AN ONTOLOGY-BASED APPROACH TO THE MODELLING OF COLLABORATIVE ENTERPRISE PROCESSES

Dynamic Managing of Functional Requirements

M. V. Hurtado, M. Noguera, M. L. Rodríguez, J. L. Garrido
Departamento de Lenguajes y Sistemas Informáticos, Universidad de Granada, E.T.S.I.I.T.
c/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain

Lawrence Chung
Department of Computer Science, University of Texas at Dallas, Richardson, Texas 75083, USA

Keywords: Ontology-based collaborative modelling, Enterprise modelling techniques.

Abstract: Enterprise models describe and analyze collaborative processes and provide stakeholders with a common view of requirements. A core challenge to tackle the management of collaborative business processes is the continuous translation between business requirements and the current collaborative process model of the involved enterprises. This model is constituted by multiple IT systems, resources, and human labour. This paper presents a novel approach to modelling business processes from the perspective of collaborative systems. The proposal consists of a multi-level design scheme based on ontologies for the description of complex collaborative systems. The use of this ontology-based framework enables machine reasoning which can be applied to automated or semi-automated control and propagation of changes in the functional requirements specification. Benefits related to integrating ontology-based models are also presented.

1 INTRODUCTION

Nowadays, collaborative systems need to be developed in order to operate in dynamic environments. This kind of systems supports users who perform flexible and creative tasks within defined business rules.

The development of models and mechanisms is mandatory in order to enable the specification of collaborative systems which can capture the increasingly dynamic nature of both intra and inter-enterprise processes. Enterprise modelling techniques aim at modelling the behavior and domain entities in order to identify the fundamental business principles of an organization (Ambler, 2003) (Jaekel, 2005). They are recognised for their value in describing complex organizational domains, but usually in an informal way. Therefore, the modelling methods need to be improved with a new approach to integrate methods and tools which are appropriate to enterprise modelling and to the management of change. Ontology based integration of methods and tools can solidly provide a more precise means to model collaborative enterprise processes (Gruninger, 2000).

This research work presents an ontology-based framework which aids software engineers in the description of domain requirements and their appropriate association with the collaborative model elements (by means of concepts and relationships among them) at different levels. From the perspective of collaborative business processes, an ontology-based framework contributes to the formal definition of business semantics in a common vocabulary and supports the modelling and analysis of functional requirements. Moreover, this framework enables the dynamic management of domain requirements for collaborative processes within and across enterprises.

The remainder of this paper is organized as follows. Section 2 shows the main motivations on the basis of the relevant issues to be addressed for an advanced description of collaborative systems. Section 3 introduces an ontology-based framework to modelling and analysing functional requirements

V. Hurtado M., Noguera M., L. Rodríguez M., L. Garrido J. and Chung L. (2007).
AN ONTOLOGY-BASED APPROACH TO THE MODELLING OF COLLABORATIVE ENTERPRISE PROCESSES - Dynamic Managing of Functional Requirements.
In *Proceedings of the Second International Conference on Evaluation of Novel Approaches to Software Engineering* , pages 87-94
DOI: 10.5220/0002587000870094
Copyright © SciTePress

of collaborative systems. Section 4 presents how the proposal adequately supports the dynamic management of the functional requirements satisfying desirable properties for a formal description. Finally, the conclusions and future work are given in Section 5.

2 MOTIVATIONS

Collaboration processes in today's global business environment are a critical issue to the innovation and creation of new business models. Collaborative business processes involve organizations, workgroups, applications, documents and different sources of information. Stakeholders need to be assisted with tools that help them to describe, verify, validate and share their perspective in both modelling processes and domain requirements capture.

Domain requirements are related to the properties and functionalities of the system to be designed. They are categorized into functional and non-functional. There are several approaches dealing specifically with non-functional requirements which support the elicitation, documentation, verification and validation of this requirements (Chung, 2006).However, functional requirements are usually given in natural language or using semi-formal notations (e.g. UML use case (OMG, 2003)) during the modelling process. The main difficulty of this method resides in the gaps between domain users and requirements engineers. This will be the main source of inconsistent and ambiguous requirements (Yuqin, 2006).

Conversely, requirements generated by different members in a collaborative process may use different terminology to specify their system views. Hence, the same term may be applied to different concepts and different terms can be used to designate the same entity.

