Reuse of Use Cases Diagrams: An Approach based on Ontologies and Semantic Web Technologies

Belén Bonilla-Morales[1], Sérgio Crespo[2] and Clifton Clunie[3]

[1] Universidad Tecnológica de Panamá
Panama City, Panama

[2] Universidade do Vale do Rio Dos Sinos
Sao Leopoldo, Brasil

[3] Universidad Tecnológica de Panamá
Panama City, Panama

Abstract

Software reuse is defined as the use of any artifact, or part thereof, created before, on a new Project. This practice has significant benefits in reducing costs and increasing quality and productivity in software development. Numerous approaches have been proposed aimed mostly at the source code reuse, but this type of reuse has its limitations because development platforms and technologies are constantly changing. Then, it is necessary to apply reuse over software artifacts created at higher levels of software life cycle such as requirements specification. This paper presents a tool for the reuse of use case diagrams by storing their information in OWL ontology and the use of Semantic Web technologies.

Keywords: *Software reuse, use cases diagram, ontology, Semantic Web.*

1. Introduction

Software reuse is defined as the process of creating software systems from existing software; it is the use of any device or part thereof, previously created in a new project [1].

Reuse of software has been the subject of research for many years in the software community because of the great benefits it provides, mainly in terms of reducing development time and cost, and increasing the quality of software systems [2].

In most cases, software reuse is associated only with source code reuse, but as indicated in its definition, we can reuse any type of software artifact [3]. Furthermore, source code reuse turns out to be problematic because the development platforms and technologies are constantly changing. Therefore, it is necessary and appropriate to apply software reuse over artifacts created at higher levels of software life cycle such as requirements specification which includes use case diagrams.

Use cases diagrams are a type of UML diagram whose purpose is to define graphically the functionality of a system in terms of actors, use cases and relations [4]. They have great importance as a technique for extracting and defining functional requirements from the user point of view [5].

Reuse of use cases diagrams then gives the opportunity to have formal definitions and validated functional requirements of a software system created earlier and thus be able to use them as often as necessary in different software projects. In addition, reuse could be applied on a higher level of abstraction, avoiding the limitations in terms of changes in technologies and platforms.

But it is not enough with the initiative and the definition of the process if we do not have the necessary tools to operationalize the idea of reuse of requirements specifications. Consequently, we implemented a tool that allows reuse of use case diagrams by storing, searching and retrieving them using ontologies and Semantic Web technologies. The application allows software engineers to store the information of use case diagrams into OWL ontology. Then, we can make semantic searches on the repository where we store the RDF triples that define the ontology. Thus, the software development teams can create their ontological database of use cases diagrams and to have an application that allows to search and retrieval them, and finally reuse them in new projects. It promotes a savings of time and effort during the stage of requirements analysis and modeling of software.

2. State of Art

Reuse of software is generally defined as the process of building or assembling software applications or software systems from previously developed software [1] [2]. Its application involves not only the source code, but also the different artifacts that occur during the life cycle of software like requirements, UML diagrams, tests, manuals, experiences, etc. [3].

The benefits gained through the reuse of software artifacts are many, but certainly the most important are:

- A reduction in development costs.
- An increase in product quality.
- An increase in productivity through improvement of the times in which it is developed new software projects.

Reuse of software has been a topic of popular debate and research for nearly forty years in the software community. Many approaches to software reuse have been proposed during this time, mainly oriented to the development of tools and methodologies.

Among the outstanding works we can mention the proposed by Monegan [6] who developed a tool called Object-oriented Reuse Tool (ORT) which supports the reuse of object-oriented software by maintaining a library of reusable classes and record information about reuse and information associated with design and verification. For his part, Gicca [7] developed a software tool in ADA language called Reuse System to promote the reuse of software components and requirements, high level design, source code, etc. through a repository. Unlike the tool proposed by Monegan, Gicca's tool supports reuse of software components in the different phases of software life cycle, not just source code.

