A Case Study on the Construction of Application Ontologies

Luis Eduardo Santos and Rosario Girardi
Computer Science Departament
Federal University of Maranhão
São Luiz, Brazil
luis.php89@hotmail.com, rosariogirardi@gmail.com

Paulo Novais
Computer Science Departament
University of Minho
Braga, Portugal
pjon@di.uminho.pt

Abstract— **Appropriate techniques for the development of application ontologies are needed and GAODT ("Goal-Oriented Application Ontology Development Technique") technique described in this article contributes to this purpose. GAODT translates the goals and facts in natural language expressing the requirements of a knowledge-based system into rules and facts in first-order logic. Next, this knowledge base is mapped to an application ontology. GAODT was evaluated through the development of a case study on the construction of the application ontology of a knowledge-based System for the domain of Intestate Succession. A software tool to support the application of GAODT was also developed.**

Keywords-Application Ontologies; Intestate Succession; Knowledge-Based Systems; Knowledge Bases

I. INTRODUCTION

Ontologies are knowledge representation structures capable of expressing a set of entities in a given domain, their relationships and axioms, being used by modern knowledge-based systems (KBS) as knowledge bases to represent and share knowledge of a particular application domain. They allow semantic processing of information and a more precise interpretation of data, providing greater effectiveness and usability than traditional information systems [18]. The Semantic Web, a next generation Web in which the semantics of the documents, in most cases expressed only in natural language, would be expressed with ontologies is one of the largest applications of this type of knowledge representation [9][13].

An ontology is classified according to its generality, as high-level, domain, task or application ontology [14]. High-level ontologies describe generic concepts like time and space, independently of a particular domain. Domain ontologies make explicit concepts of a domain and their relationships, for example, the concepts "client", "legal-case" are the relationship "has(client, legal_case)" in the legal field. Task ontologies describe the activities of a domain, for instance, similarity analysis in the information retrieval related activities. Finally, application ontologies are specializations of domain and task ontologies, being used in a particular application, for example, the task relationship "similarity analysis" between the concepts "old legal case" and "new legal case" in a legal information retrieval system.

According to Guarino [14], this hierarchy promotes the reuse of ontologies, i.e., to build application ontologies it is necessary to extend both domain and task ontologies, and these in turn, extend high-level ontologies. However, in practice, building reusable ontologies is a costly process. Therefore, building application ontologies first and then generalizing them to domain and task ontologies is a suitable alternative [18].

Several techniques have been developed to support the process of ontology construction. However, most of them focus just on the development of domain and task ontologies. Appropriate techniques for the development of application ontologies are needed and the GAODT ("Goal-Oriented Application Ontology Development Technique") technique described in this paper contributes to this goal.

GAODT translates the goals in language natural expressing the requirement of a KBS to rules and facts in First-order logic (FOL) [19] and then extracts the elements that constitute the application ontology.

GAODT was evaluated through the development of an application ontology to be used in a KBS to support decision making in Intestate Succession domain, the branch of law that comprises the set of rules that governs the transfer of assets of someone after his death [5].

The paper is organized as follows. Section II presents the case study, emphasizing the advantages of the goal-oriented development cycle adopted by the GAODT technique and how GAODTool support its activities. Section III discusses a comparative evaluation between GAODT and some representative state of the art techniques. Section IV concludes the paper and points out some future work.

II. DEVELOPING APPLICATION ONTOLOGIES

In this section a case study that uses GAODT in the development of an application ontology in the domain of Intestate Succession is presented.

To facilitate the development of ontologies with GAODT, a semi-automated software tool (GAODTool) was developed. GAODTool has an intuitive interface, provides support to all GAODT activities and automates the creation of rules in RuleML [4] and the OWL file containing the application ontology developed.

CPS
Conference Publishing Services

A. An Overview of the GAODT Technique

Fig. 1 illustrates the GAODT technique along with its four activities: "Selection of Goals and Facts", "Representation of Predicates in FOL", "Specification of Axioms in FOL" and "Specification/Extension of the Application Ontology".

