A REUSE-BASED REQUIREMENTS ELICITATION PROCESS

Sangim Ahn, Kiwon Chong
Department of Computer, Soongsil University, Sangdo-Dong, Dongjak-Ku, Seoul, Korea

Keywords: Reuse-based, Requirements elicitation process, Gap Analysis.

Abstract: Establishing good requirements is important in an initial phase of software development not to make over time and cost of projects and low quality of software products. In the context of Requirements Engineering (RE), reuse is effective in particular because it can help to define requirement explicitly and to anticipate requirement change. We propose a reuse-based process approach to elicit potential requirements from various stakeholders. To achieve our goal, we present (1) analyzing gaps between requirements maps of collected and reused in the repository, and (2) potential requirements elicitation process with these maps. The former is composed of classifying styles of requirements, requirements representation formalism, and gap analysis using generic gap types. The latter is sequential procedure to look for potential requirements in addition to Plus Minus Interests(PMI) method. We illustrate our approach through a credit system case study.

1 INTRODUCTION

There are many reasons for project failure. The main ones related to requirements are: (1) Most of users don't define correctly what they want. (2) Requirements are changing frequently (3) There is no enough time to analyze user requirements (4) Developers don't understand perfectly user requirements. These reasons make over time and cost of projects, and low quality of software products. Therefore, establishing good requirements is important in initial software development. The introduction of reuse in software development aims reducing the maintenance and development costs, reducing the deadlines and improving quality. The reuse was confined at the level of developers, who used libraries of reusable programs. Today, more and more works try to integrate it throughout the whole cycle of production, from the phase of requirements expression to the phase of maintenance. In the context of Requirements Engineering (RE), reuse is effective in particular because it can help to define requirement explicitly and to anticipate requirements change(Ian, 2002)(Lauesen, 2002)(Ounsa, 2001).

We propose a reuse-based process approach to elicit potential requirements from various stakeholders. A map is a requirements representation formalization that allows the representation of a set of functional or non-functional requirements. A collected requirements map is made based on a preliminary method such as user interview, workshops, and observing user work. Whereas, A reused requirements map is stored in the repository. In this paper, we suppose that the repository is already installed and a lot of assets including valuable requirements of many successful projects are accumulated in the repository. To achieve our goal, we present (1) analyzing gaps between requirements maps of collected and reused in the repository (2) potential requirements elicitation process with these maps. The former is composed of classifying styles of requirements, requirements representation formalism, and gap analysis using generic gap types. The latter is sequential procedures to look for potential requirements in addition to Plus minus interests(PMI) method. We illustrate our approach through the finance system case study.

The rest of the paper is organized as follows: In Section 2, we describe related works with requirement reuse and a meta-model for generic gap typology. In Section 3, we discuss gap analysis using collected and reused requirements maps in details. In Section 4, we discuss potential requirements elicitation process. In Section 5, we illustrate our approach through the finance system case study. Finally, in Section 6, we draw some conclusions and discuss research issues for future work.

Ahn S. and Chong K. (2006).
A REUSE-BASED REQUIREMENTS ELICITATION PROCESS.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - ISAS*, pages 403-409
DOI: 10.5220/0002449604030409

2 RELATED WORKS

2.1 Approaches to Requirements Reuse

Informal requirement specification is commonly used in the early phases of software development. Such documents are usually produced in natural language. In spite of many problems in their handling, they are still regarded as one of the most important communication mechanism between developers and users. The lack of formality, structure and ambiguity of natural language makes requirements documents difficult to represent, process, and reuse. To overcome these problems, firstly, requirements statements need to be formalized to accommodate reuse tasks. Second, a repository of reusable requirements artifacts is made to analyze related assets. Third, automated tools are developed to use in conveniences. Several methods, techniques, tools and methodologies were suggested as useful in supporting these tasks(Jacob, 2000).

