Software Requirement Text Reuse

Sooyong Park

Center for Software Systems Engineering
George Mason University
4400 University Drive
Fairfax, VA 22030-444

Tel: (703) 993-3684
Fax: (703) 993-1521
Email: spark@mason1.gmu.edu

Abstract

Reuse is widely recognized as a solution to improving software productivity, quality and reliability. While there has been much research on code and design reuse, there has been little research effort extended in the requirement reuse. However, among the software development phases, requirements analysis is one of the most important and least supported phase of the software development process. The requirement reuse is proposed to increase productivity and quality of requirement analysis phase.

We focus on requirement text reuse research since requirements are, in most case, expressed in natural language. We use conceptual and numerical clustering techniques to build reusable requirement components. Automatic indexing techniques are applied to classify the components. The architecture of a requirement reuse support system and the contributions of this research are included in the conclusion.

Keywords: Software Reuse, Requirement Reuse, Text Reuse, Requirement Engineering, Requirement Clustering and Indexing

Workshop Goals: Exchanging idea of current research issues

Working Groups: Domain Analysis/engineering, Reuse and OO Method

1 Background

The George Mason University (GMU) Center for Software Systems Engineering (CSSE) research in systems and software requirement engineering is directed to improve and promote the development of requirements that reflect the needs of the user. The framework for this research is an Advanced Integrated Requirement Engineering System (AIRES). AIRES includes requirement assessment, prototyping, transformation and reuse. AIRES processes the requirements as a natural language text.

The requirement reuse project was initiated by the observation that, even though there are many common requirements, we do not have any method to reuse. In development of complex software system, eliciting and analyzing the system requirements has been a serious problem. In many cases, erroneous requirements produced a wrong product[1]. One way to alleviate that problem is to reuse requirements from an already-developed system. The reused requirement can be used as an example so that it can be a solution of the well known problem "users often do not know what they want". For example, in development of an Automatic Gas Station(AGS), users knew that they needed a security system but, in many cases, they did not know what specific security they wanted. If we had security requirements in an Automatic Teller Machine (ATM) system, the security requirements of ATM system could be a good example in AGS system.

2 Approach

Since we are dealing with natural language requirement, it is very to abstract or parameterize the natural language requirement automatically. But our purpose in requirement reuse is to provide requirement examples. Therefore, parameterization or abstraction of natural language requirement are not necessary. The issue is how to cluster the requirement sentences to build reusable requirement components. For example, a requirement about security can appear through out the requirement document. Extraction of these requirements is a key to building reusable requirement components.

Our approach to requirement text reuse has two phases - the requirement component extraction phase, and the classification and retrieval phase.

2.1 Requirement Reuse Process

The requirement reuse process consists of four steps. These are;

STEP 1: Text Analysis

STEP 2: Component Extraction

STEP 3: Component Classification

STEP 4: Retrieval

The step 1 is to identify requirement concepts. We use requirement concepts for objects, functions, and quality goals of the system requirements. Based on the requirement concepts, clustering is performed to create reusable components. The reusable components are indexed by automatic indexing techniques. Following subsections discuss each processing step in detail.

2.2 Text Analysis

The purpose of this process is to develop the rules and algorithms to identify the requirement concepts automatically. In the identification of requirement concepts, there are two approaches;

1. Using extensive domain knowledge and reasoning process. – This is an AI approach.
2. Using lexical analysis and syntactic pattern analysis.

We adopted the second approach since it is typically more practical and also domain independent. The object and function concepts can be identified by syntactic pattern analysis and noun and verb classification. The identification of quality goals use of a quality goal table, thesaurus analysis, and syntactic pattern analysis.

2.3 Reusable Component Extraction

Reusable component extraction is performed by clustering requirement with respect to identified requirement concepts. We use both numerical clustering and conceptual clustering. ¿From these two clustering techniques, we can extract requirements that relate to each identified requirement concept. The extracted requirements are body of reusable requirement component with respect to the requirement concept.

2.4 Component Classification and Retrieval

This process is based on automatic text indexing techniques which have been established in information retrieval area[2, 3, 4]. The main technique is extracting lexical affinities from the requirement component. The observation of lexical affinities in large textual document has been shown to convey information on both syntactic and semantic levels and provides us with a powerful way of taking context into account[5]. Lexical affinities will serve as indices of the component.

3 Related Work

There is no known directly related work, but the research is related to two other areas of research;

- Informal requirement specification process by linguistic analysis[6, 7, 8].
- Specification reuse by analogy[9, 10]

Since we are focusing on reuse in this paper, we will discuss the second item.

Maiden and Sutcliffe[9] proposed to reuse domain abstractions by analogy analysis. From the description of the system, they try to find the similarities among the already prepared domain abstractions. The system shows the domain abstractions with analogical mapping between the retrieved domain abstract and the target system domain. They reported that it was useful for novice software engineers in analyzing the problem domain since domain abstractions can be a good example.

