EsaCake: A Semantic Software Environment for Sharing Software Projects Knowledge based on the ESA software methodology

Juan Miguel Gomez, Myriam Mencke, Javier Chamizo, Ricardo Colomo, Angel García-Crespo

Universidad Carlos III de Madrid
{juanmiguel.gomez, myriam.mencke, javier.chamizo, ricardo.colomo, angel.garcia}@uc3m.es

Abstract

There is an increasing need of defining standards at the beginning of any engineering project. The correct specification of standards has become essential in software development in order to handle correctly the development of projects. Nowadays the need to define standards at the outset of any engineering project is more evident than ever. The specification of standards has become an essential topic which universities try to teach to software engineering students. In fact, software creation processes are required intrinsically to be produced according to a systematic methodology to enable constant control during the project life cycle. This paper presents a new environment that enables semantic and social interaction of documentation produced during software development processes.

1. Introduction

Presently, the need to share and define metadata is a valuable tool for engineering projects, such as Knowledge Management [1] or Enterprise Application Integration [2] applications. Such metadata can be used for valuable knowledge sharing and also, it can optimize search from a semantic perspective, as discussed in [3].

In addition, the need for standards at the outset of any engineering project is more evident than ever. The specification of standards has become an essential topic which universities try to teach to software engineering students. It is an intrinsic requirement of the software creation process that it is produced according to a systematic methodology which enables constant control throughout the project life cycle. This methodology includes generating documentation for each stage of the life cycle, in order to accurately report all client requirements, functionalities, architectural design, and the complete detailed design of the software, as the project advances. The application and use of these standards is highly beneficial for every aspect of the finished software product, as well as for the continued maintenance of the product during its life cycle. However, the use of these standards is frequently viewed as laborious and tedious, and consequently, they are not applied as they should be. This problem generates a requirement for applications which assist software engineers to correctly apply these standards, making the process less laborious and more efficient.

This paper is organized as follows. The problem statement is outlined in section 2. The ESACake architecture proposed is discussed in section 3. Section 4 presents the implementation and tool evaluation. Finally, section 5 concludes the paper.

2. Problem Statement

When the development of a software application commences, several problems related to the application of software engineering methodologies appear.

The first problem which arises is the tediousness of working exactly according to the methodology specified in the ESA standards, and applying the guidelines. The above-mentioned standard is divided into different phases related to the analysis and design of a software product and, in each of these phases, a document is produced. These documents are:

- URD (User Requirements Document) – This document details what the client wants the system to do.
- SRD (Software Requirements Document) – This document explains, based on the user's needs captured in the user requirements phase, what the system will and will not do. In this

document all requirements and restrictions are defined.

- ADD (Architectural Design Document) – This document explains which architecture the system is going to be built upon. Modules, design patterns, etc. are defined.
- DDD (Detailed Design Document) – In this document, the exact design of the application is explained. Classes, methods and attributes are clearly defined in this phase. The next step is coding.

The simplest way to accomplish these phases is to employ the guides provided by the ESA for producing the stipulated documents in each phase. However, following the guidelines in this way results in a large amount of documents, particularly given that there may be many versions of each document.

One software development domain where such a problem often arises is in the case of UML diagrams. A development team may decide to construct their diagrams using pen and paper, or they may make use of a non-UML drawing application. The consequence of both options is the same; in the case of using a drawing application, a large amount of files is generated, and in the case of manual drawings, the development team has to store cumbersome amounts of papers.

Fortunately, in order to construct any kind of UML diagram, developers can choose from a wide variety of tools to aid them to make the diagrams, and benefit from another even more important functionality – the unification of all diagrams into projects. This enables developers to have an organized structure for all of the elements created.

Creating such a structure is the first problem which the application described in this paper attempts to address. ESACake is a tool with which developers can create all of the elements which comprise each of the phases of the ESA standard for software engineering, therefore, providing a means of effectively organizing and managing the elements.