On the other hand, Henniger [8] developed the tool CodeFinder which is composed of three main parts: the tool (PEEL) which is responsible for populating the repository with reusable components of functions and routines obtained from source code in Emacs Lisp [9], a search mechanism, and a fitting tool to refine the repository when necessary. The tool uses two techniques: an intelligent recovery method which finds information related to the query, and a query building supported through incremental refinement of queries. The system provides a user interface that implements the search and navigation mechanisms that allows the user to view and navigate the hierarchy of the repository and build search queries.

There has been more specific work focused on the reuse of UML diagrams, such as Blok and Cybulski [10] who proposed a method for reuse use cases specifications using WordNet language to classify the lexical and semantic flows events. The tool calculates the similarity of the use cases according to information obtained in the flow of events and uses an information retrieval technique. Meanwhile, Robinson and Woo [11] proposed techniques for reuse UML artifacts, specifically the sequence diagrams. The main idea of the work is to find the model that best fits the desired features or functionality through REUSE tool, which uses Subdue algorithm [12] to find links between different sequence diagrams using the information stored in elements of stereotypes such as names, classes, etc. These two works differ from ours in aspects like UML software artifacts to reuse and the methodology used.

Happel et al. [13] presented Kontor, an approach which aims to store and query XML-based metadata, on different software artifacts, including UML components, in a central repository to encourage reuse. It has several ontologies to describe knowledge about the artifacts and technologies and / or programming languages, software licenses, etc. The work also includes a number of SPARQL queries that can be executed by the software developer to recover pieces of software to fit a need to develop specific application. In this work, like in ours, ontologies are used to store and retrieve software artifacts, but it is not geared to specific reuse of use cases diagrams.

3. Reuse of Use Cases Diagrams: An Approach based on Ontologies and Semantic Web Technologies

As part of our research on the topic of software reuse, we developed a tool that allows reuse of use cases diagrams in UML. The main idea of this paper is to manipulate and store relevant information of use cases diagrams in an OWL ontology. By having this ontological representation of the use cases diagrams information within a repository, the tool allows the user to make parameterized queries, through a graphical interface, about the diagrams that he is interested in getting, while it increasing the accuracy of the results obtained by taxonomic characteristics, capacity for inference and management concepts that have OWL ontologies.

Below, it is presented the technologies used to develop the tool. Then, we define the architecture of the tool and its implementation details.

3.1 Technologies and Tools

In this section we briefly describe the technologies and software tools used during the development of our tool:

a) UML and Use Cases Diagrams

UML is the most used and known language to model application structure, behavior and architecture but also business process and data structure. It is a graphical

language for visualizing, specifying, constructing, and documenting a software system [14].
Use cases diagrams are a type of UML diagram. Use cases diagrams are a technique to specify the behavior of a system, capture its requirements and guide the development process [15].
The most important concepts that define the use cases diagrams are:
Actors: An actor is a representation of a person, system or machine that interacts with the software system being developed.
Use Case: A task that must be undertaken with the support of the system being developed. A use case represents a particular functionality of the system.
Relations: Represent dependency between use cases or between actors and use cases, so that we can set the behavior of the system by integrating each feature. Use case diagrams in UML support partnership, inclusion, extension and generalization relations.

b) XML Metadata Interchange – XMI

XMI is a standard of the Object Management Group (OMG) for exchanging metadata level information via XML. It is a specification for exchanging diagrams.
According to the active site OMG, XMI is defined as: "A model driven XML integration framework for defining, interchanging, manipulating and integrating XML data and objects" [16].
XMI architecture allows simplifying the communication between applications of different technologies saving much time and works, also enhances the reuse of objects and components.

