OBFRE: An Ontology-Based Framework for Requirements Engineering

Verónica Castañeda[1], Luciana Ballejos[1], Ma. Laura Caliusco[2], Ma. Rosa Galli[2]

[1]CIDISI - UTN - Facultad Regional Santa Fe
Santa Fe, Argentina
vcastañeda@frsf.utn.edu.ar; lballejos@frsf.utn.edu.ar

[2]CIDISI - UTN - CONICET
Santa Fe, Argentina
{mlcaliusco, mrgalli}@santafe-conicet.gov.ar

Abstract. The nature of requirements engineering involves capturing domain knowledge from diverse sources, including stakeholders with their own interests and points of view. A different under-standing of involved concepts may lead to an ambiguous and incomplete specification and, thus, major rework after system implementation. Therefore, it is important to ensure that all participants in the requirements engineering phase have a shared understanding of the domain knowledge and of the elicited requirements themselves. In order to solve this problem, diverse ontology-based approaches were defined by different authors. The purpose of this paper is to present a framework for using ontologies in a comprehensive and integrated way.

Keywords. Requirements Engineering, Ontologies, Framework.

1 Introduction

With the advent of the Semantic Web and the technologies for its realization, the possibilities for applying ontologies as a means to define information and knowledge semantics become more and more accepted in different domains [1]. Recently, the use of ontologies in software engineering has gained popularity for two main reasons: (i) they facilitate the semantic interoperability and (ii) they facilitate machine reasoning.
Ontology can be defined as an explicit formal specification of how to represent the entity semantics that exists in a given domain and the relationships among them [2]. In general, for an ontology to be useful, it must represent a shared, agreed upon conceptualization.
There is an increasing amount of research devoted to utilizing ontologies in Software Engineering (SE) in general [3-7], and Requirements Engineering (RE) in particular [8]. In the latter, ontologies were used separately for describing the semantics of the

requirements specification documents, for the representation of requirements knowledge and for the representation of the requirements domain.
The main objective of this paper is to examine the synergy between ontologies and RE further and propose a framework for integrating these approaches comprehensively. Then, the contribution of this paper is twofold:

- The analysis and classification of the work already done in defining and using ontologies in RE.
- A framework for integrating and systematically applying the previous proposals in RE.

The remainder of the paper is structured as follows: Section 2 presents the main concepts related to Requirements Engineering and Ontological Engineering. Section 3 analyzes the benefits of applying ontologies in Requirements Engineering and presents a framework, called Ontology-Based Framework for Requirements Engineering (OBFRE), for integrating ontologies in Requirements Engineering. In Section 4, the OBFRE framework is instantiated with an example. Finally, in Section 5, the conclusions and future trends are discussed.

2 Requirements Engineering and Ontological Engineering

2.1 Software Engineering and Requirements Engineering

Software Engineering (SE) is an engineering discipline which compounds all aspects of software development [9]. Requirements Engineering (RE) is understood as a subtask of SE, which proposes methods and tools to facilitate the definition of all desired goals and functionalities of the software. Thus, system's requirements specify what the system must do (its functionalities) and its essential and desired properties.

The primary measure for an information system to be successful is the degree in which it meets the intended purpose. RE has the goal of discovering that purpose by identifying stakeholders and their needs, and documenting them for their future analysis, communication, and subsequent implementation [10].

Figure 1 shows an iterative cycle of core activities executed in RE [10]. All tasks presented in the figure generate diverse deliverables, in order to document obtained results along the RE process. There are diverse requirements specifications. They are mainly created in the "Requirements Representation" stage in Figure 1. These specifications are generally complementary, and very difficult to define. Thus, software engineers often have to deal with the necessity to redesign and iterate the process shown in Figure 1, due to the lack of information and differences in interpretation [11].

Diverse other challenges must be faced during RE activities in order to generate, at early stages of software development, consistent and complete requirements and to efficiently feed subsequent stages. One of those challenges is the management of participating organizations (through their stakeholders) in requirements gathering, considering the frequent lack of technical knowledge. Therefore, effective tools must be provided to achieve a complete analysis, considering particular and general needs and to manage requirements as a complete collaborative process [12].

Moreover, in RE processes there is a continual need for efficiently managing the great volume of information and knowledge generated and used during all activities presented in Figure 1. Thus, ambiguous requirements must be minimized since they produce waste of time and repeated work. They arise, for example, when different stakeholders produce different interpretations for the same requirement during the "Requirements Analysis" activity.

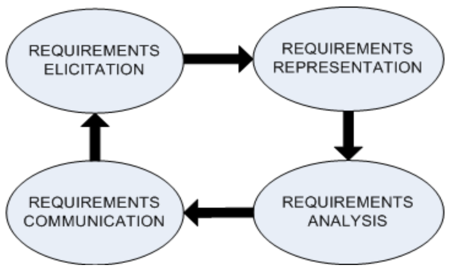


Fig.1. Requirements Engineering Activities.

