Requirements Reuse for Software Development

Oscar López Villegas
Technological Institute of Costa Rica
San Carlos Regional Campus
olopez@infor.uva.es

Miguel Ángel Laguna
University of Valladolid
Department of Informatics
mlaguna@infor.uva.es

Abstract

This paper presents doctoral research addressing the problem of applying a reuse based approach to the early phases of software development, reusing requirements models. Our proposal is aimed at defining meta-model for requirements diagrams, a Petri net based approach for representation of requirements models, and an approach to compare the requirements proposed by the software developer against the domain requirements models stored in a repository. Being able to determine the similarity between the requirements models, through an operative support, makes it possible to answer the developer´s queries ensuring the system requirements satisfaction by selecting assets from complex structures called mecanos.

1. Introduction

Software production has a three-constrain-set: quality standards, time-to-market, and requirements. Software applications must satisfy strict quality standards, they must be produced as soon as possible, and they must satisfy a defined set of system requirements. Related to quality standards and time-to-market, several software reuse approaches as alternative ways of solution have been proposed. Software reuse is aimed to significantly increment the quality and productivity of software artifacts. Software quality is favored by using proven and validated components. Software productivity is improved by reducing time-to-market for software applications. Besides qualified software is supposed to adequately satisfy the system requirements, software reuse per se does not ensure that system requirements are really satisfied. An approach ensuring the system requirements satisfaction by reusing software components is necessary. As a result, we take *satisfactory software development by reusing components is achieved through an approach based on software requirements reuse* as hypothesis of present research.

Software reuse implies more than simply using components to make the same application, or a similar one, over and over. True software reuse consists of the systematic use of experience and previously developed systems as *assets* which have to be cultivated to effectively better the future development of applications. The *asset* term refers to any reusable product in the software life cycle (models and domain architectures, requirements, designs, code, data base components, documentation, tests) [9]. Different models of software reuse have been established, e.g. domain engineering - application engineering, proposed by U.S. Department of Defense [5], and the development for - development with reuse, proposed by REBOOT [9].

Activities of development for reuse and development with reuse are related to a components library and based on operative support to enact, manage and use the reusable components. The assets may be reused from two perspectives: composition and generation. In composition the assets are taken as passive elements which act like construction blocks to compose software applications. On the other hand, in generation, the assets are viewed as active elements which might be transformed to obtain software applications. It could also be thought of as some hybrid composition-generation, as has been proposed at the University of Valladolid in the MRG approach (in Spanish, Modelo de Reutilización GIRO). The MRG is based on fine grain elements (they can be of three kinds: analysis, design, and implementation) and large grain elements (structures that are formed by related assets of different levels of abstraction) where the generation of large grain elements can take place by composition of fine grain elements in reuse time. This hybrid focus provides great flexibility in the reuse process because the assets recovered from the repository are used to generate the new required elements, according to developer necessities.

Software requirements have to be treated adequately inside of a reuse strategy. Both in composition and generation environments the reuse process is triggered by a set of requirements which are proposed by the developer. These requirements are supposed to satisfy reusing assets which are stored in the repository. Requirements have to previously be classified to be retrieved when developers send their queries. Classification and retrieval of general assets have been approached with schemes and methods based on text, ontology or facets. Nevertheless, requirements contain knowledge from both the domain and the development process. The complexity of this knowledge forces the application of sophisticated techniques for requirements classification and retrieval [6]. To efficiently answer the developers´ queries by selecting, composing or generating elements, a robust strategy based on software requirements reuse is required.

Systematic reuse of software requirements demands a special operative support to enact, manage and use the requirements assets. It has to be defined correctly in order to represent and to store specifications in the development for reuse step. Then, for an effective requirements reuse, the way to compare and to adapt the reusable elements in the step of development with reuse has to be defined. With necessary operative support, software requirements reuse improves general software reuse if system requirements satisfaction can be ensured when answering the queries of the developers. In this way, an approach for requirements reuse drives the software development process inside of quality, time-to-market, and requirements constraints.

In summary, software reuse aims to satisfy with good quality and efficiency the demand for software products. The correct satisfaction of requirements is a feature of any qualified software product. The issue of reusing the design and implementation elements is a requirements driven one. An adequate operative support makes it possible to answer developer queries and hence ensure the system requirements satisfaction. The establishment of a requirement reuse approach supporting the software development with reuse is the fundamental objective of the present research.

2. Background

Software reuse is aimed to systematically use different products of life cycle to develop new software products based on a components library [17] as opposed to the traditional develop-from-scratch approach. The assets have to encode the information about development from different levels of abstraction. This information is referred to when comparing new situations with previous ones, and during duplication of already developed actions and objects, as well as during adaptation to support new requirements [16].

