Software Engineering 2.0: A Social Global Repository Based on Semantic Annotation and Social Web for Knowledge Management

Ricardo Colomo-Palacios[1], Juan Miguel Gómez-Berbís[1], Ángel García-Crespo[1], and Inmaculada Puebla-Sánchez[2]

[1] Universidad Carlos III de Madrid, Computer Science Department
Av. Universidad 30, Leganés, 28911, Madrid, Spain
{ricardo.colomo,juanmiguel.gomez,angelgarcia}@uc3m.es
[2] Universidad Francisco de Vitoria, Computer Science Department,
Ctra. Pozuelo-Majadahonda Km. 1.8, 28223 Pozuelo de Alarcón, Madrid, Spain
i.puebla.prof@ufv.es

Abstract. The effective management of the software development process has become an essential for business survival in an ever more competitive industry. In order to gain business strengths from the development process, organizations need to carry out software development in the most efficient manner possible, avoiding redundancy and time losses. This paper presents an architecture which combines the strengths of two technologies, Web 2.0 and the Semantic Web, as a solution to reuse and extrapolate knowledge and software products across projects and organizations.

Keywords: Software Engineering, Web 2.0, Semantic Web, Reuse, Knowledge extrapolation.

1 Introduction

The spread of Information Systems in organizational environments in recent years has turned their development into a critical task for corporations. In this setting, the crucial development process, as well as the large volumes of information which support this process, have meant that the management of the process, in the context of reutilization, extrapolation and transferability of Software Engineering (SE) elements, has become an essential research field. Additionally, the globalization of technologies, such as the Internet, and its subsequent reinvention as the Web 2.0 [1] have lead to a scenario where the possibilities for reuse and transfer of SE products are multiplied, and transcend organizational boundaries. Globalization and participation have opened up infinite opportunities for exploiting the capacities which a network of users can contribute to the software development process.

The current research is set within this background, and represents the fusion of some of the most important topics in knowledge management and knowledge reuse: the application of semantics and the integration of Web 2.0 elements. The present work proposes Social Global Repository (SGR), a tool created for the exploitation of the collective knowledge generated by software processes. The use of this knowledge is realized by the benefits gained from the combination of various aspects: firstly, the semantic annotation of the different products which are generated during the software development process. The second benefit is gained from the transferability between the products generated, and the last factor which is exploited is the social interaction of the users of the platform, inspired by their experiences with the products and their use of the products in projects.

2 Background

The term "Semantic Web" was coined by [2] to describe the evolution from a document-based web towards a new paradigm that includes data and information for computers to manipulate. Ontologies [3] are the technological cornerstones of the Semantic Web, because they provide structured vocabularies that describe a formal specification of a shared conceptualization. The fundamental aim of the Semantic Web is to answer the ever-growing need for data integration on the Web. It is precisely the integration of data on the Web which is the foundation that provides the starting point for the current research. Semantic Web provides a complementary vision as a knowledge management environment [4] that, in many cases has expanded and replaced previous knowledge management archetypes [5]. In other hand, Web 2.0 is seen as a new deal for software management [6].Particularly, in the SE domain, the capacities of the Semantic Web to be used as a Corpora of Reusable Contents [7] have been identified, and its potential uses for reutilization and transfer of knowledge in various environments have been established including experience management [8].

A specific example of the application of such technology is in the field of Requirements Engineering, where semantics has been used for diverse aspects such as how to apply the use of Semantic Wikis for the determination of requirements [9] or the application of semantics for Aspect-Oriented Requirements Engineering [10]. However, the efforts to integrate the Semantic Web and Web 2.0 have now gone beyond Requirements Engineering, including aspects such as the modeling of ontologies for the CMMi maturity model [11] of the software process [12], [13] or software maintenance [14]. In this specific field, which is focused on information reuse, extrapolation, and integration, in the context of software development projects, a number of initiatives have been launched to benefit from the capabilities brought about by the advent of the Semantic Web.