2.1.1. Problems in Requirements Engineering

Information systems' requirements process should result in the establishment of well-defined functionalities and attributes agreed by the stakeholders. Nevertheless, if the specifications are incomplete or incorrect, the software may not meet users' expectations. Factors that could lead to inadequate results of requirements management process according to Wiegers [13] can be:

1. *Ambiguous Requirements*: which produce lost of time and repeated work. Their origin resides in stakeholders, who produce different interpretations of the same requirement. Moreover, one stakeholder can interpret the same requirement in diverse ways. Thus, the ambiguity conduces to unverifiable requirements and mistaken product tests.

2. *Insufficient Specifications*: they produce the absence of key requirements. This conduces to developers' frustration, because they base their work in incorrect suppositions and, so, the required product is not developed or desired requirements are not satisfied, which displeases the clients.

3. *Requirements not completely defined*: they make impossible the project secure planning and monitoring. Poorly understood requirements leads to optimistic estimations, which return against when agreed limits are surpassed. They also lead to unsatisfied requirements.

4. *Dynamic and changing requirements*: which require constant requirements revision and change in order to help integrating them and understanding new clients needs, identifying how they can be satisfied.

2.2 Semantic Web Technologies

2.2.1. Ontology: Definition and Classification

In computer science, an ontology is understood as a representational artifact for specifying the semantics or meaning about the information or knowledge in a certain domain in a structured form [14]. An ontology is used to reason about the properties of that domain, and might be used to describe the domain.

Ontologies can be classified according to the task they are meant to fulfill [15]. Knowledge representation ontologies describe the modeling primitives applicable for knowledge formalization. Top-level ontologies, also called upper-level ontologies, try to comprehensively capture knowledge about the world in general, describing for example: space, time, object, event or action, and so forth, independently of a particular domain. Domain ontologies and task ontologies contain reusable vocabularies with their relations describing a specific domain or activity. They can specialize the terms of top-level ontologies.

2.2.2. Ontology Development Methodologies

Several methodologies for developing ontologies have been described during the last decade [16, 17]. The objective of these methodologies is to define a strategy for identifying key concepts in a given domain, their properties and the relationships between them; identifying natural language terms to refer to such concepts, relations and attributes; and structuring domain knowledge into explicit conceptual models.

In literature, two groups of methodologies can be figured out. The first one is the group of experience-based methodologies represented by the Grüninger and Fox methodology defined in the TOVE project [18] and by the Uschold and King methodology based on the experience of developing the Enterprise Ontology [19]. The second one is the group of methodologies that propose a set of activities to develop ontologies based on their life cycle and the prototype refinement, such as the METHONTOLOGY methodology [15], the Ontology Development 101 Method [20] and the methodology defined by Brusa et al. [21]. Usually, the first group of methodologies is appropriate when the ontology purposes and requirements are clear, while the second group is useful when the environment is dynamic and difficult to understand, and the objectives are not clear from the beginning [22].

2.2.3. Ontology Representation Languages

Different languages exist for ontology representation in a machine-interpretable way. Ontology languages are usually declarative languages commonly based on either first-order logic or description logic. The ones based on first-order logic have higher expressive power, but computational properties such as decidability are not always achieved due to the complexity of reasoning [23]. The most popular language based on description logic is OWL DL, which have attractive and well-understood computational properties [24].

Another relevant language for representing ontology in a machine interpretable way is the Resource Description Framework (RDF) [25]. RDF was originally meant to represent metadata about web resources, but it can also be used to link information stored in any information source with semantics defined in an ontology. The RDF data model is similar to classic conceptual modeling approaches such as Entity-Relationship or Class diagrams, as it is based upon the idea of making statements about resources in the form of subject-predicate-object expressions. These expressions are known as triplets in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one way to represent the notion "The stakeholders define the requirements" in RDF is as the triplet: a subject denoting "the stakeholders", a predicate denoting "define", and an object denoting "requirements".

3 OBFRE: An Ontology-Based Framework for RE

The potential uses of ontologies in RE include the representation of: (i) the requirements model, imposing and enabling a particular paradigmatic way of structuring requirements, (ii) the semantics of structures and documents for describing requirements and domain elements, and (iii) the knowledge of the application domain. Then, three different ontologies can be defined:

- *Requirements Ontology*. The Requirement specifications are the descriptions of the desired software characteristics specified by the customers. This model can be defined using an upper-level ontology. This ontology can be used during the elicitation process to reduce ambiguous requirements and avoid incomplete requirements definitions. Moreover, restrictions about requirements can be defined in this ontology, since they help in requirements validation and verification. Diverse proposals exist which describe requirements ontologies [8, 16, 26-33].