Complex structures, like mecanos [7], have been proposed because of the necessity to relate the different levels of abstraction. A mecano is a large grain reusable element consisting of a set of fine grain elements which correspond to distinct levels of abstraction and are associated by inter-level and intra-level relations. The mecano is designed to increase the abstraction level of reuse and to facilitate traceability and navigation between components inside the software reuse process. The higher level of abstraction of software relies on its requirements. Nevertheless, in order to integrate software requirements in complex reuse structures, adequate models to promote reusability (e.g. comprehension, retrieval, and adaptation) are required.

2.1. The requirements reuse

Since requirements engineering triggers the software development process [19], and since requirements engineering is in charge of the necessities of stakeholders [10], the requirements reuse can empower the software life cycle [6]. Requirements reuse is an approach to systematically use existent requirements documents to reduce the general effort inside the software life cycle. Although it has received little attention [10] [21], reusing early software products and processes can impact the life cycle from two basic points of view: improving the requirements engineering process [6][20], and supporting the development with reuse [3] [4] process.

From the point of view of improving requirements engineering, requirements reuse aids by recording the adopted suppositions, made decisions, and adopted alternatives for future reference. It makes the change management of requirements easier. More, requirements reuse is helpful in the assistance, guidance and advising for the requirements engineer in the process of requirements acquisition [19]. It is assumed that both process and products of requirements engineering are reusable artifacts. In the whole requirements process there exist different models and ways to enact, handle and use models. Both, products and process are potentially reusable.

From the point of view of supporting development with reuse, requirements management is essential for supporting domains and product lines [8] [2]. In addition, the ideal of software reuse claims the knowledge in abstract form for reuse. The more abstract level of knowledge about a particular domain is represented by requirements. Hence, reusing requirements is a way to increment the potential of software reuse. Requirements related to constraints inside a domain, or to styles of data presentation or to policies of the organization are all potentially reusable. These categories could represent more than 50 percent of the requirements. Then, requirements reuse is an attractive alternative in the perspective of reducing software development costs [10].

Besides its potential benefits in the software engineering, requirements reuse faces as principal trade-off its difficulties to enact, to process and hence to reuse, the requirements. The documentation of the requirements is originally oriented to be a communication media between users and analysts. For this reason, its representation is done with diverse notations and formats. This diversity implies the necessity of particular actions to analyze documents of requirements and its organization in a repository of reusable artifacts [6]. The systematic requirements reuse to develop software requires two specific actions. First, to define the adequate way to model and to store specifications in the phase of development for reuse. Second, to define a process to compare and to adapt the reusable requirements in the phase of software development with reuse.

2.2. Requirements representation for reuse

Common classification and retrieval techniques show limited utility in representing requirements for reuse [6]. Some different alternatives based on knowledge representation [14], analogical reasoning [15] to reuse the requirements from a knowledge base have been proposed . These techniques place emphasis on the semantics of requirements documents and it demands artificial intelligence applications to acquire and to manage the encoded knowledge in requirements documentation. Others techniques are based on meta-models [19], evolutionary development [3] and formal methods [22] all of which emphasize the process for development and maintenance the reusable requirements. Cybulski [6] has proposed a set of techniques based on structural properties of documents and tasks emphasizing the way to enact and to use the requirements in the life cycle of requirements engineering. All these techniques rely on the reuse of requirements as a way to improve the requirements engineering process.

A few studies are intended to reusing the requirements for software development. Besides it is known that requirements from domains and similar tasks are more likely to show similarities than others software elements [21], the requirements reuse as support to develop applications has received relatively little attention. A tool and methodological support is necessary to reuse from the requirements toward the implementation [1]. Complex structures to accomplish the requirements related to design and code have been proposed [7]. Nevertheless, to integrate assets of requirements and design and code it is necessary to find out how to adequately model them. Requirements are modeled with a diversity of techniques. Hence we propose a normalization process for requirements [12] to ensure requirements reusability when stored in a repository. The normalized requirements when related to assets of different levels of abstraction provide an interface of large grain to increase the abstraction level of reuse process.

2.3. Comparing and adapting requirements

The techniques for comparison and adaptation of reusable requirements have received less attention than others in the requirements reuse area. Reubenstein [18] started this work when he proposed a heritage-based technique to reuse domain descriptions and task specifications. Maiden and Sutcliffe [20] apply techniques based on artificial intelligence to support the structural and semantic matching when retrieving requirements. Frameworks and patterns in general software reuse have shown their utility for classification and characterization of reusable requirements. Nevertheless, all these works are intended to improve the requirements engineering process.

The requirements reuse for supporting software development requires a high abstraction level. We have proposed an abstraction of generic functional requirements which, being able to be adapted to various applications, could be combined with generalization criteria in order to manage assets from the repository [11]. Generalization criteria provides the necessary support for the requirements description and these criteria make the generic requirements compatible inside the classification scheme. The generic part acts as the element for the comparison [13] between both problem requirements and assets from the repository. From this comparison it is possible to retrieve assets to apply them to a given problem and to adapt them to satisfy the requirements.