Table 1 shows many approaches to requirements reuse, such as text processing, knowledge management and process improvement. The text processing approach focuses on the text of requirements, its parsing, indexing, access and navigation. They rely heavily on the natural language grammars and lexicons, statistical text analyzers, and hypertext. The knowledge management approach aims at elicitation, representation and use of knowledge contained in requirements documents, and reasoning about this captured knowledge. These methods commonly focus on the modeling of a problem domain, and they utilize knowledge acquisition techniques and elaborate modeling methods. Sometimes they also utilize knowledge-based systems and inference engines. The process improvement approach aims at changing development practices to embrace reuse.

Our approach concentrates on taxonomic representation, reuse-based process, meta-model to create a repository and analyze gaps in three above-mentioned aspects of requirements handling.

2.2 A Meta-Model for a Gap Typology

The major point of our approach is to model a gap between requirements specification of collected and reused in repository. Intuitively a gap expresses a difference between collected requirements map and reused requirements map such as the deletion or addition of a collected requirement's element in reused requirements. In functional requirements domain, gaps are related to actors, use cases, relationships which transform elements of maps. In order to facilitate gap analysis, we need to define a gap typology and a set of gap types related to elements in maps.

A number of attempts have been made to make explicit the elements that compose any model to define meta-models(Colette, 2004)(IRDS,1990) (Marttiin, 1994)(Plihon, 1997)(Prakash, 1999). There are different meta-models depending on the meta-modelling purpose. (IRDS,1990) is a standard to facilitate the evolution of model representation in CASE tools, (Prakash, 1999) aims at a formal definition of a method and (Marttiin, 1994) searches for a generic repository structure of meta-Case environments. (Colette, 2004) is targeted to the identification of key significant transformations that can occur in a model.

We adopt the meta-model similar to Collett's approach. This meta-model is drawn in Figure 1 using UML notations. This figure shows that any model is made of elements, every element having a Name, and is characterized by a set of Property. In the meta-model, there are two orthogonal classifications of Elements. The first classification makes the distinction among Use cases, Actor, Relationship Elements. Use cases elements are decomposable into fine-grained ones that can be simple or Use cases elements whereas Actor and Relationship Elements are not decomposable into other Elements. The second classification is a

Table 1: Various Approaches to Requirement Reuse.

Type	Text	Knowledge	Process
APPROACH	parsing specifications natural languages processing use of hypertext finding repeated phrases assessment of similarity	taxonomic representation logic-based specifications analogical reasoning knowledge-based systems analysis patterns domain mapping domain analysis	reuse-based process meta- and working models wide-spectrum reusability family of requirements reuse-based maintenance CASE-support of early reuse

partition of elements into Link and NotLink. An element of the type Link is a connector between two elements, one being the Source and the other the Target. Elements, which are not links, are referred to as NotLink. An element is-a another element, might inherit from another element.

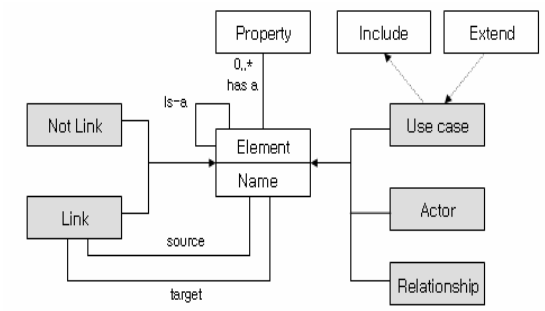

Figure 1: The meta-model for gap typology.

3 REUSE-BASED MAPS

3.1 Styles of Requirements

In requirements specification, a requirements item may express many types of information. This information should be divided to analyze requirements meaning and to reuse for next projects. We classify requirements into four styles such as data, functional, quality, and managerial requirements (Lauesen, 2002).

❑ Data requirements styles are related to database and input/output formats. The system has to store the corresponding data in some kind of database or other internal objects. Database data is independent of the interface. However, input and output data appear on the various interfaces. The data requirement should in principle specify the detailed data formats for each interface.