c) Ontology Web Language – OWL

OWL is a markup language for publishing and sharing data using ontologies, which are defined as explicit specifications of conceptualizations, in order to enable the behavior and reuse of knowledge [17] [18].
OWL is a Semantic Web technology, which is defined by the W3C as an extended Web, endowed with greater meaning in which any Internet user can find answers to his questions more quickly and easily with better-defined information. Thus, through the Semantic Web we can get solutions to common problems in finding information through the use of a common infrastructure, whereby it is possible to share, to process and transfer information easily [19].

d) Java and Netbeans IDE

Java is an object-oriented programming language developed by Sun Microsystems, independent of the platform on which we use the applications made with this language. For its part, Netbeans is a development platform that lets us work with Java and other programming languages. It lets work with the Swing graphics library.

e) JDOM

JDOM is an open source library for handling XML data optimized for Java. It allows create, read and manipulate XML files easily in any Java application [20].

f) Jena Framework

Jena is a Java framework for building Semantic Web applications. It provides a collection of tools and Java libraries to help us to develop semantic web and linked-data applications, tools and servers. It includes a programming environment for RDF, RDFS and OWL, and persistent memory storage, a SPARQL query engine and an inference engine based on rules [21].

g) SPARQL Language

SPARQL is a query language and a protocol for accessing RDF. Like SQL, it is necessary to distinguish between the query language and the engine for storage and retrieval of data; for this reason, there are multiple implementations of SPARQL, usually associated with development environments and platform technologies. For our specific case, ARQ is used as a query engine that supports SPARQL [22].

3.2 Tool Architecture

The steps required to implement reuse of use case diagrams through our tool is defined as follows:
Creation of use case diagrams and exportation to XMI: The use cases diagrams are modeled using a CASE tool for UML like StarUML [22], Rational Rose [23], ArgoUML [24], among others. Once they are created, we export them via XMI export option offered by most of these programs. The XMI files are stored in a specific directory within the server.
Storage of information of use case diagrams into OWL ontology: Our tool has an option that allows the user to load XMI files derived from use cases diagrams in UML. Also it has a GUI that allows entering information which can enrich the knowledge we have of the diagrams. Immediately after the XMI file is loaded into the system, tool handles it internally to obtain information about actors and use cases. Once this information and the information entered through the GUI are obtained, it creates new individuals which form part of the OWL ontology.
Persistent storage of OWL ontology in MySQL: Once the new individuals are obtained, the ontology is stored in a MySQL database. Thus, the ontology and its instances will be available for later reference.
Search and retrieval use case diagrams in XMI format: Through the tool, the user can search use case diagrams

that were stored previously. The user employs a number of parameters to define his query, the tool interprets the user's request as a query over OWL ontology that is stored in MySQL, then it retrieves those individuals that match the query and finally results are presented as XMI files associated with these individuals or entities.
Figure 1 shows a component diagram that conceptualizes the architecture of our tool, according to the phases we have described here.

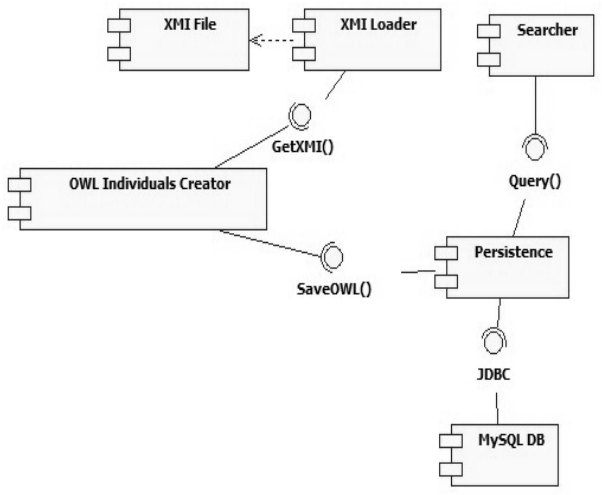

Fig. 1 Architecture of the tool

3.3. Implementation

Our tool is developed using Java programming language and the Jena framework. It consists of two main GUIs created using the Swing graphics library: one for the XMI file upload and insert additional information for use cases diagrams and another for searching and retrieving information.
It was created an OWL ontology, using the Jena framework, which has all the necessary classes and properties to store information of diagrams. The ontology provides a categorization by type of business and project.
Figure 2 shows the graphical interface through which the user uploads an XMI file and enters additional and relevant information associated with the use cases diagram. The user searches and enters the XMI file and fills additional information about the use case diagram. A class named XMILoad handles the XMI file loading process and obtains actors and use cases of the file with the help of JDOM, and the class named AddIndividuals builds the OWL structures necessary to add new individuals with their properties in the base ontology.