3. Research Plan

3.1. Problem Definition

Inside the general frame of this doctoral research, the problem to be resolved is established in following terms:

Let S be a given set of requirements which are proposed by a software developer with reuse. S represents a subset of a software requirements specification, then the problem we are dealing with is to supply an approach for selecting the minimal model M such that M satisfies all the requirements in S, and M belongs to the universe of requirements models which are stored and classified in a repository, and they are derived from domain analysis applied on specific business areas.

3.2. Objectives

The general objective of this Thesis is to define an approach for comparing requirements models, looking at model inclusion and the satisfaction of specific properties, to support the establishment of an approach for software development based on reusing complex structures called mecanos in the MRG environment. Specifically, our aim is to:

1. Establish a way to represent and relate the requirements which are stored as assets in a repository.

2. Establish an equivalence relation between requirements models and the sufficient condition to determine

the similarity between the requirements models.

3. Establish a process to compare requirements so that it supports software development by reusing mecanos from the starting point given by developer queries.

3.3. Expected products

We expect to obtain three main products from this researching.

Wider the mecano model. We are going to propose an adaptation of the mecano model in order to focus on issues about domain engineering. Specifically, it will get a specialized mecano model for including generic requirements.

Process for requirements reuse. It will establish a work guide for developing software applications with reuse from functional requirements and focusing on recent advance in process theory and software reuse.

Tools for requirements reuse. It will specify and implemented, if possible, some tools for an ease-to-reuse process from functional requirements. At least it should be specified corresponding tools for: (a) Automatic generation of requirements assets from domain models. (b) Requirements abstraction for obtaining generic assets. (c) Support for comparison of requirements models. (d) Requirement-driven navigation through the mecano structure.

4. Proposal Overview

We have focused on six issues to be resolved in order to establish an approach for requirements reuse supporting software development, see figure 1. First, we are based on (A) the availability of good quality and proved assets of requirements which are related to design and/or implementation assets, and (B) the support given by GIRO Reuse Model and the repository. The issues dealt with are:

1. *Diversity of techniques for requirements modeling*: There are many different diagrams for the representation of software requirements. We classify them in three categories (interaction oriented, data oriented, and object oriented). We are proposing a requirements meta-model to manage this diversity of techniques.

2. *Low formality level in requirements diagrams*: Because software requirements diagrams are intended to facilitate communication and agreement between stakeholders strong formal models are generally avoided. An adequate formal level to automatically investigate the behaviour of requirements models is necessary. We propose a coloured Petri net based approach to solve the low formal level in requirements diagrams stored in a repository.

3. *Relationships between requirements*: It is well known that requirements are related each to the others. Some requirements are in a mandatory relationship, or in an optional one. The requirements could also be alternative each to the others. We are defining a way to represent these relationships inside the mecano structure to store them within the assets.

4. *Requirements are specific to a given software solution*: Requirements elicitation normally takes place for a specific problem, then requirements diagrams show this particular reality. As a result, we are dealing with generic requirements, i.e. domain requirements into assets.

5. *How to compare requirements models*: Requirements assets are composed by requirements models. Also, the requirements which are proposed by the developer with reuse have to be taken as a model. As a result, we are working to define a process for comparison of requirements models.

6. *Process for software reuse from functional requirements*: From former issues we are establishing a process to give guidance and assistance to the developer when reusing assets and mecanos.

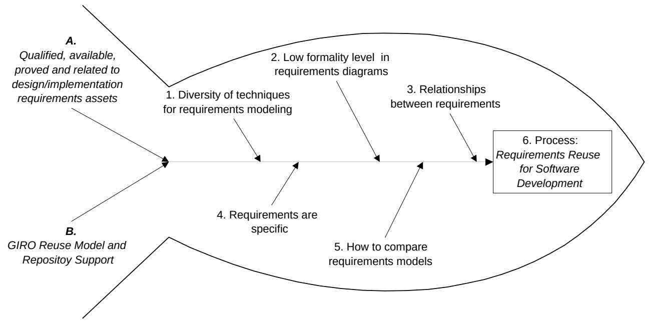
A.
Qualified, available, proved and related to design/implementation requirements assets
B.
GIRO Reuse Model and Repositoy Support
1. Diversity of techniques for requirements modeling
2. Low formality level in requirements diagrams
3. Relationships between requirements
4. Requirements are specific
5. How to compare requirements models
6. Process: Requirements Reuse for Software Development

Figure 1. Overall focused issues.

Acknowlegments

This work is sponsored by DOLMEN Project inside of CICYT-TIC2000-1673-C06-05, Ministry of Technology and Science, Spain.
Oscar López thanks to Spanish Agency for International Cooperation (AECI) and to the Ministry of Technology and Science of Costa Rica.