Possibly the most relevant initiative is the proposal to facilitate Software Reuse by searching the knowledge repository and suggesting relevant knowledge for the current task the user is performing [15]. Without a doubt, the initiative described in the current work is an innovative proposal, and which opens up new horizons for the possibilities brought about by the reuse of knowledge generated in Software Development projects. The functionalities of the tool presented in this work combine the benefits of search and organization of information offered by the Semantic Web, the transferability of the products generated by the SE process, and extend the functionalities to users by incorporating their participation using Web 2.0 tools.

3 Social Global Repository (SGR)

In our particular case, the breakthrough of adding semantic metadata to a Software Repository is the ability to enable automatic or semi-automatic sharing and discovery of a number of features. This approach is at risk of the so-called chick-en-egg problem of metadata. The provider of the service would request for a good reason, a good application or benefit, of providing the metadata. However, if the metadata is not generated, no application or value-added functionality can be achieved. Metadata is provided through the tagging system, which certainly constitutes an interesting development, since emerging folksonomies (a set of tags, useful in learning and knowledge environments [16]) are organically appearing, because a number of people are interested in particular information and are encouraged to describe it, being it rather than a centralized form of classification, a free bottom-up attempt to classify information [17]. Users are moving towards the concept of shallow ontologies which comprise relatively few unchanging terms that organize very large amounts of data, by using a set of very common and always-showing-up terms and relations.

This issue has loomed over recent sharing-oriented software projects and it is of the utmost importance for our approach. The lack of motivation and accuracy of efficiency from the user perspective in providing the metadata could hamper the SGR's full potential. However, a twofold strategy has been developed which overcomes the problem in the SGR approach:

1 Stakeholders of the SGR are gaining in terms of productivity and efficiency from the very first moment they provide metadata and use another stakeholder's metadata. The sharing of knowledge about software project elements enables a quid pro quo benefit situation as described in ProLink [18]. Particularly, the ever-changing nature of IT is the perfect growing field for various experiences that can very much help the lack of knowledge, background and expertise, by distributing the knowledge gained from these experiences across different infrastructures and environments. This can be achieved by means of sharing resources. Harnessing the potential spread of knowledge through a Social environment is not new, but must be leveraged with a technology that allows the determination of expertise that are hidden somewhere around the world wide web.

2 Metadata is also clearly creating the boundaries of sharing in organizations. There is a critical tradeoff associated with the tension between user privacy requirements, and providing persistent and increasingly broad visibility of their activities. Identity tradeoffs in community networks are even greater - in exchange for our privacy, we expect to gain a sense of security and well-being. The significance of adding privacy-enhancing technologies (PET) in virtual community networks is overwhelming [19]. In the SGR both premises are addressed since sharing of requirements and knowledge gains visibility (and it is used by a broad base of software projects stakeholders), while also protecting their privacy.

Integration of Software Requirements in the SGR conceptual framework through semantics is a growing and recognized challenge that can revolutionize IT working environments as we know them today. Nevertheless, it must rely on a consistent architecture.

Software architectures are becoming increasingly intelligent and interactive. By replacing locally managed hardcoded software structures, with an intelligent on-demand information paradigm, this model changes how business applications are delivered, bringing new levels of ease, adoption and success to the challenging area of Information Systems. In this section, we will discuss and depict the main components of our software sharing knowledge intensive platform. Conventional application architectures, at least those of interactive software applications supporting end users, have at least seven architectural layers [20]:

- Graphical User Interfaces (GUIs) in Web browsers
- User interface logic drivers
- Business Processes
- Business logic implementations
- Business rules constraining valid operations
- A persistence layer
- Storage systems for storing and recalling data

These seven layers execute any successful user request on the GUI, and any response travels through them all on the way back to the GUI. That means fourteen layers back and forth. For our particular context, it is noteworthy that our platform will be using a "semantic" data representation as well as a data-interpretation model [20]. For the sake of simplicity, we have coupled and regrouped several of these seven layers as can be seen in Figure 1.