❑ Functional requirements styles are related to the functions of system. The function may present how it records, computes, transforms, and transmits data. Traditionally, function means that for any given input and any given system state, it will deliver some output and set the system state to something new. In practice, when we give the system a command, the response may be some visible output, some invisible change in database contents, and other variables.

❑ Quality requirement styles are related to non-functional requirements such as performance, usability, and maintenance. Performance means how efficiently the system should work with the hardware corresponding response time, accurate results and stored data amount. Usability means how efficiently the system should work with the user corresponding easy learning. Maintenance means how easy the system should be to repair defects and add new functionality.

❑ Managerial requirements styles are related to delivery time, legal responsibility, and penalties corresponding contractual issue.

3.2 Generating Maps

From the viewpoint of reuse, the importance of requirements is function, non-function, fine or medium granularity, and representation formalization. That reasons are that the end users see their problems in this way and the search in the repository that solve these problems should start from these bases. In order to deal with these requirements, it is necessary to make requirements representation formalize so they are more easily identifiable, comparable and can be related to each other(Miguel, 2004).

The most frequent approaches are scenarios, in diverse variations, goals, and business rules. The most widely used scenarios are the use cases, introduced by (Jacobson, 1993) and updated in UML(Booch, 2005). However, other variations should be considered, in particular business processes or workflows. The scenarios are usually based on natural or structured language. Thus, from the point of view of reuse, it is convenient that this type of requirements follow some kind of norm which allows them to be compared for their incorporation to new requirements.

From the structural point of view, (Durán, 1999) breaks down the scenarios (as use cases)into their elemental parts. This possibility of breaking a requirement down into its atomic parts is fundamental in order to exchange, or even automatically generate, requirements in different formats, compatible with different tools.

Our approach to generate maps was based on the linguistic patterns proposed by Durán, and on the standardization proposed by Miguel. The former can be used both during elicitation meetings with clients and users and to create a case graph(CG). The latter can be used in standard phrases which have been identified that are usual in requirements specifications. The structuring of the information in the form of a template and the standard phrases proposal facilitate the writing of the requirements.

For our purposes, the most interesting templates are those related to four styles as mentioned above.

Figure 2 shows the reference framework used to standardize the requirements. It provides a preliminary definition of the user requirements and the functionality of the system is modeled from this by using a case graph (CG). By analyzing the case graph, business use cases (BUC) and use cases (UC) are obtained. Finally, the elements of use cases generated in this way are ready to be used in the context of requirements reuse. Consequently, the general framework leads to requirements elements that are suitable for being associated, through the repository management interface. In a similar way, other transformations can be defined in order to obtain scenarios, data flow diagrams (DFD) or activity diagrams.

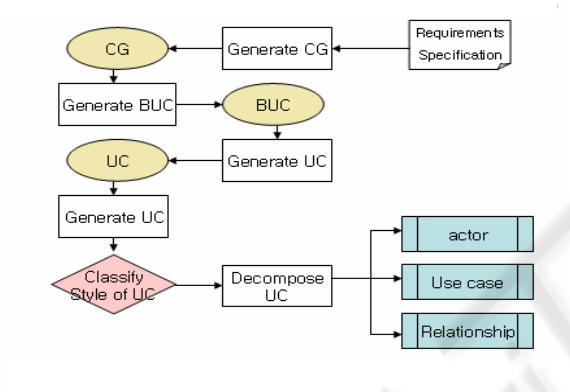

Figure 2: Generating Maps.

3.3 Gap Analysis

Having analyzed gaps of the elements that compose a map, we should identify the map gap types. The gap typology is composed of a set of operators applicable to Element. Each operator identifies a type of difference between requirements maps of the collected and the reused in repository. For example, as Modify is an operator, Use Case Element imply that there is the name difference between elements.