As long as models are not described sharing a common terminology, there will not be appropriate instruments to support the exchange of information and to ensure certain properties (consistency, completeness, etc) in the description and management of functional requirements using different models. Recent research points to ontologies as an appropriate technology to solve this problem. An ontology is a formal description of objects and their properties, relationships, constraints, and behaviour (Gruber, 1995), (Guarino, 1995). It allows defining a common vocabulary for users who need to share viewpoints of each particular domain. Consequently, the use of an ontology-based method (Zhi, 2000), (Lu, 2000) focusing on representing domain concepts and relations among them, can be used to share both intra and cross-enterprises models.

Furthermore, requirements are also altered during the system design due to changes of the business process and rules, customers' objectives, etc.

Formalizing ontologies with standard languages, for instance OWL (Web Ontology Language) (Smith, 2004), machine reasoning can be applied to an automated or semi-automated control and propagation of the functional requirements changes. The underlying Description Logics (Baader, 2003) to the OWL language allows OWL-based reasoners to perform certain verification procedures such as consistency check, concept satisfiability, classification and realization (Sirin, 2006).

Therefore, an ontology-based approach provides mechanisms to propagate changes, and focuses on the evolutionary nature of Collaborative Business Processes.

3 FRAMEWORK FOR MODELLING OF COLLABORATIVE ENTERPRISE PROCESSES

We propose an ontology-based design scheme for the modelling and requirements analysis of collaborative system design. This ontology-based framework provides both terminology to specify different functional requirements visions and communication of system functions by means of a well-defined terminology, syntax and semantics.

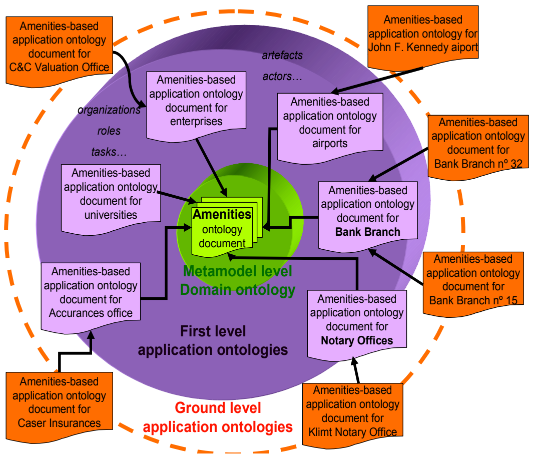
Amenities-based application ontology document for C&C Valuation Office
Amenities-based application ontology for John F. Kennedy aiport
artefacts
actors…
Amenities-based application ontology document for enterprises
organizations
roles
tasks…
Amenities-based application ontology document for airports
Amenities-based application ontology document for Bank Branch nº 32
Amenities-based application ontology document for universities
Amenities ontology document
Amenities-based application ontology document for Bank Branch
Metamodel level Domain ontology
Amenities-based application ontology document for Accurances office
Amenities-based application ontology document for Bank Branch nº 15
First level application ontologies
Amenities-based application ontology document for Notary Offices
Amenities-based application ontology document for Caser Insurances
Ground level application ontologies
Amenities-based application ontology document for Klimt Notary Office

Figure 1: Three-tier ontology design for collaborative systems.

At the highest abstraction level of the adopted design scheme (see Figure 1) for building the framework, a domain ontology defines the collaborative system terminology. Using this ontology, domain users and modellers can describe systems functions in terms of a common vocabulary (organizations, group, actor, role, capability, law, task, subactivity, information object, interaction protocol, etc.).

Figure 2 shows a diagrammatic representation of a domain ontology description, basically concepts and relationships between them, using the Protégé ontology editor (Knublauch,.2004).

The metamodel for the domain ontology is adopted from AMENITIES (Garrido, 2005a), a methodology for the study and development of collaborative systems. We define the meaning of the terminology using OWL which gives a precise and unambiguous semantics for each term.

The correctness and unambiguity avoid possible conflicts and diverse interpretations by different modellers. Accordingly, modellers use the same terminology and can work in the system specification promoting a participatory design.

At the subsequent level, based on the domain ontology, different application ontologies would appear in order to define specific elements for each particular business process (bank branch, valuation office, notary office, assurances office…), but with the adequate abstraction level so that they can be employed in similar or related systems (e.g. bank manager, head of risk and cashier roles).