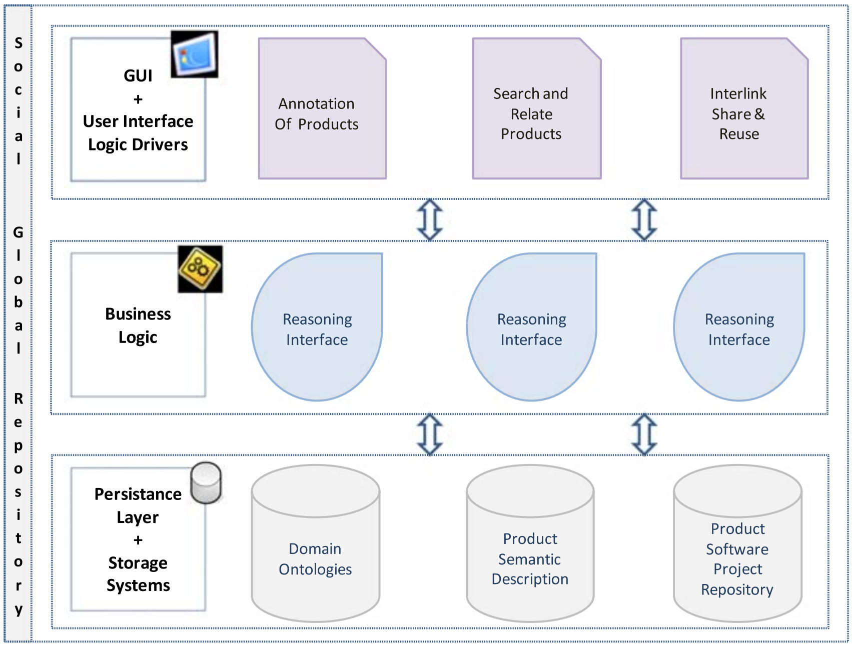
Social Global Repository
GUI + User Interface Logic Drivers
Annotation Of Products
Search and Relate Products
Interlink Share & Reuse
Business Logic
Reasoning Interface
Reasoning Interface
Reasoning Interface
Persistance Layer + Storage Systems
Domain Ontologies
Product Semantic Description
Product Software Project Repository

Fig. 1. SGR System Architecture

Particularly, we have grouped the first two into a presentation layer, which covers the annotation of software products, reuse, sharing and interlink, in addition to search, functionalities. Business processes and business logical implementations, together with business rules constraining valid operations are concentrated on the Business Logic layer. Finally, the persistence layer and storage systems for storing data are located in the last layer. Hence, we finally derive into the canonical three-tier architecture, due to the fact that we want to decouple the views, the business and the data access management. Each of these tiers will contain one or more subsystems.

In the following, we will present the different layers of the architecture, de-scribing the components belonging to each layer. Firstly, the User Interface and User Interface Driver layer is composed OF three components. The Annotation of Software Products component provides semantic annotation through visualization of the various semantic descriptions (and their underlying ontologies). Annotation is simply the adding of extra information asserted with a particular point in a document or other piece of information, in our case, semantic information. Secondly, the Search component, the core of the GUI provides extra functionality to find and relate software products from among the various software projects included in the SGR. The Search component is hence the entry-point to locate and retrieve software products from the whole platform. Finally, the Interlink, Share and Reuse component.

The Business Logic Layer is the added-value component of the platform. The Reasoner and Inference engine enables required reasoning capacities that would derive knowledge from the user queries and preferences related to the current semantic descriptions of a number of software products. Inference can intelligently match preferences of users and semantic descriptions for extracting new knowledge. The Business Rules engine validates if operations can be applied, and the Visibility Constraints refers to the tradeoff between public awareness and public concerns mentioned in the previous section, SGR, as discussed by [19].