The developer of the application ontology and the domain expert participate in the execution of the activities. The developer is the knowledge engineer responsible for building the application ontology. The domain expert is someone who has expertise in an area of knowledge.

The technique takes as input a list of all the goals and facts of the system provided by the domain expert. The goals are the requirements that the KBS has to achieve, for instance, "Calculate the inheritance of a person" and the facts are general statements like "A person has descendants". In the activity "Selection of Goals and Facts", the developer, in consensus with the domain expert, selects the most representative goals and facts to be used as input of next activity. In the activity "Representation of Predicates in FOL", the developer translates the goals and facts in natural language to predicates in FOL.

The activity "Specification of Axioms in FOL" takes as input the predicates specified in the previous activity and specifies in FOL the rules needed to achieve the goals of the system. This activity is iterative, that is, a goal predicate may require the achievement of other subgoals. For example, to satisfy the goal "Determine the ascendants of a person", other subgoals should be achieved, such as "Determine the genitor of a person". All the process is iteratively executed until all the goals have been decomposed and expressed as simple facts. Finally, the activity "Specification/Extension of the Application Ontology" uses axioms generated on the previous activity and extracts from them the necessary elements to compose the application ontology. The created application ontology can be extended by performing a semantic search in a repository of application ontologies. In the next sections GAODT activities are explained in further detail.
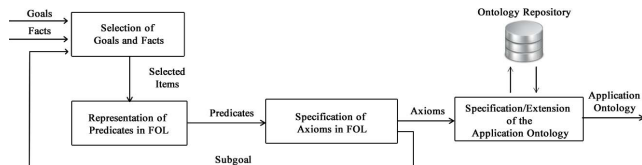

Figure 1.   An overview of the GAODT.

B. Selection of Goals and Facts

This activity takes as input a list of all the goals and facts of the system, provided by the domain expert. From this list, the developer and the specialist sets which of them will be given as input to the next activity. Fig. 2 shows a partial view of the goals and facts informed to GAODTool tool for the build of an application ontology for the domain Intestate Succession.

Initially it must be defined the general goal and the main specific goals of the system. For instance, for the general goal 1: "Calculate the inheritance of a person", the main specific goals are 2: "Identify the heirs of a person" and 3: "Determine

the inheritance of the heirs". To satisfy these subgoals, other goals could be defined in subsequent iterations, in a process performed recursively for all the goals in the list.

Figure 2.   List of goals and facts of the ontology

C. Representation of Predicates in FOL

This activity consists in translating the items selected in the previous activity, expressed in natural language to predicates in FOL. It consists of seven sub-activities: "Identification of entities", "Redefinition of entities", "Identification of relationships", "Redefinition of relationships", "Definition of the arity", "Definition of predicates" and "Redefinition of the entities of the predicates".

In the sub-activity "Identification of entities", all explicit or implicit subjects and objects in a sentence are identified from the items selected in the previous activity and exemplified in Fig. 2. The result of this sub-activity is shown in Fig. 3.

Figure 3.   Inputs and outputs of the sub activity "Identification of entities''.

The sub-activity "Redefinition of entities" takes into account the entities identified in Fig. 3, and for each one verifies if that is an entity or a relationship. The entity "Heirs" is actually a relationship between two "People", i.e., "A person is heir of another person". So, "Heirs" is redefined as a "Person", considering the entities that integrate the relationship. However, the word "Heirs" is not discarded, it will be useful in the sub-activity "Redefinition of relationships". Fig. 4 shows the result of this sub-activity.

Figure 4.   Inputs and outputs of the sub-activity "Redefinition of entities".

620

The sub-activity "Identification of relationships" uses the items selected in the activity "Selection of Goals and Facts" to identify verbs in the phrases, which represent the relationships to be extracted. For instance, in the selected item "Calculate the inheritance of a person", the verb "Calculate" is identified as a relationship. Fig. 5 shows the relationships identified.