The second issue which ESACake tries to solve is the actual use of the standard itself. The aim is to help analysts and designers to improve their experiences with developing applications, using the ESA methodology for software projects. Using ESACake it is possible to define the most important elements of each phase, that is: user requirements in the user requirements phase, software requirements in the software requirements phase, architectural components in the architectural design phase and detailed components in the detailed design phase. It is also possible to manage the relations between these components. This is one of the most interesting aspects of ESACake, as it assists the definition of each of the elements.

However, the definition of the elements itself is not a task which considerably contributes to an improved user experience, as the elements consist of a list of attributes which must be filled by the analyst or the designer in the table inside the corresponding document. As mentioned above, the major functionality which ESACake provides is the management of the relations between each element. Software requirements are derived from user requirements, and, at the same time, the software requirements help to define the design of particular components of the applications architecture or detailed design.

These kinds of relationships are created in the ESA standard documents as traceability matrices. There is one traceability matrix for each relationship: between user requirements and software requirements, between software requirements and architectural components, and between software requirements and detailed components. These traceability matrices are easily red, but laborious to create. It is at this point where ESACake is the most useful.

A third problem, related to the previous one, appears when an element of a specific phase is modified. Due to the relationships presented between different phases' elements and between the ones of the same phase, it is possible that modifying one element will affect other requirements and components. When this situation arises, the analyst or designer uses the traceability matrices to know which elements are related to the one that is being modified, and check them in order to fix possible changes. ESACake shows this information automatically when a modification is performed.

Finally, one of ESACake's mayor features is focused on knowledge reuse. It provides access to the documentation of one or more similar projects during the projects life cycle, which can help to speed up the design of the application. Previously, accessing these documents consisted of the laborious process of searching through large amounts of archived documents, and consulting each document was required in order to extract the useful information.

ESACake has a tool that allows the users to search in the application's database. In practice, a user can search for a requirement or a component with fixed characteristics and obtain it quickly.

So far, ESACake has been described only as an ESA methodology supporting tool for analysts and designers. However, ESACake tries to go further by adding some features in a way that improves the user experience, both with the application and with ESA methodology. Those features are: the design of ESACake as a Web application, and the addition of semantics to the elements of the different phases of the standard.

ESACake is a Web application, hence it does not need to be installed individually on every computer where it is required. This provides a great advantage, as ESACake users can work with the application from any computer with an internet connection and a Web browser, avoiding the requirement for cumbersome installations and configurations.

Regarding the semantic component, ESACake includes a semantic repository in which the information concerning the application's different elements is stored.The semantic repository is based on a requirements ontology defined for ESACake, which establishes the existing attributes of user requirements, software requirements, architectural components, detailed components, and the relations between them. Other ontologies, named domain ontologies, coexist with this requirements ontology, allowing users to link a project's elements, whether they are requirements or components, with the terms of a specific ontology, so that the available information about these elements will be enriched and the possibility of searching through them will be enhanced.

3. ESACake Architecture

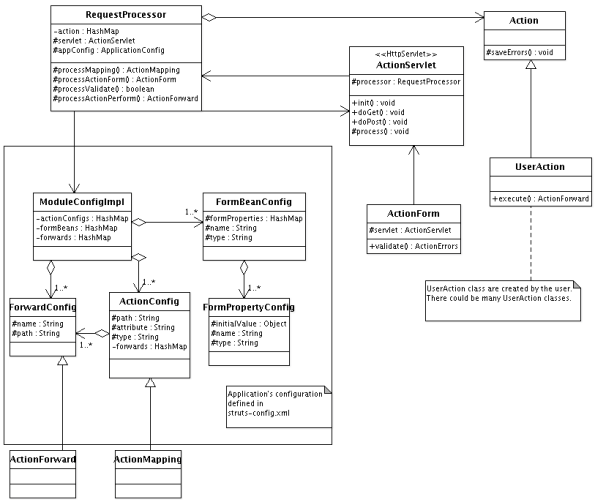
As mentioned in the previous section, ESACake is a Web application, so it is based on the client-server paradigm, ESACake being the server and a web browser the client.