The problem is that preparation and representation of domain abstractions is a non-trivial task. Also, there is no guidelines for domain abstraction. To find an analogy, the system need to be specified or understanding of the abstract requirements is need. This means the software engineer need to understand the abstract requirement to reuse abstract requirement which can be contradiction.

Miriyala and Harandi[10] proposed specification derivation from the existing specification by analogy. The general concept is similar to that of Maiden and Sutcliffe but they found the similarity in the specification structure instead of in the domain abstraction. They view the system as a tree structure. By the similarities in edges and nodes, they retrieve the specifications. To be processed by analogy reasoning, all the system specification need to be represented as structured diagram like tree structure. The target system needs to be represented in the same way. The problem is that there is no unique system structure for a problem domain. In other words, the system structure can be represented in different tree structures depending on the perspective.

In general, the analogy approach in specification sounds reasonable since we can only reuse when there are similarities. However, these approaches need a lot of domain knowledge. Preparation of that domain knowledge can be as difficult as requirement analysis.

4 Conclusion

Based on discussed requirement reuse process, we are developing a supporting tool called the Requirement Reuse System (Re^2S). The general architecture of Re^2S is shown in figure 1.

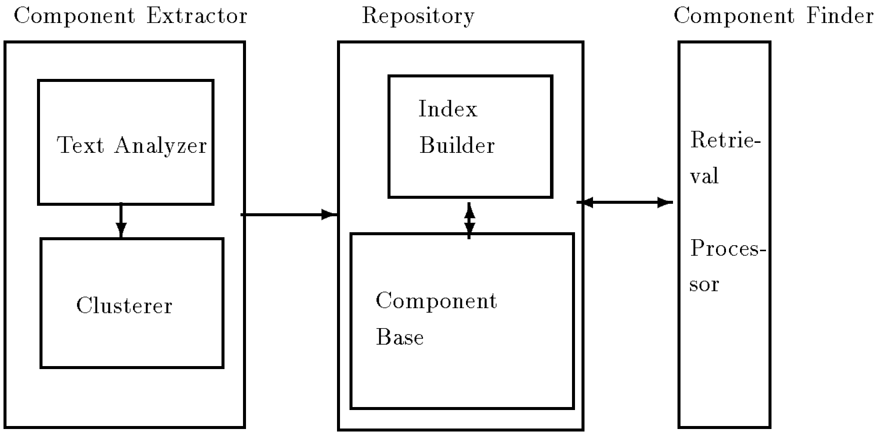

Figure 1 General Architecture of Requirement Reuse System

The contribution of this research includes

1. Increased productivity by reducing the risk and effort needed to develop requirements.
2. Increased reuse effect by improved cross-referencing of requirement component to design, code, and test segment.

3. Text reuse method can be applied to various software documentation reuse.

Besides implementation of Re^2S, the future research will include an empirical study of requirement elicitation and analysis by reuse, and study of traceability between reusable requirement and design components.

References

[1] A. Davis, *Software Requirements Analysis and Specification.* Englewood Cliffs, New Jersey 07632: Prentice Hall, 1990.

[2] Y. Maarek, D. Berry, and G. Kaiser, "An in formation retrieval approach for automatically constructing software libraries," *IEEE Trans. on Software Engineering*, pp. 800–813, Aug 1991.

[3] M. Luhn, "The automatic creation of literature abstracts," *IBM J. Res. Develop.*, pp. 159–165, Apr 1958.

[4] J. Palmer and Y. Liang, "Indexing and clustering of software requirements specification," *Information and Decision Technologies*, vol. 18, pp. 283–299, 1992.

[5] F. Smadja, "Lexical co-occurrence: The missing link," *J. Assoc, Literary and Linguistic Computing*, vol. 4, no. 3, 1989.

[6] M. Saeki, H. Horai, and H. Enomoto, "Software development process from natural language specification," *11th International Conference on Software Engineering*, pp. 64–73, 1989.

[7] R. Abbott, "Program design by informal english description," *Communication of ACM*, pp. 882–894, November 1983.

[8] C. Rolland and C. Proix, "A natural language approach to requirement engineering," *4th International CAiSE Conference*, pp. 257–277, 1992.

[9] N. Maiden and A. Sutcliffe, "Exploitung reusable specifications through analogy," *Communication of ACM*, April 1992.

[10] K. Miriyala and M. Harandi, "The role of analogy in specification," *6th Annual Knowledge-Based Software Engineering Conference*, pp. 117–126, 1991.

5 Biography

Sooyong Park is a research assistant and doctoral student in the School of Information Technology at George Mason University. His research interests are software reuse, requirement engineering, and Object-Oriented domain analysis in real-time system. He has been researched in software reuse area for three years.

Sooyong received a BS in computer science from the Sogang University in Seoul Korea, an MS in computer science from the Florida State University.