At the lowest level, more specific entities of the system we are dealing with would be declared (e. g. “Anna Riemann” and “Donald Johnson” as actors present in the bank collaborative system of the Branch nº 15).As a leading example, we will consider the process of granting a mortgage in a bank branch. In this context collaborative systems can help a branch to offer a wider range of services. The corporate activity includes some collaborative tasks among different organizations (*Bank Branch*, *Valuation Office* and *Notary Office*)

Likewise, it is necessary that diverse actors belonging to different organizations are involved, although they are all part of the same group in charge of the granting process. Among others, bank actors can play the *bankManager*, *headOfRisk,* and *cashier* roles. Different banker tasks are considered in order to provide various banking services. During the process of granting a mortgage several information objects are handled. One of subactivity is called *calculate_leverage_coefficient* which is composed by others such as the subactivities *queryASNEF* and *queryRAI*, which obtain information contained in two databases: the ASNEF databases (banking database of possible non-fulfilment and their current situation) and RAI database (database referred to unpaid banking effects, bills of exchange etc.). As a result of the *calculate_leverage_coefficient* subactivity and together with additional information provided by the customer, the *headOfRisk* generates the mortgage report and suggests the approval, refusal or interruption of the operation. If the bank manager approves the operation, the collaborative task *agreeAndsign* will be carried out. A *connection_law* indicates that to perform that action, bank manager, notary and client must sign the mortgage document.

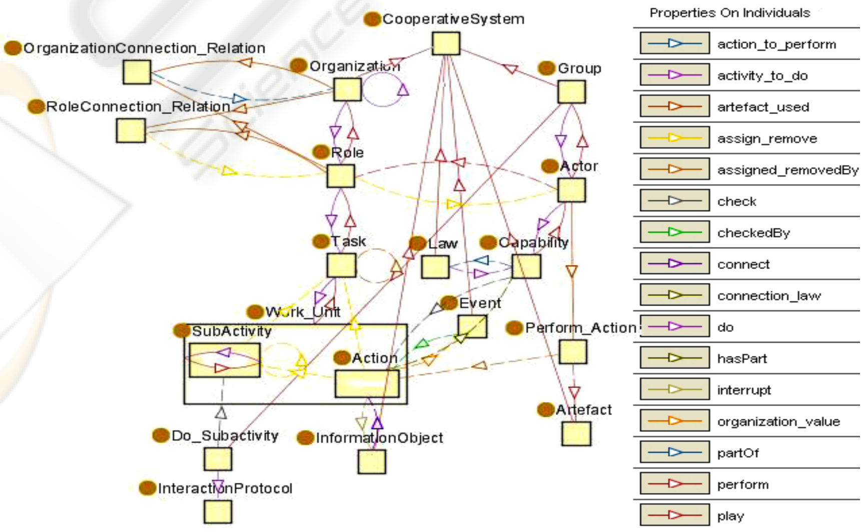

Figure 2: Diagrammatic representation of the AMENITIES metamodel description in OWL (domain ontology).