The internal architecture of ESACake is based on a MVC (Model-View-Controller) pattern[1]. This pattern allows the separation of a business model, user interface and application's control logic in a way that it is possible to make modifications in each one of these layers without making great changes on the others. Struts framework has been chosen to apply a MVC pattern. Struts provides a controller called *ActionServlet* which evaluates user's requests and the corresponding responses based on a configuration file named *struts config.xml*.

[1] http://java.sun.com/blueprints/patterns/MVC-detailed.html

4. Evaluation and Implementation

ESACake has been developed using Sun Microsystems' JEE (Java Enterprise Edition) technology. This technology has been designed to develop and run distributed and multi-layered Java applications.



In ESACake, the classes that define the application's behaviour implement *Action* interface. These classes contain a method named *execute* that carries out the operations needed for each kind of action and they are in charge of accesing application's model, making the appropriated modifications on it. *Action* classes are supported by other classes named *ActionForm*. These classes gather the information introduced by the user in the form, validate it and make it available for the corresponding *Action* class.

The data layer in ESACake is divided into two elements: the database that stores the control information of the application, such as login information, and the semantic repository where all the data of user's projects is stored. This semantic repository leans on a database instance to obtain persistance.

Jena has been used to provide semantics to ESACake. Jena is a framework for Java that provides an API for writing and extracting data from RDF graphs. Jena has been chosen because, in contrast to other frameworks like Sesame, Jena provides OWL support.

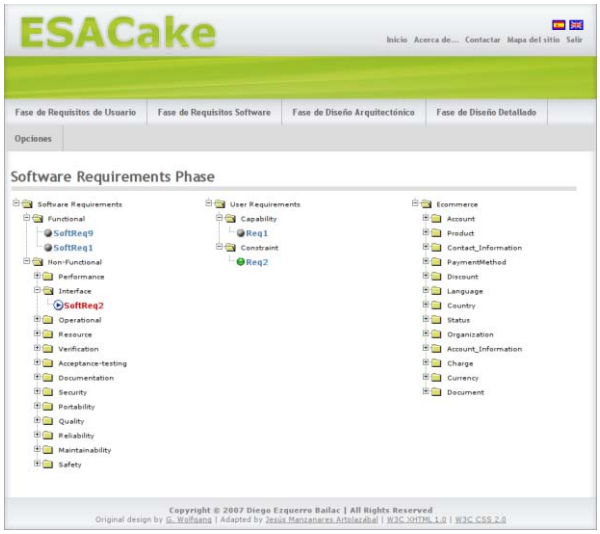
For improving ESACake's performance, the data layer manager SDB has been chosen instead of RDB (the default database manager in Jena). It has been specifically designed to work with SPARQL, the query

language developed by the W3C. The differences between them are taken from [7], the most important factor being that “RDB uses a denormalised database layout in order that all statement-level operations do not require additional joins. The SDB layout is normalised so that the triple table is narrower and uses integers for RDF nodes, then does do joins to get the node representation. This optimizers for longer patterns, not API operations. In SPARQL queries, there is often a sufficiently complex graph pattern that the SDB design tradeoff provides significant advantages in query performance**”.**

The organizational aspect of ESACake is arranged around projects. This means that the main unit with which the users will work is the software project. There is no possibility of working with the application without creating a project and developing it in terms of the ESA methodology for software engineering.

Inside a software project, ESACake allows the user to define any number of user requirements, software requirements, architectural components and detailed components, as well as all the relations established between them. This point gives an idea of the application's organizational model. ESACake establishes that one user can work in one or more projects, each of these projects can be composed of one or more users and, as previously mentioned, the four main phases of the ESA methodology with their corresponding elements are developed in each project.

Concerning ESACake's visual aspect, all the information is visually organized in the form of trees. In every page where it is necessary to show requirements (user or software requirements), components (architectural or detailed components), ontology terms, traceability matrices or search results, hierarchical trees are used.



The interaction between users and this kind of visual representation is realized as follows. If a tree node is selected, then all the information pertinent to that node is shown in the same web page, allowing the user to see all the information about an element without seeing the rest of the tree containing all the elements of the current phase.