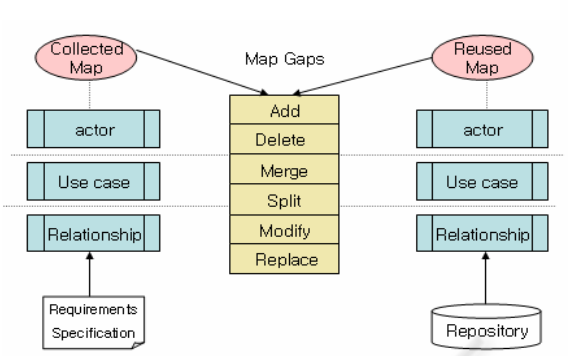

Figure 3: Map Gap Analisys in functional requirements style.

The generic gap typology identifies three major types of difference: style difference, element difference, and operator difference.

- ❑ Style difference is defined with the style of requirements. They only affect the way users want to refer to an element. Style is dealing with data, functional, quality, and managerial requirement.
- ❑ Element difference affects elements and defined with the le of requirements. They only affect the intention that users want to refer to an element. Element is dealing with use case, actor, and relationship.
- ❑ Operator difference are the most important as they correspond to express difference of the set of elements which composes the map. That is, operators are used to specify difference between requirements maps of the collected and the reused in repository.

Table 2 comprises 4 styles, 3 elements, and 6 operators. These are compounded to analyze gaps. The definition of each of the gap analysis is composed of a source and a consequence. The source identifies the status of the elements involved in the collected map, and in the reused map. The consequence is specifying difference between requirements maps of the collected and the reused in repository. Therefore, the consequence helps user to

Table 2: Map gap types.

Style	Element	Operator	Description of operators
Data	Use Case	Add	New element is add in the collected RE map
Function	Actor	Delete	Element is disappear in the collected RE map
Quality	Relationship	Merge	Elements are merged in the repository
Manage		Split	Elements are split in the repository
		Modify	Elements are modified in the repository
		Equal	Elements in the repository are same

established good requirements. For example, as operator is Add, the content of source is that use cases in collected requirement map is only exist, and the consequence assumes that new use case is add.

4 REQUIREMENTS ELICITATION PROCESS

The potential requirement elicitation process starts with the construction of the collected requirements. The refinement mechanism of map is used as a means to study gaps at different levels of detail. The refinement allows us to reduce single gaps expressed between top-level maps, and to move into a set of gaps between the refined maps. The process continues until developers and users affirm the refined map to be acceptable. Through the refinement process, the gap granularity issue is handled. More precisely, the process for eliciting gaps is an iterative one as follows(Colette, 2004):

Repeat until all maps have been considered.
① Construct the collected requirements map.
② Look for the reused requirements map in the repository.
③ Analyze gap between maps.
④ Add PMI from user.
⑤ Establish requirements.
⑥ Deliberate &Commit.

The five steps are carried out in a participative manner. This allows the consideration of different viewpoints with the aim of reconciling them cooperatively, in the construction of the collected requirements maps as well as in the refinement of gaps. Additionally, in step 4, users are given a gap report. Users can decide to add or delete elements and put their PMI(plus minus interest) in requirements specification. Each iteration is related to activities which: (We suppose that repository is already installed with successful projects' requirements map)

1. First, system analysts gather requirements with many elicitation methods. They make an initial requirements specification. Then, they construct the collected requirements map as input is the initial requirements specification.
2. Second, they look for similar situation in reused requirements repository.
3. Third, they analyze map gaps between requirements of the collected and the reused in the repository. Then, they make a gap report to users.
4. Fourth, they carry out the meeting with users to check the gap report. At that time, users give analysts own opinion and analyst should reflect users'PMI(plus minus interest) in the refined collected map.
5. Finally, system analysts establish the refined requirements specification. If the refinement is need, they carry out iteration from 1 to 5 again.