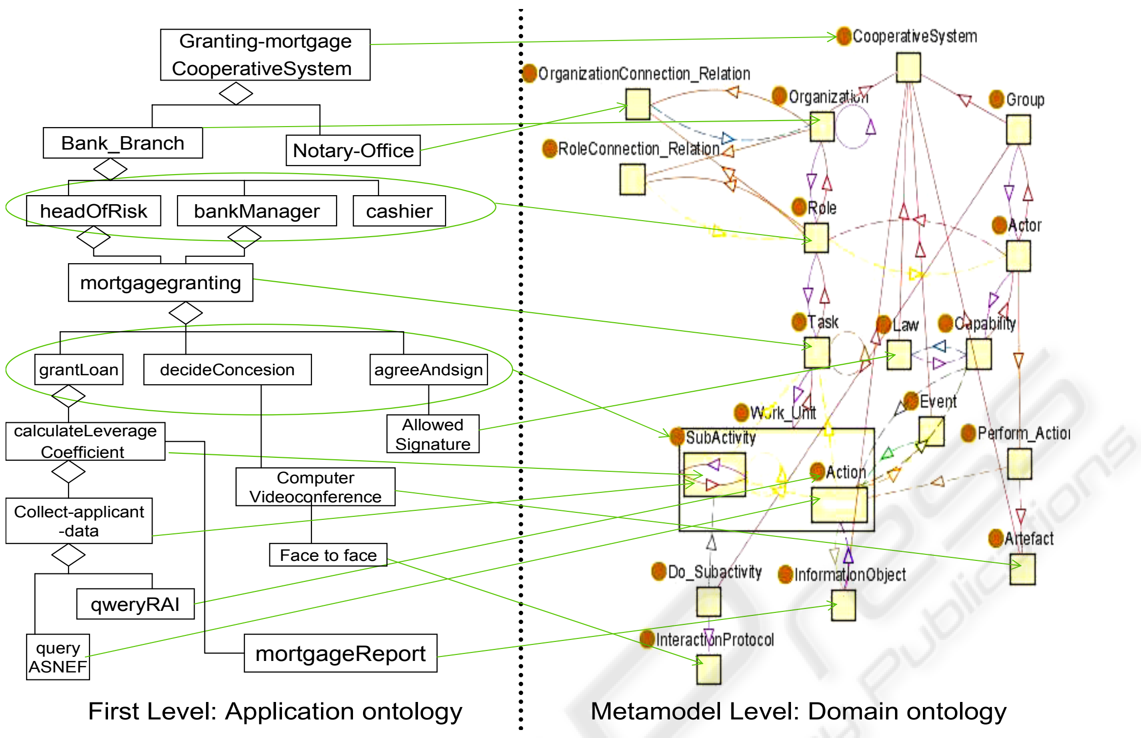

Figure 3: Mapping of some application ontology elements (Bank Branch Example) to metamodel elements.

4 DYNAMIC MANAGEMENT OF FUNCTIONAL REQUIREMENTS

The ontology based approach offers the potential to meet some needs of requirements management. These include abstraction mechanisms using different levels (see Figure 3), description for each term (concepts, properties, instances, as well as concept inheritance), inference mechanism, and model driven trace requirements capture (for example using Laws, pointing out the type InteractionProtocol…).

4.1 General Analysis

Collaborative business processes evolve with time. Hence, the system description is necessarily an iterative and dynamic process. The reasons for changes are inherent in the complexity of reality and in the limited ability of humans to cope with this complexity. Thus, the specification system must be able to change for a number of reasons, among others the following:

- The system specification often contains “design errors” and sometimes does not meet the requirements of its users.
- The business environment in which the system operates can change unpredictably, thereby invalidating the requirements made when the system was designed.
- Users’ requirements can change after the system is initially built, requiring that the existing specification evolve to meet the new requirements.

Since most of the changes usually affect local parts of the system or organization, it is compulsory that the changes are managed without affecting those elements that are unrelated to these changes and without being necessary to put them “out of service”. Likewise, appropriate tools and strategies for change propagation are needed.

The proposed three-tier ontology design allows us to manage systematically the modelling of the dynamic nature of the functional requirements for this kind of systems. The domain ontology and the application ontologies provide vocabularies and restrictions to help identifying and analysing system functions changes. Constraints capture statements that must be satisfied by design after change.

Table 1: Operations in the ontology-based scheme.

	Add	**Remove**	**Modify**
Concept	Add concept	Remove concept	Rename concept
Concept hierarchy	Add subConceptOf relationship	Remove subConceptOf relationship	Set subConceptOf relationship
Property	Add property	Remove property	Rename property
Property Domain	Add property domain	Remove property domain	Set property domain
Property Range	Add property range	Remove property range	Set property range
Instance	Add instance	Remove instance	Rename instance
Property Instance	Add property instance	Remove property instance	Set property instance

In addition, related to the dynamic aspects of the collaborative system, a different level can be used to answer many questions on the system behaviour, by deduction and using the reasoning logic.

In order to illustrate changes and their propagation (a change in one element of an ontology may have relevant consequences on other elements of the scheme-based ontology), the following situations will be considered for changes in:

- the relationships between instances and the terms of the application ontology at the ground level.
- the terms of the application ontology at the first level
- the structure of the domain ontology

Each of these changes can be carried out by one of the meta-change transformations: add, remove and modify ontology elements (Stojanovic,2002). As a result, a set of operations (Table 1) can be defined by the cross level of the set of entities of the ontology model, which form the ontology-based scheme, and the set of meta-operations.