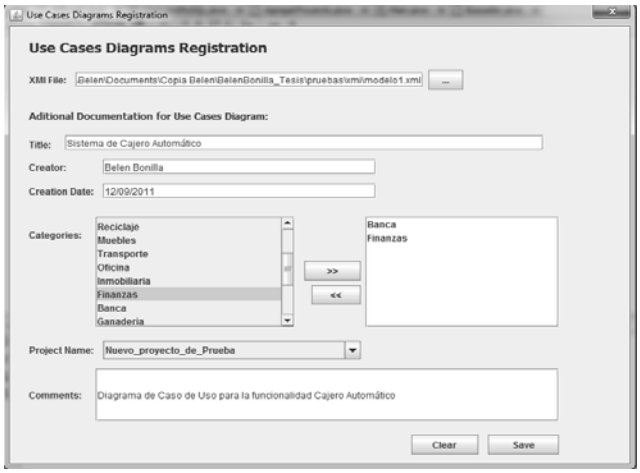

Fig. 2 GUI for Use Cases Diagrams Registration

The base ontology is stored in a MySQL database. Jena allows persistent storage of ontologies through its RDB subsystem. A class named AddIndividuals makes a connection to the base ontology stored in MySQL and adds any new individuals.
As part of the additional information, it is allowed to enter the category or type of business which is associated with the use case diagram, besides the project to which it belongs. This information is available because there is an interface that allows adds new categories and another to enter new projects. Then, these new categories and projects are also stored in the ontology as individuals thus allowing being loaded from the database to their use.
The search and retrieval of the use case diagrams in XMI format is carried out through two classes, one for the building of SPARQL queries and another for the retrieval and presentation of information.
Figure 3 shows the graphical interface to search use case diagrams in XMI format.

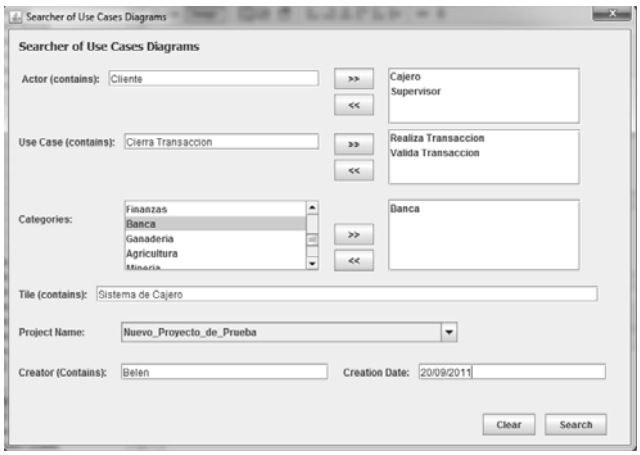

Fig. 3 GUI for Search Use Cases Diagrams

Through a class named GetQuery it is constructed SPARQL sentences from the information entered by the user via the GUI.
A class, named IndividualsRecovery, brings back and presents use case diagrams in XMI format which match with user search's parameters, including a brief description of them. This is possible because individuals in the ontology, that represent use case diagrams, have an ID which identifies them with their respective XMI file. The user will select the XMI file he wants and will download it for reuse.
Each time an XMI file that defines a use case diagram is downloaded for reuse, a weight will be added in the ontology so that it serves as feedback to determine how useful being the diagram and to give it priority in future searches.
Figure 4 shows the graphical user interface that presents the results obtained after executing the query according to the parameters entered.