- *Requirements Specification Document Ontology*. Different approaches are used in RE as intermediate steps for modeling and obtaining requirements. One of these approaches is the scenario technique [34], which proposes the generation of exemplary descriptions of the usage of the planned system to reach a defined goal. The use of ontologies for describing the structure of requirements documents reduce, between others, insufficient requirements specifications. Furthermore, they can greatly help in the definition of several structures for showing the same knowledge, in order to, for example, involve several stakeholders in the validation and analysis of the elicited requirements. Moreover, they can also help in reusing structured representation for diverse objectives or projects, only changing their content.

- *Application Domain Ontology.* Represents the application domain knowledge and business information required for developing software applications for a specific domain. It also includes the semantic relationships established among their concepts from a real-world point of view. Application domain ontology is useful to identify dynamic and changing requirements since it helps in understanding the domain and considering it through other stages of software development.

With the aim of improving the requirements definition, the ontologies previously described can be used for semantically annotating a requirements specification document. Figure 2 shows a framework, called OBFRE, which depicts the interrelations between the ontologies and a requirement specification document. The arrows between the requirements specification document and the ontologies represent an *instance-of* relation. The Requirements Specification Document Ontology (O_{REDO}) is related with the Requirements Ontology (O_{RO}) by the *DefinedBy* relation, indicating that the requirements defined in the document follow the semantics of the O_{RO}. This allows obtaining answers to questions involving them, such as: In which documents are represented certain requirements? or how is represented certain requirement?, or which requirements are described in certain document?, among others.

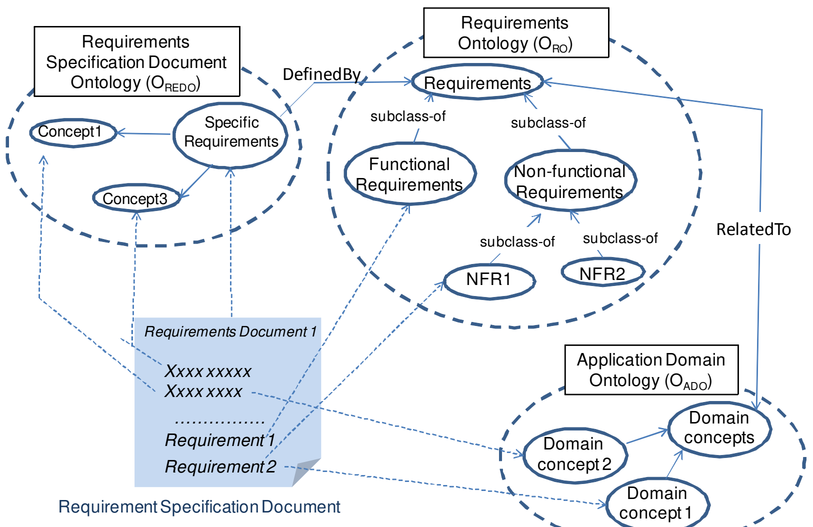


Fig. 2. Ontology-based framework for supporting semantics based Requirements Engineering.

In addition, the requirements are related with the domain concepts by the *RelatedTo* relation. This allows to obtain answers to questions involving them, such as: Which requirements are related to certain domain concept?, or which domain concepts are included or consider certain defined requirement?, among others.

Finally, a unique ontology composed by the O_{REDO}, O_{RO} and O_{ADO} ontologies is generated by integrating the results of the *RelatedTo* and the *DefinedBy* relations. This ontology will be later used for semantically annotating the requirements specification documents.

In order to implement the OBFRE Framework two decisions have to be made. One decision is related to the tool for implementing the semantic annotations, and the other one is related to the ontologies and the editor to be used. A survey on semantic annotation tools can be found in Reeve and Han [35]. Regarding ontology editors, the most popular one is Protégé-OWL that supports the Web Ontology Language (OWL).

Following, ontologies for representing knowledge from each of the areas mentioned above are described. Then, a research made towards ontologies in RE is resumed.

3.1 Ontologies in RE

3.1.1. Ontologies for Describing Requirements Specification Documents

A well-characterized requirements specification is important to the design stage of software development and to the evaluation and reuse of elicited requirements. Specifications are formed of both, the document structure and its content. In this sense, Groza et al. [36] affirm that the structure of a document has a very important influence in the perception of its content.

Reuse is one of the most required features for any software product or deliverable. It is based on the form in which requirements are specified, documented and structured. Nevertheless, the reuse faces several challenges. These challenges are caused by insufficient support for its steps, such as search, evaluation and adaptation. One way of exchanging reusable requirements specification documents is through Wiki systems, which allow the self-organized reuse since the community provides and organizes the artifacts to be reused [33].

The analysis of Wikis as solutions in this area is a very novel approach. The proposals conclude that requirements specification documents can specially benefit from ontologies, moreover when the content of those documents grows in a chaotic way. One way of solving this issue is structuring the knowledge by enriching the documents with additional metadata. This allows finding interrelated useful content and adding semantics to the documents through the generation of wiki extensions with RDF. In this way, the semantic is expressed in a machine-understandable format. This solution is known as Semantic Wiki and can be considered as a lightweight platform.