Finally, ESACake has printing functions, that is, users can export their work to a printable format, like PDF or RTF, this format being editable. The user can configure the characteristics of the document (size, kind and colour of the fonts, titles, tables, etc.). The internal format of these documents follows the ESA methodology specification.

5. ESACake Use Case Scenarios

One typical scenario of ESACake use is as follows:

A team comprised of analysts and designers undertake a new software project, and decide to use the ESA methodology for software engineering.

The development team has decided to work with ESACake, a software engineering tool that is installed in a company's web server and has been used in other projects in the past. As it is a web tool, each team component can access his/her work (and other team member's) from any computer that has an internet connection, whether it be in the office, at home or traveling by plane; without installing in every single computer needed.

Initially they will interview the client to determine what his expectations and wishes are with regard to the application or the system that is going to be built. These interviews with the client plus the experience of the development team will result in the user requirements document.

These user requirements will be created using ESACake. This will allow the team to associate each requirement with different terms of, for example, a domain ontology pertaining to e-commerce (supposing the project they are developing is related to e-commerce) and relate each requirement with other user requirements with any points in common. These relationships between requirements allow them to know which requirements could be affected by the modifications made in another user requirement. This avoids a loss of time when reviewing all requirements in order to determine errors.

Once the user requirements capture and analysis phase ends, the software requirements phase starts. Using the ESACake tool, the development team defines the software requirements based on the user requirements obtained in the previous phase. ESACake allows the definition of relationships between the user requirements and the newly created software requirements. The traceability matrix between user requirements and software requirements is automatically generated.

In the rest of the project's phases, designers can use ESACake for organizing the system components, the diagrams of these components and the relationships between the components and the software requirements they are derived from. Similarly to the rest of the phases, they can establish the relations between the components and the domain ontology. They can relate, for example, the component responsible for carrying out credit card transactions with the term named *Transaction/Credit card*.

At any point during the project, the development team can use ESACake printer to generate PDF or RTF documents in order to give them to the client or any other member of the company. The development team can also allow the client or any other interested individuals to access the application with the user name *observer*. The user name *observer* has permission to consult information about the project, but cannot edit nor delete data.

Once the project terminates, the team will generate the required documents using ESACake. These documents will complement the rest of the documentation generated in the different phases of the project. At this point, the analysis and design phase is followed by the coding phase.

It is the coding phase which displays the most promising part of the application, since future analysts and design teams will be able to use ESACake carry out successful projects with the semantic database at their disposal. In other words, if a new team has to develop an application for another company which requires an online shopping module, they can benefit from using ESACake's search engine in order to look up requirements and components related to E-commerce; for example, credit card transactions. Thus, knowledge about the requirements and components of the first team's project can be re-used in the design of the current application.

6. Related Work

There are many applications that are similar to ESACake in the way that they are designed for helping analysts, designers and developers in creating software products. These kinds of applications are divided into two big groups: CASE (Computer Aided Software Engineering) and CAKE (Computer Aided Knowledgement Enviroment) tools.

- CASE tools are applications aimed towards increasing productivity of software development, reducing costs in terms of time and money.
- CAKE tools are applications and methodologies for identifying, classifying, retrieving, organizing, managing and reusing knowledge.

There is an application that is more similar to ESACake than the others, because of its orientation towards software reuse and software engineering processes (although it is not based in ESA methodology). This application is The Reuse Company's swReuser suite[2] [3].

The mayor features of swReuser are:

- UML support
- Knowledge reuse system
- Colaborative working
- Requirements support
- Risks management
- Project estimation technique
- Design Patterns
- UML model comparison
- Code generation
- Test case management
- Integrated forum system
- Trazability management

There are many differences between swReuser and ESACake: ESACake is a web application, ESACake gives support to ESA standard methodology for software engineering, swReuser suite includes an UML CASE tool, etc.

swReuser is a very mature tool and includes a lot of tools and features, but ESACake is focused on other

[2] http://www.reusecompany.com/swREUSER.aspx

aspects, improving knowledge reuse and giving complete support to ESA methodology (with the exception of code development phases).