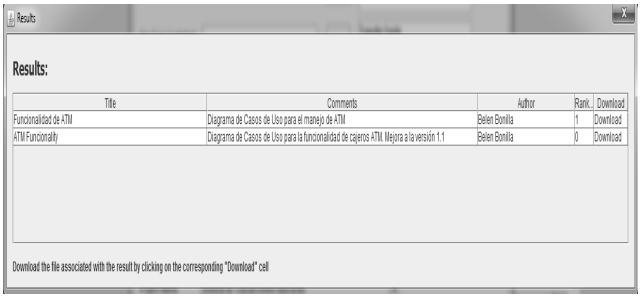

Fig. 4 GUI for Results Presentation

After the XMI file that defines the use case diagram required is downloaded, it can be opened through any UML modeling tool.
For example, Figure 5 shows how an XMI file that was recovered by the tool can be imported and displayed in Rational Rose, ArgoUML and StarUML.

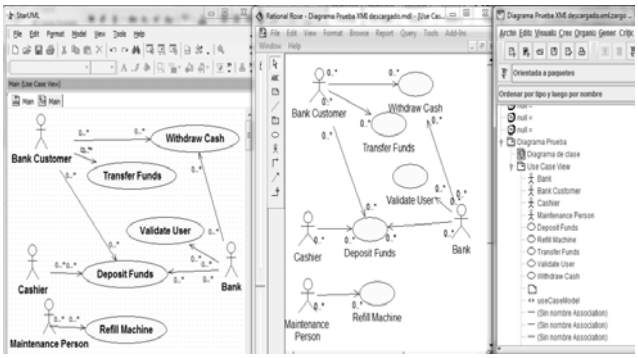

Fig. 5 Visualization of an XMI File retrieved through the tool

4. Conclusions

In this paper we present an approach to reuse use case diagrams by storing their information in OWL ontology and the implementation of a tool in Java, using Semantic Web technologies and tools like Jena framework and SPARQL query language. This tool allows querying individuals of the OWL ontology and retrieving associated use case diagrams, in XMI format, according to the users' input parameters through the GUI search.
The storage of the information of use cases diagrams in an ontology allows adding semantic to their definition, which benefits the process of searching and retrieving them, as they leverage the capabilities of inference that have the OWL ontologies, feature that allows to generate knowledge from previous knowledge.
We believe that the main advantages obtained with the use of this tool are: saving time and effort during the stage of requirements analysis and modeling, and reliability in the use case diagrams that have been recovered.
It is very important to remark that this work is limited to the reuse of use case diagrams. Currently it does not work for other UML diagrams.
As future work, we plan to carry out a series of case studies that test the features and benefits of the tool. Then, we will work on extending the tool to work with any UML diagram, not only use case diagrams.

Acknowledgments

Work funded by the Secretaría Nacional de Ciencia, Tecnología e Innovación (SENACYT) of Panamá through the Proposal No. APY-GC10-024B.

References

[1] C.W. Krueger, "Software Reuse". ACM Computing Surveys, Vol. 24, No.2, 1992, pp.131-183.
[2] W.N. Robinson, and H.G. Woo, "Finding Reusable UML Sequence Diagrams Automatically", IEEE Software, Vol.21, No.5, 2004, pp.60-67.
[3] Y. Kim, and E.A. Stohr, "Software Reuse: Survey and Research Directions". Journal of Management Information Systems, Vol.14, No.4, 1998, pp. 113-147.
[4] H. Eichelberger, "Automatic layout of UML use case diagrams", SoftVis '08 Proceedings of the 4th ACM symposium on Software visualization. Munich, Germany, 2008, pp.105-114.
[5] A. Cockburn, Writing Effective Use Cases. Addison-Wesley Longman Publishing Co. Inc., Boston, MA. 2000.
[6] M.D. Monegan, An Object-Oriented Software Reuse Tool. Technical Report, Massachusetts Institute of Technology Cambridge, MA, USA, 1989.
[7] G. Gicca, "Reuse system software repository tool concepts", ACM SIGAda Ada Letters, Vol.11, No.1, 1991, pp. 70-89
[8] S. Henninger, "An Evolutionary Approach to Constructing Effective Software Reuse Repositories", ACM Transactions