Furthermore, reuse cannot be possible if requirements documents do not have two main attributes carefully balanced, as described by Hull et al. [37]: *readability* and *processability*. They both can be greatly enhanced by the use of ontologies in requirements documentation. One clear example is adapted by Decker [33] from the Use Case approach. They add diverse documents and new structures to the traditional Use Cases documentation. These new documents are known as templates and allow capturing knowledge. Each one has metadata, besides the ontology of the documents. The authors also allow the extension of the ontology linking different Use Cases to facilitate the search of documents of the same type with other projects.

Another approach using templates is proposed by Groza et al. [36]. They describe a solution for generating different representations of the same document, known as templates, based on the metadata created. They use a particular authoring and annotation framework. Proposals like this can be of great help in order to represent RE specifications structures, thus promoting the reuse of RE specifications content allowing diverse structures representations.

The use of ontologies helps stakeholders to clarify their information needs and comes up with semantic representations of documents. For example, Dragoni et al. [38] present an approach for the ontological representation and retrieval of documents and queries for Information Retrieval Systems, using a vector space model which uses concepts instead of terms, where the documents are represented in a conceptual way, and the importance of each concept is numerically calculated.

All these approaches can be, in some way, integrated in order to define an ontology for representing RE documents structures, and so, promoting the adaptation of the same content in diverse formats in order to be understandable by all stakeholders. Moreover, an ontology with this goal can be reutilized in diverse projects in order to structure knowledge proper for each one.

3.1.2. Ontologies for Formally Representing Requirements

The use of ontologies for the representation of requirements knowledge has been under study since a long time ago. One of the initial approaches in this area was presented by [27]. They propose a generic solution that provides an unambiguous, precise, reusable and easy to extend terminology with dependencies and relationships among captured and stored requirements. The proposal can be applied to any kind of product in order to reach diverse requirements: communication, traceability, completeness, and consistency. It also supports the detection of redundant or conflicting requirements. The developed ontology is implemented using Prolog. The authors propose the use of first order logic to identify the axioms and capture the definition, constraints and relationships among the objects. They also allow integrity checking of the design knowledge. Besides being a very complete proposal, one of its disadvantages is that the involved terminology is only shared by the engineers of the project, and thus, the customer is not aware of it. This way, some requirements might stand ambiguous.

The relationships among captured and stored requirements help in defining the traceability of the RE process. Traceability is the ability to describe and follow the life of software artifacts in Software Engineering [39]. More specifically in RE, those artifacts are the requirements and specifications where they are documented. Thus, in order to trace requirements to their sources and to the intermediary and final artifacts generated from them all over the development process, it is mandatory to consider and represent information related to their source and the requirement's history.

Traceability also facilitates the reuse of requirements and their related information. In this sense, Veres et al. [31] define diverse requirements models and give rules for the mapping and traceability among them in order to facilitate requirements' reuse. Meanwhile, Lasheras et al. [26] pointed out the importance of requirement reuse since it improves the productivity and quality of software process. They also propose an ontology-based framework in order to check cycles and inconsistencies in requirements' traceability. It is used for modeling security requirements in risk analysis, based on two ontologies: the risk analysis ontology and the requirements ontology. The first one conceptualizes the risk analysis domain and the other one models reusable requirements, with their meta-information and relationships. The combination of both ontologies will enable the specification of security requirements with all their meta-information, relationships and semantic constraints.

Following the need of security requirements, Mouratidis et al. [28] extended the Tropos ontology towards three main directions: firstly, they introduced security constraint concepts. Secondly, existing concepts such as goals, tasks and resources must be defined with and without security in mind. Thirdly, security-engineering concepts such as security features, protection objectives, security mechanisms and threats must be introduced as well. This extension allows performing a formal analysis of the introduced concepts and thus, it provides formalism to their approach. In addition, by considering the overall software development process, it is easy to identify security requirements at early requirements stage and propagate them until the implementation stage.

Also Decker et al. [33] promote reuse by establishing a common requirements structure to be considered along SE activities. This is related to which Brewster et al. [40] affirm, that to build systems in order to solve real-world tasks, not only conceptualizations must be specified, but also, clarity over the problem solving must be given. In this way, Riechert et al. [29] present a semantic structure for capturing requirements relevant information, in order to support the RE process semantically and to promote the collaboration of all stakeholders in software development processes. They also apply and evaluate the proposal in an e-government case study.

The KAOS (from Knowledge Acquisition in autOmated Specification) methodology is a goal-oriented requirements engineering approach with a rich set of formal analysis techniques. It is described as a multiparadigm framework which allows combining different expression and reasoning levels: semi-formal for modeling and structuring goals, qualitative for selection among the alternatives, and formal when needed for more accurate reasoning [32].