A set of operations can be applied to an ontology in a valid state, and after all changes are performed, the ontology and dependant collaborative system must convert into another valid state. It means that every change is guaranteed to maintain the domain constraints.

According to software engineering, a set of properties for the specification has to be maintained for instance:

- Consistency – A consistent description system satisfies all invariants of the domain ontology model (Amenities ontology document). Invariants are constraints that must fulfill in every state of an ontology. For example, an organization has at least one role class.
- Validity – it is necessary to distinguish between syntax and semantic validity of an ontology. Syntax invalidity arises when undefined entities are used or model constraints are invalidated. Semantic invalidity arises when the meaning of an ontology entity is modified. Conversely, a valid instance adjusts to the constraint specified in the ontology document.

4.2 Detailed Examples of Control and Propagation in Requirements Changes

On the basis of the previously described case study (section 3), some particular examples of functional requirements changes are illustrated in this section.

System Behavior Properties

The most frequent changes are related with the dynamical aspects of the system which occur at the ground level. It allows us to specify functional requirements related to system behaviour, i.e. state changes in the described system.

See the example of "Anna Riemann" who was playing the cashier role changes to the headOfRisk role (Figure 4).

1.***Ground level*
is (*Anna Riemann* play *cashier)*?
TRUE=> (Remove_instance (*AnnaRiemann* play *cashier*) AND Add_instance (*AnnaRiemann* play *headOfRisk*)
FALSE => (Add_instance (AnnaRiemann play headOfRisk)
2.***Model level*
No-changes
3.***Metamodel level*
No-changes

Figure 4: Meta-operations and propagation (changes at ground level).

In the *Bank_Branch* organization can be considered the need of including a new role called proxy with the intention of realizing chief's tasks, hereby the *bankManager* is released from some tasks. This requirement modification takes place in the first level application ontology of the scheme description (Figure 5).This operation of adding the role would provoke changes in the following level because an actor system could play a proxy role (i.e. *Donald Johnson*) and, in the accomplishment of certain actions, the actor playing this role may replace the bank manager; this is the case for the task *signature-load*. Therefore, to realize the above mentioned subactivity, it would be necessary to modify the specification of the responsible roles in order to include the connexion between the proxy and the bank manager roles by means of an exclusive-or relationship.

1.***Model level*
Add concept (*proxy*)
AddsubConcept Of relationship (*proxy*, *Role*)
Add property (proxy.role, signature-load.task)-> *Actor* play *proxy* replace *Actor* play *bank-manager* AND check replace-bank-manager.Law
Rename signature-load.subactivity

2.***Ground level*
is (*Donald Johnson* instance of *Actor)*?
TRUE=> (Remove_instance (*Donald-Johnson* instance of *Actor*)) AND Add_instance (*Donald Johnson* play *proxy*)
FALSE=> (Add_instance (*Donald-Johnson* instance of *Actor*) AND Add instance (*Donald-Johnson* play *proxy*)
3.***Metamodel level*
No-changes

Figure 5: Meta-operations and propagation (changes at the model level).

Finally, there is another kind of less usual changes which take place at metamodel level. For example, for the highest abstraction level in the system description, a new way of arranging the work could be required, in particular, to have pending tasks classified according to different factors. In the branch context, such factors might be related to the type of asset transaction (credit, loan, mortgage,…), deadlines to be met, etc. In order to fulfil this new requirement, stakeholders could decide to add the new concept worklist in the domain ontology. This concept would be connected by an aggregation relationship with the concept task, The cardinality of relation would be of 0..n. This kind of change modifies the conceptual model. However, there is not propagation to the previously created application ontologies at the first and ground levels (Figure 6).

1.**Metamodel level
Add concept (*worklist*)
AddsubConcept Of relationship (*workist*, *Task*)
Add property (*worklist*, *cardinality*)
Add property range (*woklist.cardinalyty*, 1:n)
2.**Model level
No-changes
3.**Ground level
No-changes

Figure 6: Meta-operations and propagation (changes at metamodel level).

Another kind of change could be to remove one element of the domain ontology. For example, when the term subactivity is removed, the change involves modifications in order to guarantee the consistency of same and lower abstraction levels (application ontologies), each instance of the subactivity concept must be substituted with the set of final actions it includes.