on Software Engineering and Methodology (TOSEM), Vol.6, No.2, 1997, pp.111-140
[9] R.J. Chassell, An Introduction to Programming in Emacs Lisp. GNU Press - Free Software Foundation, Boston, MA, 2009
[10] M.C. Blok, and J.L. Cybulski, "Reusing UML Specifications in a Constrained Application Domain", APSEC 98 Proceedings of the Fifth Asia Pacific Software Engineering Conference. Washington, USA, 1998, pp. 196.
[11] W.N. Robinson, and H.G. Woo, "Finding Reusable UML Sequence Diagrams Automatically", IEEE Software, Vol.21, No.5, 2004, pp.60-67.
[12] D.J. Cook, et al., "Subdue: compression-based frequent pattern discovery in graph data", Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations (OSDM 05). Chicago, USA, 2005, pp.71-76.
[13] H. Happel, et al., "KOntoR: An Ontology-enabled Approach to Software Reuse", Proceedings of the Eighteenth International Conference on Software Engineering Knowledge Engineering (SEKE'2006). San Francisco, USA, 2006, pp. 349-354.
[14] OMG, 1997. UML Resource Page. [online] Disponible en: http://www.uml.org/
[15] I. Jacobson, et al., Object-Oriented Software Engineering – A Use Case Driven Approach, Addison – Wesley, 1992.
[16] OMG, 2005. Catalog of OMG Modeling And Metadata Specifications. [online] Disponible en: http://www.omg.org/cgi-bin/doc?formal/07-12-02
[17] Bechhofer, S. et al., 2004. OWL Web Ontology Language Reference. [online] Disponible en: http://www.w3.org/TR/owl-ref/
[18] G. Guizzardi, "The role of foundational ontologies for conceptual modeling and domain ontology representation", 7th International Baltic Conference on Databases and Information Systems. Vilnius, Lithuania, 2006, pp.17-25.
[19] W3C, 2005. Guía Breve de Web Semántica. [online] Disponible en: http://www.w3c.es/divulgacion/guiasbreves/websemantica
[20] JDOM Project, 2000. JDOM. [online] Disponible en: http://www.jdom.org/
[21] SourceForge, 2000. Jena - A Semantic Web Framework for Java. [online] Disponible en: http://jena.sourceforge.net/
[22] G. Lausen, et al., "Foundations of SPARQL query optimization", Proceedings of the 13th International Conference on Database Theory (ICDT '10). Lausanne, Switzerland, 2010, pp. 4-33.
[23] StarUML. StarUML - The Open Source UML/MDA Platform. [online] Disponible en: http://staruml.sourceforge.net/en/
[24] IBM, Rational Rose Enterprise, [online] Disponible en: http://www-01.ibm.com/software/awdtools/developer/rose/enterprise/index.html
[25] CollabNet, 2009, ArgoUML. [online] Disponible en: http://argouml.tigris.org/

Belén Bonilla-Morales is a student of the Master of Science in Information and Communication Technology at the Technological University of Panama. She was awarded a Bachelor's degree in Engineering and Computer Science from the Technological University of Panama in 2009. Her research interests include Software Engineering, Semantic Web, Grid Computing and other topics.

Sérgio Crespo is a professor at the Universidade do Vale do Rio Dos Sinos - Brasil, PhD awarded by Pontifícia Universidade Católica do Rio de Janeiro.

Clifton Clunie is a professor at the Technological University of Panama, PhD awarded by the Universidade Federal do Rio de Janeiro.