The Persistence and Storage Systems layer enables data to perform the business process execution of the platform. There are three main components, namely: the Domain Ontologies, Semantic Descriptions and Software Project Repositories. Both the first and the second consist of a RDF (or another potential Semantic language, such as OWL, for that matter) semantic data store system that allows semantic querying, and offers a higher abstraction layer to enable fast storage and retrieval of large amounts of RDF while keeping a small footprint and a lightweight architecture approach. An example could be the OpenRDF Sesame RDF Storage system, which deal with data and legacy integration. Currently, we have focused for our implementation in RDF, given that the advantages of using RDF as a "lightweight" ontology language are supported by reliable implementations, software scalability and a mature base of developers and users.

In what follows, we focus on our proof-of-concept implementation, the SGR system which has been used for the management of a set of software projects, related to the European Software Agency (ESA) standard. SGR 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 SGR, the classes that define the application's behavior 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 accessing 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 SGR 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 persistence.

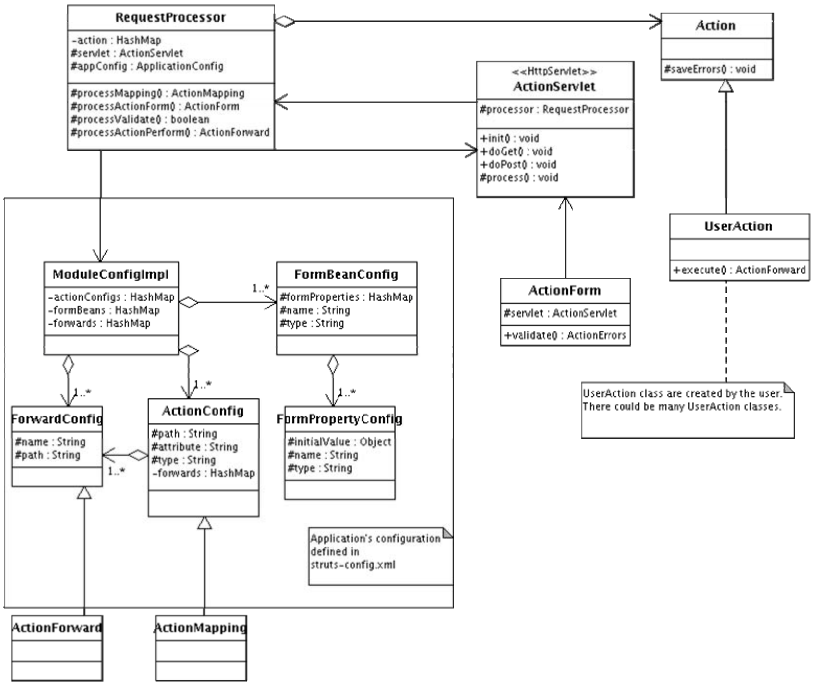

Fig. 2. SGR Conceptual Model

Jena has been used to provide semantics to SGR. 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 SGR'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 [21], 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 normalized so that the triple table is narrower and uses integers for RDF nodes, then does do joins to get the node representation. 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 SGR 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 SE. Inside a software project, SGR 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. SGR 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 SGR'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.

4 Conclusions

Integration of SE products and artifacts through semantics is a growing and recognized challenge that can revolutionize the application development environment as we know it. With the rise of the Semantic Web, the ontology-based approach to social networks has gained momentum. In such a context, sharing and taking advantage of a number of information sources, tracing products and artifacts, knowledge, experience and expertise in different contexts can bridge the gap of knowledge integration and product extrapolation and reuse. In this work, we have presented a novel approach to achieve knowledge extrapolation and software lifecycle products reuse across projects and organizations through a semantics-based social network, providing an architecture and a proof-of-concept implementation.

Our future work in application areas of the framework presented will focus on the extension of the system constructed, incorporating semantic descriptions of web services, which can be developed as an additional component of the future platform. This generates an extra software product for the user, which can be reused and transferred in the same way as User Requirements, Software Requirements, and the other products which comprise the Software Development process. Therefore, this new research has the objective of offering to users of the architecture the ability to integrate web services previously disconnected to the platform, as well as benefiting from complete documentation for projects, which will be generated during a standardized development process. This characteristic, which extends the concept of free software towards new horizons, would permit users to incorporate disintegrated web services, not only at the application level, but also as a fundamental part of the corporate development process.