All goal-oriented approaches are more applicable for complex systems. They are commonly based on the not easy task of identifying goals. Then, nonfunctional requirements (NFRs) are derived from them. Their analysis and management is much more difficult than the functional ones. As a more specific approach for using ontologies for representing NFRs knowledge, Dobson and Sawyer [8] propose an ontology for representing dependability between requirements. It considers diverse NFRs, such as: availability, reliability, safety, integrity, maintainability, and confidentiality. Meanwhile, another proposal in this area is given by Kassab [30], who develops an ontology which provides the definition of the general concepts relevant to NFRs, without making reference to any particular domain. Through the proposed ontology he describes diverse glossaries and taxonomies for NFRs. The first ones are used for generalization to the common NFRs concepts.

Wouters et al. [16] point out that one of the biggest problems in reusing use cases was to find similar or related ones when considering the importance of knowledge reuse and its application in RE. Thus, and in order to accomplish the reuse, they propose a semiformal description which, used together with a natural language format, can make possible the reuse of use cases. The defined ontology has three categories of information: labels, concepts and relations. Diverse rules and queries can be created with these concepts which, under a logic inference machine and together with algorithms, make it possible to find similar use cases.

3.1.3. Ontologies for Formally Representing Application Domain Knowledge
Domain ontologies are specific, high-level knowledge models underlying all things, concepts, and phenomena of a given domain of discourse. As with other models, ontologies do not represent the entire world of interest. Rather, ontologists select aspects of reality relevant to their task [41]. Then, the selection of the methodology to be used for developing an ontology depends on the application that ontologists have in mind and the extensions that they anticipate.

In software development, an ontology can be used at development time or at run time [42]. The use of an ontology during the development stage enables designers to practice a higher level of knowledge reuse than is usually the case in SE. At run time, an ontology may enable, for instance, the communication between software agents, or it may be used to support information integration. In both cases, the creation of the ontology starts with the RE process.

Any software development process implies multiple stakeholders which collaborate with a common goal. At development time, domain ontology can be used as a way of facilitating the understanding between stakeholders. Pohl [43] affirms that RE must elicit and understand the requirements from the relevant stakeholders and develop the requirements together with them. Thus, in order to maximize environment comprehension, a common understanding of the involved concepts must be carried out. This means, the requirements analysts should be endeavored and must work towards understanding the language used in the universe of discourse, to then initiate its modeling. A model of the environment represents the reality and considerably improves its comprehension.

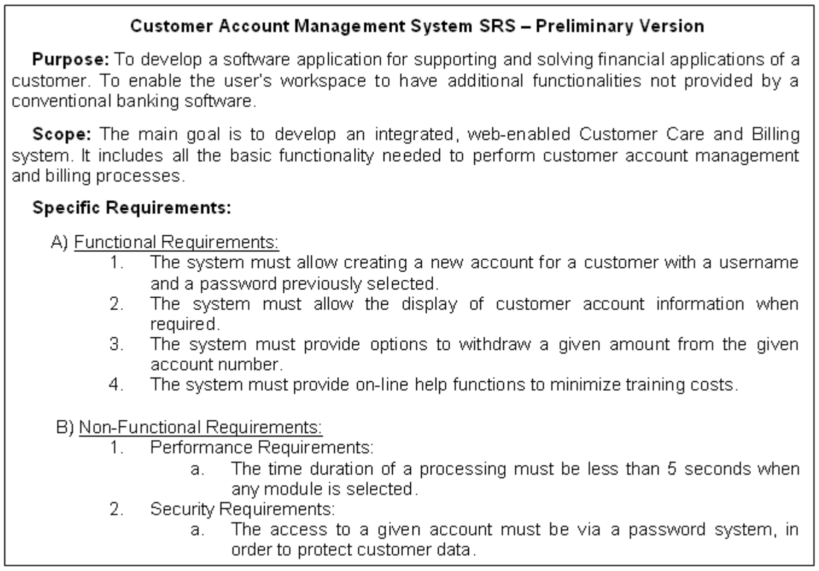
Thus, a crucial part of RE is the establishment of a common terminology by diverse stakeholders. To this aim, the methodologies described in Section II.B can be used at the first stage of the software development process. However, traditional methodologies for developing ontologies appear to be unusable in distributed and decentralized settings, and so, systems depending on them fail to cope with dynamic requirements of big or open user groups [44].

Breitman and Sampaio do Prado Leite [45] propose a process for building application ontologies during the requirements process based on the Language Extended Lexicon (LEL). The lexicon will provide systematization for the elicitation, model and analysis of ontology terms. The underlying philosophy of the lexicon resides in the contextualism category, according to which particularities of a system's context of use must be understood in detail before requirements can be derived. This approach is new to ontology building, which traditionally associates generalization and abstraction approaches to the information organization. Application ontologies are much more restricted than domain ontologies and have a much more modest objective. The authors see the ontology of a web application as a sub-product of the requirements engineering activity.