References

[1] K. Barber, S. Jernigan, and T. Graser. Increasing opportunities for reuse through tool and methodology support for enterprise-wide requirements reuse and evolution, 1999.

[2] L. Bass, P. C. Clements, S. Cohen, L. Northrop, and J. Withey. Third product line practice workshop report. Technical Report CMU/SEI-99-TR-003, Software Engineering

Institute. Carnegie Mellon University, Pittsburgh, PA 15213 (USA), March 1999.

[3] R. Bellinzona, M. G. Fugini, and V. de Mey. Reuse of specifications and designs in a development information system. In N. Prakash, C. Rolland, and B. Percini, editors, *Information System Development Process*, pages 79–96, Amsterdam, 1993. North-Holland.

[4] R. Bellinzona, M. G. Fugini, and B. Pernici. An environment for specification reuse. Technical Report POLIMI-UDUNIV.92.E.2.9E.8.4, ITHACA, Nov 1992.

[5] R. E. Creps, M. A. Simos, and R. Prieto-Díaz. The STARS conceptual framework for reuse processes. In *Proceedings of STARS'92*, November 1992.

[6] J. L. Cybulski. Patterns in software requirements reuse. Technical report, Department of Information Systems. University of Melbourne, July 1998.

[7] F. J. García. *Modelo de Reutilización Soportado por Estructuras Complejas de Reutilización Denominadas Mecanos*. PhD thesis, Universidad de Salamanca, Spain, 2000.

[8] F. J. García, M. P. Romay, J. M. Marqués, and Y. Crespo. Mecanos: Exposición resultados y líneas abiertas en reutilización sistemática del software. In *Actas de Las II Jornadas Iberoamericanas de Ingenería De Requisitos y Ambientes Software (IDEAS'99)*, pages 193–204, San José, Costa Rica, Marzo 1999.

[9] E.-A. Karlsson, editor. *Software Reuse. A Holistic Approach*. Wiley Series in Software Based Systems. John Wiley and Sons Ltd, 1995.

[10] I. Kotonya, G.; Sommerville. *Requeriments Engineering: Processes Techniques*. USA Wiley, 1997.

[11] Ó. López, M. Á. Laguna, and F. J. García. Reutilización de requisitos para desarrollo de software en el modelo mecano. In *I Jornadas de Trabajo DOLMEN*, Sevilla, Spain, Junio 2001.

[12] Ó. López, M. Á. Laguna, and J. M. Marqués. Generación automática de casos de uso para desarrollo de software con reutilización. In *Actas de Las V Jornadas de Ingeniería Del Software y Bases de Datos (JISBD'2000)*, pages 89–101, Spain, Noviembre 2000.

[13] Ó. López, M. Á. Laguna, and J. M. Marqués. Reutilización del software a partir de requisitos funcionales en el modelo mecano. In *Actas de Las IV Jornadas Iberoamericanas de Ingeniería De Requisitos y Ambientes Software (IDEAS'2001)*, pages 104–116, San José, Costa Rica, Abril 2001.

[14] M. Lowry and R. Duran. Knowledge-based software engineering. In *The Handbook of Artificial Intelligence*, pages 241–322, Massachusetts, 1989. A. Barr, P.R. Cohen, and E.A. Feigenbaum, Editors. Addison-Wesley.

[15] N. Maidenand and A. Sutcliffe. Exploting reusable specification through analogy. *Communications of ACM*, 35(4):55–64, 1993.

[16] R. Prieto-Díaz. Classification of reusable modules. *Software Reusability, Vol 1. Concepts and Models*, 1:99–123, 1989.

[17] R. Rada. *Reengineering Software, How to Reuse Programming to Build New, State-of-art Software*. USA AMACON, 2nd. edition edition, 1999.

[18] H. Reubenstein and R. Waters. The requirements apprentice: Automated assistance for requirements acquisition. *IEEE Transactions on Software Engineering*, 17:226–240, March 1991.

[19] C. Rolland and N. Prakash. From conceptual modelling to requirements engineering. Technical Report Series 99-11, CREWS, 1999.

[20] A. Sutcliffe, N. Maiden, S. Minocha, and D. Manuel. Supporting scenario-based requirements engineering. *IEEE Transactions on Software Engineering*, 24(12), December 1998.

[21] A. van Lamsweerde. Requirements engineering in the year 00: A research perspective. In *22nd. International Conference on Software Engineering*, Limerich, June 2000. ACM Press.

[22] M. Wirsing, R. Hennicker, and R. Stabl. Menu - an example for the systematic reuse of specifications. In *2nd European Software Engineering Conference*, pages 20–41, Coventry, England, 1989. Springer-Verlag.