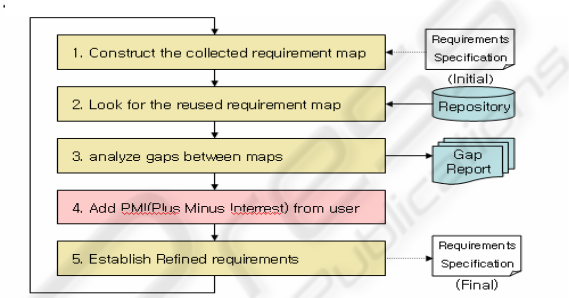

Figure 4: The process of elicit potential requirements.

5 A CASE STUDY

In order to demonstrate the feasibility of our approach, we illustrate the elicitation process with reuse-based map through the finance system case study, especially a credit subsystem. We suppose that only functional requirements exit in this case study. To understand situation corresponding a case study we describe firstly the initial requirements specification of the credit subsystem. Then, we illustrate three key activities for one iteration of the process, except looking for the similar reused requirements and establishing refined requirements.

5.1 Initial Requirements Specification

The credit system serves various mandatory functions because it is the basic business part in financial institution. A credit clerk should guide corresponding business to customers. A credit clerk approves a credit relationship after examination if a customer applies a credit. At this time, the customer can be individual or employee. It remains the other businesses such as submission payment bill, reporting to outside public institutions, and extending maturity.

The initial requirements specification of the credit

system is as following:

- ✓ **R1**. If customers apply a credit and submit corresponding documents, a credit clerk carries out document examining and requests a approval. An immediate credit approver can approves firstly the credit, and the final credit approver approves finally the credit.
- ✓ **R2**. The credit clerk pays loaned money to customer about approved credit item. At this time, the content of payment should be reflected to a account system.
- ✓ **R3**. The credit clerk transfers bill directly three times per a month about outside customer, and a salary are deducted automatically if the customer is employee.
- ✓ **R4**. Customer should pay the loaned money and interest according to a credit engagement. The credit clerk receives and handles payment.
- ✓ **R5**. The operator of the credit system requests payment to customer's surety when customer delayed payment.
- ✓ **R6**. The credit clerk carries out balancing account work.
- ✓ **R7**. The credit system can accessed through mobile equipments.

5.2 Constructing the Collected Requirements Map

In this stage, the collected requirements map is constructed. As mentioned 3.2(Figure 2), First, a case graph (CG) is build form requirements specification. Second, business use cases (BUC) are obtained by analyzing the case graph. Third, use cases (UC) are obtained by analyzing BUC. Finally, elements of use cases are obtained. These elements are 7 use cases, 7 actors, and 12 relationships.

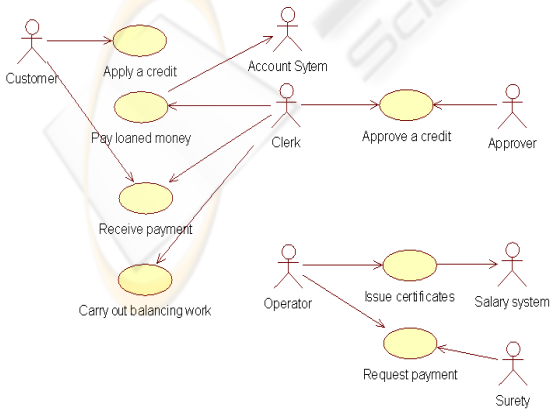

Figure 5: Use Case Diagram for a credit system.

5.3 Analyzing Gap Between Maps

In this stage, the gap analysis is carried out between the collected requirements map and the reused map. As mentioned 3.3, this analysis work is compounded of styles, elements and operators. The source identifies the status of the elements involved in the collected map, and in the reused map. The consequence is specifying difference between requirements maps of the collected and the reused in repository. The consequence can be expressed by operators.

Table 3 shows gap analysis results of the credit system about use case element. Merge operator is considered as complex analysis. This case can be confirmed when first gap analysis is finished and second gap analysis is carrying out using Delete operators. We got the gap report that U7 is added and 2 use cases should be check in case of use case, the same in case of actors, and 2 relationships should be check in case of relationship.

Table 3: Gap Analysis Results for use case element.