Completeness Property

The separation between the model and the ground level ontologies may cause that design decisions be spread throughout both levels. The formalization of the system description using ontologies allows defining the appropriate restrictions so that completeness and consistency be preserved. For example, let's consider that Anna Riemann is integrated in the group that deals with the *agreeAndSign* at the ground level ontology.

At the model level, this activity for the signing of a mortgage is defined to be part of the roles bank manager, notary and client. In that case, it has to checked that Anna_Riemann plays one of the roles the activity *agreeAndSign* is part of or add (or remove) the necessary facts to the ground level ontology accordingly in order to preserve the completeness of the system (Figure 7).

```
1.**Metamodel level
   No-changes
2.**Model level  (present facts)
   SubActivity (agreeAndSign)
   Role (bankManager)
   Role (notary)
   Role (client)
   (agreeAndSign partOf client)
   (agreeAndSign partOf bankManager)
   (agreeAndSign partOf notary)
3.**Ground level (present facts)
   Actor (Anna_Riemann)
   Group (sign-Group)
   (sign-Group do agreeAndSign)
  **Ground level (new facts)
   Add_instance (Anna_Riemann partOf (sign-Group))
=>
 (Add_instance (Anna_Riemann play bankManager)
OR (Add instance (Anna_Riemman play notary))
OR (Add instance (Anna_Riemman play client))
OR (Remove_instance (Anna_Riemann partOf (sign-
Group))))
```

Figure 7: Detection of incompleteness and system response.

Consistency Property

Finally, as an example of consistency preservation let's think of the following scenario. At the model level, the bank the branch office belongs to has made a corporate decision by which every actor playing the role proxy must assume the task agree assigned to the bankManager when this one is absent, However, the branch office of the mortgage system has forbidden that any role may share the tasks assigned to the bankManager. This inconsistency will become apparent when trying to satisfy both restrictions.

```
1.**Metamodel level
   No-changes
2.**Model level
   (agreeAndSign partOf bankManager)
   is (bankManager status absent) == TRUE? =>
      ((exist ?actor play proxy) AND
        (Add_instance (agreeAndSign partOf proxy))
=> System_Restriction_Violation ("Generated by --",
Add_instance (agreeAndSign partOf proxy))
3.**Ground level
forall ?activity ((?activity partOf bankManager)
==TRUE) => NOT (exists ?role (?activity partOf
?role)))
```

Figure 8: Detection of consistency violation and system response.

5 CONCLUSIONS AND FUTURE WORK

Collaborative systems are dynamic (changing over time), active (carrying out processes of change) and open (changes in the business environment inducing changes in the system). A successful collaborative system at its core depends on its capacity to support both the environment and the internal changes (e.g., roles played by actors, actors' capabilities, etc.).

In this paper, we have presented a three-tier ontology for modelling and managing the intrinsic evolutionary aspects of collaborative processes. This ontology formally defines a set of essential concepts that allows for modelling functional requirements. In addition, elementary and compositionally operation has been introduced in order to control the propagation in the requirements change. These changes can also be modelled either as a snapshot at a particular instance of time or as a sequence of changes over a period of time. Also associated with the ontology are rules of changes: changes are applied in a valid collaborative system state, and after all changes are performed, result in a valid state. This ensures that every change preserves the system domain constraints. In order to show how to apply this approach, a banking system case study has been described.

In previous works (Hurtado, 2001), (Hurtado, 2002), although related to a different problem domain of that treated in this paper, have been defined mechanisms (algorithms and restrictions) on the basis of the graph theory in order to assure these properties in an automatic way. The corresponding mechanisms for the current proposal (in the collaborative system domain) are been developed by using OWL logics (Sirin, 2006), according to the multilevel scheme proposed.

Additionally, future work will be targeted to support the development of groupware applications on the basis of the proposed description and dynamic management of functional requirements. For that aim, clear connections should be established between this framework and software architectures in order to trace requirements. We are exploring ways to capture and maintain a software architecture as an instance of an architectural ontology, together with key design decisions that determine the final software architecture for each given system similarly to the approach in (Akerman, 2006).

We also plan to capitalize on our previous experiences in the development of groupware applications (Garrido, 2007), in devising an ontology-based architectural framework with evolutionary capabilities. It can provide the support for structural changes in the architecture in order to satisfy changes that can occur in the functional requirements specification.

