to insert patterns that meet the non-functional requirements in the computational support so that the validity of the proposal reaches more expressive indexes;
- The writing of the requirements document, for the second time, with the computational support, was faster due to the previous knowledge of the requirements by the students. This fact influenced the time reduction that was obtained;
- The fact that the participants are of undergraduate disciplines, in which there is a program to be fulfilled, caused time limitation for the accomplishment of these case studies, which interfered in the results obtained.

7 Conclusions

This paper presented an approach for writing requirements documents in the IS domain based on requirements patterns. This approach was developed based on the reuse of knowledge of IS domain projects. The elaborate patterns mainly help less experienced software engineers, providing a more complete and consistent basis for RD writing. However, it is worth emphasizing that due to the complexity of the real world it is impossible to assure the completeness of the needs in an RD.

To facilitate the use of the patterns, a computational support was created in a Java platform with the MySQL database, that guides the software engineer in the use of patterns and in the construction of an RD. The conceptual model used to implement computational support is presented in Section 5.1. Each element of the model was

Table 2 Summary of Case Study 2

Group	Number of participants	Requirements specified with the patterns		Respondents that considered the computational support easy to use	Respondents that considered the possibility of using patterns and reusing requirements in the elaboration of RD	
		> = 90%	Between > = 50% and < 90%		Completely	Partial
i	60	20%	27%	80%	46%	26%
ii	24	20%	30%	60%	70%	10%

Table 3 Summary of Case Study 3

Number of participants	Requirements specified with the patterns		Respondents that considered the computational support easy to use	Respondents that considered the possibility of using patterns and reusing requirements in the elaboration of RD		Respondents' opinions on the productivity of the computational support		
	> = 90%	Between > = 50% and <90%		Completely	Partial	Between > = 21% and < =50%	Between > =51% and < =89%	Between > = 90%
34	38%	46%	61%	54%	38%	8%	46%	38%

detailed and exemplified for a better understanding of the concepts related to the use of requirements patterns described in this paper.

In order to facilitate the use of computational support by the scientific community of software engineering, the elaboration of an RD by means of the computational support was exemplified, detailing the use of its functionalities.

As mentioned in Section 6, the results obtained with the accomplishment of case studies with students of undergraduate courses in two public institutions, evidenced the ease of use of the computational support. It may also be noted that the elaborated patterns collaborated for the completeness of RDs, as related patterns are suggested during the writing of the requirements. In this way, when a less experienced software engineer is writing a requirement, the creation of another requirement is suggested. There is also the possibility of reusing requirements from previously finished projects, therefore increasing the chances of producing a more complete RD.

The case studies also resulted in improvements in computational support in terms of usability and functionalities, which were implemented after each case study. In addition, in the last case study, after the implementation of all improvements, students were asked to present their considerations about the perception of productivity of the computational support. With the analysis of the results, there is evident confirmation of increased productivity.

Regarding future works, new patterns can be elaborated and inserted into the computational support. With the realization of other case studies, improvements can be identified regarding functionality and usability of the computational support. An expected improvement is the extension of the pattern instantiation module, allowing the study of the incorporation of new values for the set of values suggested for the parameters, based on the new values provided by the stakeholders. Another extension is the development of patterns that meet non-functional requirements. More in-depth studies should be carried out so that non-functional requirements already specified by other authors can be reused. The inclusion of these non-functional requirements must occur in the computational support as well as the realization of controlled experiments in order to verify the effectiveness of the proposal. Another intention is to implement a module for the management of modifications of RDs and yet another for the realization of cost and time estimation for the development of the specified projects.

8 Additional files

Additional file 1: Frame 1. Structure Adopted for Presentation of Patterns. (PDF 159 kb)
Additional file 2: Frame 2. Functional Requirement Pattern: Process Transaction. (PDF 175 kb)
Additional file 3: Frame 3. Business Rule Pattern: Execution Condition. (PDF 154 kb)
Additional file 4: Frame 4. Pattern Catalog for the Information Systems Domain. (PDF 211 kb)
Additional file 5: Frame 5. Sales Order System Overview. (PDF 100 kb)

Abbreviations

DSS: Decision Support Systems; IS: Information Systems; JDBC: Java Database Connectivity; MIS: Management Information Systems; PDF: Portable Document Format; RD: Requirements Document; RE: Requirements Engineering; SRP: Software Requirements Pattern; TPS: Transaction Processing Systems

Funding

No funding was received.

Availability of data and materials
The computational support, as well as the elaborated patterns and questionnaires used in the case studies are available at http://advanse.dc.ufscar.br/index.php/tools/erdsi, the use is free. The non-functional requirements are not yet available in the support provided.