4 Applying the OBFRE Framework

In order to exemplify the application of OBFRE Framework, Figure 3 describes a preliminary version of a Software Requirements Specification (SRS) for a customer account management system. With the aim of describing semantically this document the Protégé editor was used. Figure 4 shows a portion of the resulting populated ontologies that show the specification of the third functional requirement and the security requirement. In this instantiation, the following ontologies were considered:

- Requirements Specification Document Ontology (O_{REDO}): this ontology was built considering the IEEE Recommended Practice for Software Requirements Specifications. In Figure 4, the elements that belong to this ontology are those whose prefix is "Oredo:".
- Requirements Ontology (O_{RO}): this ontology is an adaptation of the ontology proposed by Lasheras et al. [26] for representing the security requirements. In Figure 4, the elements that belong to this ontology are those whose prefix is "Oro:".
- Application Domain Ontology (O_{ADO}): In Figure 4, the elements that belong to this ontology are those whose prefix is "Oado:". The main challenge when building this ontology was the granularity level since the relation *RelatedTo* has to be established between instances or objects. As a result, the ontology is an upper-level ontology in which the main entities of the domain, such us *account*, are instances.



Customer Account Management System SRS – Preliminary Version

Purpose: To develop a software application for supporting and solving financial applications of a customer. To enable the user's workspace to have additional functionalities not provided by a conventional banking software.

Scope: The main goal is to develop an integrated, web-enabled Customer Care and Billing system. It includes all the basic functionality needed to perform customer account management and billing processes.

Specific Requirements:

A) Functional Requirements:
1. The system must allow creating a new account for a customer with a username and a password previously selected.
2. The system must allow the display of customer account information when required.
3. The system must provide options to withdraw a given amount from the given account number.
4. The system must provide on-line help functions to minimize training costs.

B) Non-Functional Requirements:
1. Performance Requirements:
   a. The time duration of a processing must be less than 5 seconds when any module is selected.
2. Security Requirements:
   a. The access to a given account must be via a password system, in order to protect customer data.

Fig. 3. Preliminary SRS version for Custumer Account Management System

Figure 4 shows how the requirements are related among them and with the domain concepts. For example, the functional requirement instantiated as *FunctionalRequirement 3* is related with the security requirement *SecurityRequierement1* through the domain concepts *FinancialAccount* and *Amount*. Thus, defining the requirements by using the OBFRE framework makes possible to trace dependencies among them, with their sources, and with the domain concepts.

5 Conclusions

The paper describes the diverse challenges that must be faced during RE activities. As mentioned before, RE involves several activities to generate consistent and complete requirements representation and specification. Nevertheless, due to the fact that stakeholders belong to different backgrounds, in addition to the great volume of information that must be managed, the need of a framework that helps in the whole process is noticeable.

The article also synthesizes diverse specific proposals based on ontologies, which were developed in order to help in diverse RE aspects. These proposals can be clearly divided into three application areas, such us: the description of requirements specification documents, the formal representation of the application domain knowledge, and the formal representation of requirements.

Although the approaches show an advance towards the demonstration of the importance of implementing technologies in certain circumstances and RE activities, more effort is still needed in order to generate an integrated framework, capable of addressing these challenges in an integrated way, and of being applied all over the RE process and its activities. This is even more important if the persistence of requirements in all the software development process is considered.

An integrated framework and its predominant characteristics were simply described in this paper. Once developed and implemented, it will be useful in requirements consistent management, specification, and knowledge representation activities during the entire software development projects.

Thus, future work will be focused on generating support for the framework in order to enhance and integrate requirements structure ontology generation, requirements content ontology generation and requirements domain ontology generation. This will allow and promote the collaboration of all stakeholders in requirements definition along all involved tasks, and moreover, to define a common structure and knowledge representation format, capable of being used in the entire software development process.

Acknowledgements. The authors are grateful to “Universidad Tecnológica Nacional (UTN)”, “Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)”, and “Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT)” for their financial support.