7. Conclusions and Future Work

As said before, ESACake can help analysts and designers of software products to improve their experience of software engineering with ESA methdology. Due to the possibility of creating requirements and components, relating them, automatically generating traceability matrices, and exporting the information to PDF or RTF documents, etc. we believe that ESACake will make the work of analysts and designers teams easier. Furthermore, ESACake organizes all the projects' elements, the result being that making modifications, version control, requirement and component monitoring, etc. becomes more efficient and easy.

The strongest point of ESACake is the possibility of knowledge reuse. This knowledge can come from other people in the company, or past projects of the same team, thus reducing time in redoing existing work or searching through lots of documents and files.

For these reasons, ESACake can improve the user experience when addressing the tasks that are fundamental in ESA methodology. These tasks, at the same time, improve the quality for the analysis, design and development processes, eliminating errors and risks, and the final product.

ESACake is still in an early phase of its life and therefore there are many possibilities for improving it, both aspects relative to its characteristics, and security and efficiency issues.

There are a number of projects where the use of ESACake could optimize the implementation issues such as [5]. Furthermore, non-functional properties such as security could also be enhanced by means of semantics and particularly, by means of the approaches discussed in [6]. Some of the possible improvements, with respect to security, are based in introducing secure protocols in a way that the communications between clients and server will be secure, avoiding the possibility of unauthorized people having access to the information. Another security improvement focuses on data stored in the databases, for avoiding important losses of information caused by any possible application errors.

Regarding improvements in the fundamental characteristics of ESACake, these are some ideas: support for other methodologies (for example Metrica), including all the sections of each ESA methodology phase, etc.

8. Acknowledgements

This work is supported by the Spanish Ministry of Industry, Tourism, and Commerce under the project GODO (FIT-340000-2007-134), under the PIBES project of the Spanish Committee of Education & Science (TEC2006-12365-C02-01) and the MID-CBR project of the Spanish Committee of Education & Science (TIN2006-15140-C03-02).

9. References

[1] Ismael Rivera, Myriam Mencke, Juan Miguel Gómez, Giner Alor Hernández, Angel García Crespo. A Collaborative Open Social Network Dataset based on Email Ranking and Filtering. Proceedings of the 3rd IEEE International Conference on Systems (ICONS). Cancún, Mexico. April, 13 18. 2008.

[2] Ismael Rivera, Myriam Mencke, Juan Miguel Gómez, Giner Alor Hernández, Angel García Crespo. SmartWorld: More than Meets the Eye in Enterprise Application Integration.
Proceedings of the 3rd IEEE International Conference on Systems (ICONS). Cancún, Mexico. April, 13 18. 2008.

[3] Juan Miguel Gomez, Ricardo Colomo Palacios, Belen Ruiz Mezcua, Angel García Crespo: ProLink: A Semantics based Social Network for Software Project. In International Journal of Information Technology and Management. Special issue: Work Change in the Era of ICTs. 2007.

[4] Juan Miguel Gomez, Ricardo Colomo Palacios, Giner Alor Hernandez, Ruben Posada Gomez, Angel Garcia Crespo. Search in the Eye of the Beholder: Using the Personal Social Dataset and Ontology guided Input to Improve Web Search Efficiency. Proceedings of the 5th IEEE Latin American Web Conference (LA WEB07). Santiago de Chile, Chile. October, 31 November, 2nd. 2007.

[5] Juan Miguel Gomez, Jose Luis Lopez Cuadrado, Ioan Toma, Angel Garcia Crespo: A Policy Aware Knowledge Oriented Framework for Web Service Conversations. Proceedings of the IADIS International Conference. Vila Real, Portugal. October, 5 8. 2007.

[6] Mohammad M. R. Chowdhury, Josef Noll, Juan Miguel Gomez: Enabling Access Control and Privacy through Ontology. Proceedings of the IEEE 4th International Conference on Innovations in Information Technology, Innovations'07, November, 18 20, Dubai.

[7] http://jena.hpl.hp.com/wiki/SDB/Query_performance