Authors' contributions
LVB and RDP participated in the conception of the approach and in the related work analysis. LVB implemented the approach and ran the case studies. RDP helped in analyzing the case studies results and writing and polishing the paper. This work was conducted while LVB did his master's degree in Computer Science at UFSCar. Both authors reviewed the paper and agree with this submitted version.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Department of Exact and Earth Sciences, Universidade do Estado de Minas Gerais, UEMG - Unit FrutalMG, Frutal, Brazil.
[2]Department of Computing, Universidade Federal de São Carlos, UFSCar, São Carlos, SP, Brazil.

Received: 15 December 2016 Accepted: 16 May 2017
Published online: 30 May 2017

References

Boehm B. and Basili V. R. (2001). Software Defect Reduction Top 10 List. IEEE Computer, vol. 34, nº 1, pp. 135-137.
Chernak Y (2012) Requirements Reuse: The State Of The Practice, IEEE International Conference on Software Science, Technology and Engineering
Fatwanto A (2013), "Software Requirements Specification Analysis Using Natural Language Processing Technique," Int Conf Qual Res, pp. 105–110.
Franch X, Palomares C, Quer C, Renault S, Lazzer F (2010). A Metamodel for Software Requirement Patterns. In: Requirements Engineering: Foundation for Software Quality (REFSQ), vol 6182. Springer, Berlin, Heidelberg.
Franch X, Quer C, Renault S, Guerlain C, Palomares C (2013). Constructing and Using Software Requirement Patterns. Book Title: Managing Requirements Knowledge, Springer Berlin Heidelberg, pp. 95–116.
Gamma E, Helm R, Johnson R, Vlissides J (1994) Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition: Addison-Wesley
iText, "iText Developers," [Online] (2017). Available: http://developers.itextpdf.com/downloads. Accessed 24 May 2017.
Java. [Online] (2017). Available: http://www.java.com/. Accessed 24 May 2017.
Ketabchi S, Sani NK, Liu K (2011) A Norm-Based Approach towards Requirements Patterns, IEEE Annual Computer Software and Applications Conference
Laudon KC, Laudon JP (2007) Management Information Systems. Pearson Prentice Hall, 9th edn
Li Y, Pelties C, Käser M, Nararan N (2012) Requirements patterns for seismology software applications, IEEE International Workshop on Requirements Patterns
MySQL, "MySQL Documentation: MySQL Reference Manuals," [Online] (2017). Available: http://dev.mysql.com/doc/. Accessed 24 May 2017.
MySQL, "JDBC driver for MySQL," [Online] (2017). Available: https://dev.mysql.com/downloads/connector/j/. Accessed 24 May 2017.
Niazi M, Shastry S (2003). Role of requirements engineering in software development process: an empirical study. 7th International Multi Topic Conference (INMIC).
Paldês R. A.,Calazans A. T. S., Mariano A. M. , Castro E. J. R. and Silva B. S., "A utilização da linguagem natural na especificação de requisitos: um estudo por meio das equações estruturais," XIX Ibero-American Conference on Software Engineering - Workshop on Requirements Engineering (WER), pp. 365–378, 27–29 Apr 2016.
Palomares C, Quer C, Franch X (2011) PABRE-Man: Management of a requirement patterns catalogue, IEEE Software Engineering Conference
Palomares C, Quer C, Franch X, Guerlain C, Renault S (2012) A catalogue of non-technical Requirement Patterns, IEEE International Workshop on Requirements Patterns
Palomares C, Quer C, Franch X, Renault S, Guerlain C (2013) A Catalogue of Functional Software Requirement Patterns for the Domain of Content Management Systems, ACM Symposium on Applied Computing
Palomares C, Franch X, Quer C (2014). Requirements Reuse and Patterns: A Survey. In Requirements Engineering: Foundation for Software Quality (REFSQ), vol 8396. Springer, Cham
Pressman RS (2009) Software Engineering: A Practitioner's Approach, 7th Edition, McGraw-Hill
Rezende DA (2005) Engenharia de Software e Sistemas de Informação. Brasport, Rio de Janeiro
Roher K, Richardson D (2013) Sustainability requirement patterns, IEEE International Workshop on Requirements Patterns
Sommerville I (2011) Software Engineering, 9th Edition., Pearson
Stair RM, Reynolds GW (2005), Principles of Information Systems. 7th Edition: Thomson, Boston, Mass.
Wei C, Xiaohong B, Xuefei L (2013) A Study on Airborne Software Safety Requirements Patterns, IEEE International Conference on Software Security and Reliability Companion
Wiegers K, Beatty J (2013) Software Requirements. Microsoft Press, Redmond
Withall S (2007), Software Requirement Patterns. Microsoft Press, Redmond.