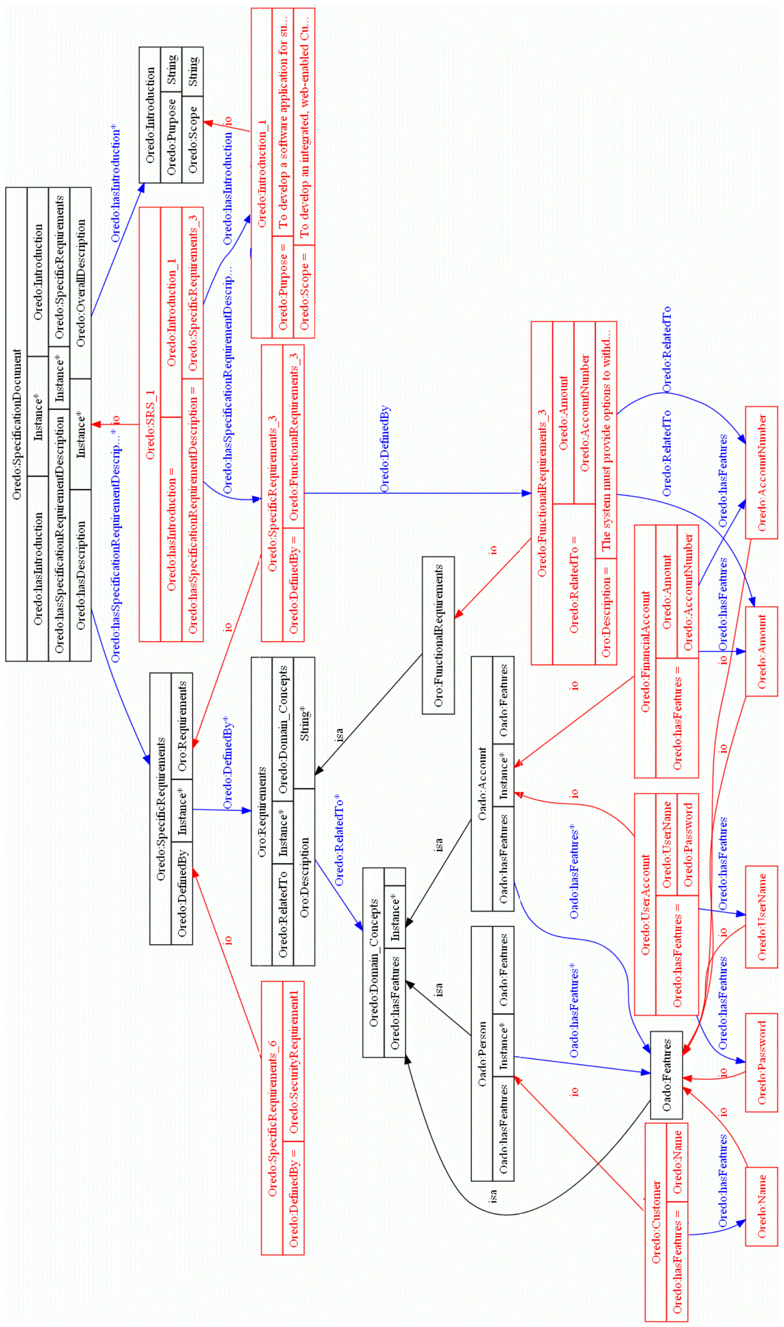


Fig. 4. A portion of the populated ontologies that belong to the OBFRE Framework.

References

1. Shadbolt, W.H., Berners-Lee, T. The semantic web revisited. IEEE Intelligent Systems, 21(3), pp. 96-101 (2006)
2. Smith, B., Kusnierczyk, W., Schober, D., Ceusters, W. Towards a reference terminology for ontology research and development in the biomedical domain. In: 2nd Int. Workshop on Formal Biomedical Knowledge Representation: "Biomedical Ontology in Action", pp. 57-66 (2006)
3. Happel, H.J., Seedorf, S. Applications of ontologies in software engineering. In: International Workshop on Semantic Web Enabled Software Engineering (SWESE'06) (2006)
4. Zhao, Y., Dong, J., Senior Member, IEEE, and Peng, T. Ontology Classification for Semantic-Web-Based Software Engineering (2010)
5. Eberhart, A., Argawal, S. SmartAPI - Associating ontologies and APIs for RAD. In: Proceedings of Modellierung (2004)
6. Ankolekar, A. Towards a Semantic Web of Community, Content and Interactions. PhD Thesis. School of Computer Science, Carnegie Mellon University (2005)
7. Dameron, O. Keeping modular and platform-independent software up-to-date: Benets from the semantic web. In: 8th International Protégé conference (2005)
8. Dobson, G., Sawyer, P. Revisiting ontology-based requirements engineering in the age of the semantic web. In: International Seminar on Dependable Requirements Engineering of Computerised Systems (2006)
9. Sommerville, I. Software engineering (5th ed.). Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA (1995)
10. Nuseibeh, B., Easterbrook, S. Requirements engineering: A roadmap. In: International Conference on Software Engineering - Conference on the Future of Software Engineering, pp. 35-46 (2000)
11. Noppen, J., van den Broek, P., Aksit, M. Imperfect requirements in software development. In P. Sawyer, B.P., (Eds.): Requirements Engineering for Software Quality - REFSQ 2007. LNCS 4542, Springer-Verlag Berlin Heidelberg, pp. 247-261 (2007)
12. Lang, M., Duggan, J. A tool to support collaborative software requirements management. Requirements Engineering Journal, 6(3), pp. 161-172 (2001)
13. Wiegers, K. Software Requirements. 2nd Edition. Microsoft Press. (2003)
14. Allemang, D., Hendler, J.A. Semantic web for the working ontologist: Modeling in RDF, RDFS and OWL. Elsevier, Amsterdam (2008)
15. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer, New York. USA (2004)
16. Wouters, B., Deridder, D., Van Paesschen, E. The use of ontologies as a backbone for use case management. In: Proceedings of European Conference on Object-Oriented Programming (2000)
17. Corcho, O., Fernández-López, M., Gómez Pérez, A. Methodologies, tools and languages for building ontologies: where is the meeting point? Data and Knowledge Engineering, 46(1), pp. 41-64 (2003)
18. Grüninger, M., Fox, M. Methodology for the design and evaluation of ontologies. In: Proceedings of the Workshop on Basic Ontological in Knowledge Sharing (1995)
19. Uschold, M. Building ontologies: towards a unified methodology. In: 16th Annual Conference of the British Computer Society Specialists Group on Expert Systems (1996)
20. Noy, N., McGuinness, D. Ontology development 101: A guide to creating your first ontology. Report KSL-01-05. Stanford Knowledge Systems Laboratory (2001)