ACKNOWLEDGEMENTS

This research is partially supported by R+D projects of the Spanish MCYT under project TIN2004-08000-C03-02.

REFERENCES

Akerman, A. Tyree, J. 2006. Using ontology to support development of software architectures. In IBM Systems Journal, Vol 45, Nº 4, pp. 813-825.

Ambler, S. W., Nalbone, J., Vizdos, M., 2003. Enterprise Unified Process: Extending the Rational Unified Process. Prentice Hall PTR.

Baader, F., Calvanese, D., McGuineness, D., Nardi, D., Patel-Schneider, P. 2003. The Description Logic Handbook. Cambridge University Press.

Chung, L. Supakkul, S., 2006. Capturing and reusing functional and non-functional requirements knowledge: A goal-object pattern approach. In Proceedings of the IEEE International Conference on Information Reuse and Integration, pp. 539-544.

Garrido, J.L., Gea, M., Rodríguez, M.L., 2005. Requirements Engineering in Cooperative Systems. Requirements Engineering for Sociotechnical Systems. Chapter XIV, IDEA GROUP, Inc.USA.

Garrido, J.L. Noguera, M. González, M. Hurtado M.V., Rodríguez, M.L., 2007. Definition and use of Computation Independent Models in an MDA-based groupware development process. In Science of Computer Programming, Vol. 66, Issue 1, pp. *25-43*, Elsevier.

Garrido, J.L. Padereswki, P. Rodríguez, M.L. Hornos, M.J. Noguera, M., 2005. A software architecture intended to design high quality groupware applications. In Proceeding of the 4th International Workshop on System/Software Architectures, IWSSA'05, Las Vegas, USA, June 2005, pp 59-65.

Gruber, T.,1995. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human and Computer Studies, Vol. 43 (5/6), pp. 907-928.

Gruninger, M., Atefi, K., Fox, M.S., 2000. Ontologies to Support Process Integration in Enterprise Engineering. In Computational and Mathematical Organization Theory, Vol. 6, Nº. 4, pp. 381-394.

Guarino, N. and Giaretta, P., 1995 Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mards (Ed.) Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. IOS Press, Amsterdam. pp.25-32.

Hurtado, M.V.,Parets, J. 2001,Evolutionary Information and Decisión Support Systems: An integration Based on Ontologies. Lecture Notes in Computer Science LNCS 2178, pp 146-159.

Hurtado, M.V.,2002 Un modelo ontológico de integración evolutivo entre sistemas de información y sistemas de ayuda a la decisión. Tesis doctoral. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Granada. Spain.

Jaekel, F.W., Perry, N., Campos, C., Mertins, K., Chalmeta, R., 2005. Interoperability Supported by Enterprise Modelling. LNCS 3762, pp. 552 – 561.

Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M., 2004. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. LNCS 3298, pp. 229 – 243.

Lu, R.Q., 2000. Ontology-based requirements analysis Journal of Software. Vol. 11(8). Pp 1009-1017.

OMG. Object Management Group: Unified Modelling Language (UML) 2.0 Superstructure Specification, August 2003. Ptc/03-08-02, pp. 455–510.

Sirin, E., Parsia, B., Cuenca-Grau, B., Kalyanpur, A., Katz, Y.. 2006. Pellet: A practical OWL-DL reasoner, Journal of Web Semantics (To Appear)

Smith, M.K. Welty, C. and McGuinness, D.L. (Eds.),2004. OWL Web Ontology Language Guide, W3C Recommendation, 10 February 2004,http://www.w3.org/TR/2004/REC-owl-guide-20040210/. Latest version available at http://www.w3.org/TR/owl-guide/.

Stojanovic, L. and Motik, B., 2002. Ontology evolution within ontology editors. In Proceedings of the OntoWeb-SIG3 Workshop at the 13th International Conference 22 on Knowledge Engineering and Knowledge Management (EKAW) , pp. 53—62.

Yuqin Lee and Wenyun Zhao, 2006. Domain Requirements Elicitation and Analysis -An Ontology-Based Approach; LNCS 3994, pp. 805 – 813. Springer-Verlag Berlin Heidelberg 2006

Zhi, Jin., 2000. Ontology-based requirements elicitation automatically. Chinese J. Computers. Vol.23, Nº.5, pp. 486-492.