References

1. O’Reilly, T.: What is web 2.0? O’Reilly NetWork (June 20, 2008), `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001)

3. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer, Berlin (2001)
4. Warren, P.: Knowledge Management and the Semantic Web: From Scenario to Technology. IEEE Intelligent Systems, 53–59 (January/February 2006)
5. Davies, J., Lytras, M., Sheth, A.P.: Semantic-Web-Based Knowledge Management. IEEE Internet Computing, 14–16 (September-October 2007)
6. Chatti, M.A., JArke, M., Frosch-Wilke, D.: The future of e-learning: a shift to knowledge networking and social software. International Journal of Knowledge and Learning 3(4/5), 404–420 (2007)
7. Tetlow, P., Pan, J.Z., Oberle, D., Wallace, E., Uschold, M., Kendall, E.: Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. W3C Working Draft (2006)
8. Mohamed, A.H., Lee, S.P., Salim, S.S.: An Ontology-Based Knowledge Model for Software Experience Management. International Journal of the Computer, the Internet and Management 14(3), 79–88 (2006)
9. Decker, B., Ras, E., Rech, J., Jaubert, P., Rieth, M.: Wiki-Based Stakeholder Participation in Requirements Engineering. IEEE Software 24(2), 28–35 (2007)
10. Chitchyan, R., Rashid, A., Rayson, P., Waters, R.: Semantics-Based Composition for Aspect-Oriented Requirements Engineering. In: Proceedings of the 6th international conference on Aspect-oriented software development, Vancouver, British Columbia, Canada (2007)
11. Capability maturity model integration (CMMI), version 1.1 CMMI for software engineering (CMMI-SW, v1.1) staged representation. Technical Report CMU/SEI-2002-TR-029, ESC/TR-2002-029, Carnegie Mellon, Software Engineering Institute, Pittsburgh (2002)
12. Liao, L., Qu, Y., Leung, H.: A software process ontology and its application. In: ISWC 2005 Workshop on Semantic Web Enabled Software Engineering (2005)
13. Soydan, G.H., Kokar, M.M.: An OWL Ontology for Representing the CMMI-SW Model. In: 2nd International Workshop on Semantic Web Enabled Software Engineering (2006)
14. Hyland-Wood, D., Carrington, D., Kaplan, S.: Toward a software maintenance methodology using semantic web techniques. In: Proceedings of Second International IEEE Workshop on Software Evolvability (2006)
15. Antunes, B., Seco, N., Gomes, P.: A Software Reuse System based on the Semantic Web. In: Proc. of the 3rd International Workshop on Semantic Web Enabled Software Engineering of the European Semantic Web Conference, Innsbruck, Austria (2007)
16. Lux, M., Dosinger, G.: From folksonomies to ontologies: employing wisdom of the crowds to serve learning purposes. International Journal of Knowledge and Learning 3(4/5), 515–528 (2007)
17. Shadbolt, N., Hall, W., Berners-Lee, T.: The Semantic Web revisited. IEEE Intelligent Systems 21(3), 96–101 (2006)
18. Gómez-Berbís, J.M., Colomo-Palacios, R., Ruiz-Mezcua, B., García-Crespo, A.: ProLink: A Semantics-based Social Network for Software Project. International Journal of Information Technology and Management 7(4), 392–405 (2008)
19. Chewar, C.M., McCrickard, D.S., Carroll, J.M.: Persistent virtual identity in community networks: Impact to social capital value chains. Technical Report TR-03-01 of Computer Science Dept. at Virginia Tech., http://eprints.cs.vt.edu/archive/00000650/01/hcic-cmc.pdf
20. Bussler, C.: Is Semantic Web Technology Taking the Wrong Turn? IEEE Internet Computing 12(1), 75–79 (2008)
21. Jena Project, http://jena.hpl.hp.com/wiki/SDB/Query_performance