21. Brusa, G., Caliusco, M.L., Chiotti, O. Towards ontological engineering: a process for building a domain ontology from scratch in public administration. Expert Systems: The Journal of Knowledge Engineering, 25(5), pp. 484-503 (2008)
22. Cristiani, M., Cuel, R. A comprehensive guideline for building a domain ontology from scratch. In: International Conference on Knowledge Management (I-KNOW'04) (2004)
23. Brachman, R., Levesque, H. Knowledge Representation and Reasoning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2008)
24. Smith, M., Welty, C., McGuiness, D. Owl web ontology language guide. Recommendation W3C 2(1) (2004)
25. Klyne, G., Carroll, J. Resource Description Framework (RDF): Concepts and Abstract Syntax. Available on-line: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-URI-reference (2004)
26. Lasheras, J., Valencia-García, R., Fernández-Breis, J., Toval, A. Modelling Reusable Security Requirements based on an Ontology Framework. Journal of Research and Practice in Information Technology, 41(2), pp. 119-133 (2009)
27. Lin, J., Fox, M., Bilgic, T. A requirement ontology for engineering design. Manuscript, Enterprise Integration Laboratory, University of Toronto (1996)
28. Mouratidis, H., Giorgini, P. and Manson, G. An ontology for modeling security: The Tropos approach. Knowledge- Based Intelligent Information and Engineering systems. Springer Berlin / Heidelberg (2003)
29. Riechert, T., Lauenroth, K., Lehmann, J., Auer, S. Towards semantic based requirements engineering. In: 7th International Conference on Knowledge Management (2007)
30. Kassab, M. Non-Functional Requirements – Modeling and Assessment. VDM Verlag Dr. Müller Aktiengesellschaf (2009)
31. Veres, C., Sampson, J., Bleistein, S., Cox, K., Verner, J. Using semantic technologies to enhance a requirements engineering approach for alignment of it with business strategy. In: International Conference on Complex, Intelligent and Software Intensive Systems (2009)
32. van Lamsweerde, A., Letier, E. From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. In: Radical Innovations of Software and Systems Engineering in the Future. Volume 2941 of Lecture Notes in Computer Sciences. Springer Berlin / Heidelberg 153- 166 (2004)
33. Decker, B., Rech, J., Klein, B., Hoecht, C. Selforganized reuse of software engineering knowledge supported by semantic wikis. In: Workshop on Semantic Web Enabled Software Engineering (SWESE) (2005)
34. Hadad, G., Doorn, J., Kaplan, G. Explicitar Requisitos del Software usando Escenarios. In: Workshop on Requirements Engineering (WER'09) (2009)
35. Reeve, L., Han, H. (2005): Survey of Semantic Annotation Platforms. In: Proceedings of the 2005 ACM Symposium on Applied Computing (2005)
36. Groza, T., Schutz, A., Handschuh, S. Salt: A semantic approach for generating document representations. In: ACM Symposium on Document Engineering, pp. 171-173 (2007)
37. Hull, M., Jackson, K., Dick, A. Requirements Engineering, Practitioner Series. Springer, New York. USA (2002)
38. Dragoni, M., da Costa Pereira, C., Tettamanzi, A. An ontological representation of documents and queries for information retrieval systems. In: 1st Italian Information Retrieval Workshop (IIR'10) (2010)
39. Winkler, S., von Pilgrim, J. A survey of traceability in requirements engineering and model-driven development. Software and Systems Modeling Theme Section (2009)

40. Brewster, C., O'Hara, K., Fuller, S., Wilks, Y., Franconi, E., Musen, M., Ellman, J., Buckingham Shum, S. Knowledge representation with ontologies: The present and future. IEEE Intelligent Systems, 19(1), pp. 72-81 (2004)
41. Devedzic, D. Understanding ontological engineering. Communications of the ACM, 45(4), pp. 136-144 (2002)
42. Fonseca, F. The double role of ontologies in information science. American Society for Information Science and Technology, 58(6), pp. 786-793 (2007)
43. Pohl, K. Requirements Engineering. Dpunkt Verlag (2007)
44. Davies, J., S.R., Warren, P. Semantic Web Technologies: Trends and research in ontology-based systems. John Wiley, England (2006)
45. Breitman, K.K., Sampaio do Prado Leite, J. Ontology as a requirements engineering product. In: 11th IEEE International Requirements Engineering Conference (2003)