Use Case	Source		Consequence
	The Collected Map	The Reused Map	
U1	✓	✓	Equal
U2	✓	✓	Equal
U3	✓	✓	Add
U4	✓	✓	Merge*
U5	✓	-	Merge*
U6	✓	✓	Equal
U7	✓	✓	Add
-		✓	Delete
-		✓	Delete

5.4 Adding user's PMI

In this stage, new requirements are added with user's PMI(plus minus interest) after the gap report is check. User's PMIs are related new business and security issue as well as the content which is in the gap report.

These are new requirement from the gap report.

- ✓ GR1. The credit clerk carries out finish work of day and month.
- ✓ GR2. The credit clerk issues various certificates that the customer requires

These are new requirement from security issue.

- ✓ SR1. Users can access the credit system after user id is approved.
- ✓ SR2. The length of user's password is at least 6 when a user login.

6 CONCLUSION AND FUTURE WORKS

Frequent requirement change is very dangerous risk when we progress the software development project. These reasons are not to establishing good requirements in an initial phase of software development. It makes over time and cost of projects and low quality of software products. In the context of Requirements Engineering (RE), reuse is effective in particular because it can help to define requirement explicitly and to anticipate requirement change.

We proposed the reuse-based process approach to elicit potential requirements from various stakeholders. To achieve our goal, we first defined requirements styles, elements, and operators to construct the map. We presented analyzing gaps between requirements map of collected and reused in the repository. The gap analysis is composed of a source and a consequence. The source identifies the status of the elements involved in the collected map, and in the reused map. The consequence is specifying difference between requirements maps of the collected and the reused in repository. Also, we presented the potential requirements elicitation process with these maps. This process is sequential procedures to look for potential requirements in addition to Plus minus interest(PMI) method. Finally, we illustrated our approach through the credit system case study. We believe that our approach contributes to elicit potential requirements efficiently. This can reduce requirements changes and reduce time or cost problem corresponding uncertain requirements.

We are conscious of the lack of consideration related to definition of operators and handling of complex operators such as merge and split. In addition, association conditions between elements should be analyzed in details during the gap elicitation process. These are considered in the further steps of our research.

REFERENCES

Ian F. Alexander, Richard Stevens, 2002, Writing Better Requirements, Addison Wesley.

Lauesen, 2002, Sofeware Requirements: Styles and Techniques, Addison Wesley.

Ounsa Roudiks, Mounia Fredj, 2001, A Reuse Based Approach for Requirements Engineering, IEEE.

Jacob L. Cybulski1, Karl Reed2, 2000, Requirements Classification and Reuse:Crossing Domain Boundaries, LNCS 1844, pp. 190-210.

Colette Rolland, Camille Salinesi, Anne Etien, 2004, Eliciting gaps in requirements change, Requirements Eng(2004) 9:1-15.

Information technology-information resource dictionary system(IRDS) Framework, 1990, ISO/IEC International Standard.

Marttiin P, 1994, Methodology engineering in CASE shells:design issue and current practice, PhD thesis, Computer science and information systems reports.

Plihon V, Rolland C, 1997, Using a generic approach to support the construction of methods, *Proceedings of the 8th international conference on database and expert systems applications (DEXA'97)*.

Prakash N, 1999, On method statics and dynamics, Inform Syst.

Miguel A. Laguna, Oscar L´opez, Yania Crespo, 2004, Reuse, Standardization, and Transformation of Requirements, *ICSR*, LNCS 3107, pp329-338.

I. Jacobson, M. Christerson, P. Jonsson, G. vergaard, 1993, Object–Oriented Software Engineering:A Use Case Driven Approach, Addison–Wesley, 4 edition.

G. Booch, J. Rumbaugh, I. Jacobson, 2005, The Unified Modeling Language User Guide, Addison–Wesley, 2 edition.

A. Durán, B. Bern rdez, A. Ruiz, M. Toro, 1999, A Requirements Elicitation Approach Based in Templates and Patterns, *WER'99 Proceedings*.