Identification of relationships	
Selected items	Relationships
Calculate the inheritance of a person	calculate
Identify the heirs of a person	identify
Determine the inheritance of the heirs	determine

Figure 5.   Inputs and outputs of the sub-activity "Identification of relationships"

The sub-activity "Redefinition of relationships" takes into account the relationships identified in the previous sub-activity (exemplified in Fig. 5) and verifies if these relationships are transitive verbs, as they need a supplement to make sense. For example, the relationship "identify" needs a supplement to give it sense, using their respective entities identified in Fig. 4 or the words that were considered entities in the first sub-activity, for example, the word "Heirs". Fig. 6 shows the result of this sub-activity applied to the examples in Fig. 4 and Fig 5.

Redefinition of relationships		
Relationships	Entities/Words redefined	Redefinition of relationships
calculate	Inheritance,Person	calculateInheritance
identify	Heirs,Person	identifyHeirs
determine	Heirs,Inheritance,Person	determineInheritance

Figure 6.   Inputs and outputs of the sub-activity "Redefinition of relationships"

The sub-activity "Definition of the arity" defines the number of entities involved in the relationships previously identified. This quantity is determined according to by the number of entities identified on each selected item. Fig. 7 shows the arity identified for the entities in Fig. 4

Definition of the arity		
Relationships	Entities	Arity
calculateInheritance	Inheritance, Person	2
identifyHeirs	Person, Person	2
determineInheritance	Person, Inheritance	2

Figure 7.   Inputs and outputs of the sub-activity "Definition of the arity".

In the sub-activity "Definition of predicates" the entities and relationships identified and illustrated in Fig. 7 and Fig. 6 are represented in FOL. Fig. 8 presents the predicates resulting from the realization of this sub-activity.

Definition of predicates		
Relationships	Entities	Predicates
calculateInheritance	Inheritance, Person	calculateInheritance(Inheritance, Person)
identifyHeirs	Person, Person	identifyHeirs(Person, Person)
determineInheritance	Person, Inheritance	determineInheritance(Person, Inheritance)

Figure 8.   Translation of the selected items into predicates in FOL.

The sub-activity "Redefinition of the entities of the predicate" aims at renaming the arguments of the predicates, defined in the sub-activity "Definition of predicates", when the arguments have the same name. For example, for the predicate "identifyHeirs(Person, Person)", the entities are considered variables since they represent distinct persons. So it is redefined to "identifyHeirs(PersonX, PersonY)" and this change is also propagated to all other predicates in Fig. 8. Fig. 9 presents the result of this sub-activity and the final product of this activity.

Redefinition of the entities of the predicate	
Predicates	Predicates redefined
calculateInheritance(Inheritance, Person)	calculateInheritance(Inheritance, PersonX)
identifyHeirs(Person, Person)	identifyHeirs(PersonX, PersonY)
determineInheritance(Person, Inheritance)	determineInheritance(PersonY, Inheritance)

Figure 9.   Inputs and outputs of the sub-activity "Redefinition of the entities of the predicate"

D.   Specification of Axioms in FOL

The purpose of this activity is to specify the rules that lead to the achievement of the goals of the system which are represented as predicates in FOL. The process is iterative, because there is an iteration with the activity "Selection of Goals and Facts". For each goal contained in a rule a search is performed in the list of goals and facts to retrieve the subgoals that satisfy it.

This activity consists of four sub-activities: "Definition of the condition and conclusion", "Definition of boolean operators", "Definition of quantifiers" and "Definition of implication or equivalence".

The sub-activity "Definition of the condition and conclusion" determines the condition and the conclusion of each rule. The conclusion is the main goal that has to be achieved and the condition can be considered as a set of assumptions or subgoals that lead to the achievement of the main goal. This sub-activity receives as input the predicates identified in Fig. 9. Fig. 10 shows the result of this sub-activity.

Definition of the condition and conclusion		
		Predicates
Conclusion	▾	calculateInheritance(Inheritance, PersonX)
Condition	▾	identifyHeirs(PersonX, PersonY)
Condition	▾	determineInheritance(PersonY, Inheritance)

Figure 10. Output of the sub-activity "Definition of the condition and conclusion of the axiom"

The sub-activity "Definition of boolean operators" specifies the boolean operators which integrate the predicates of the axiom condition. The operators used are the conjunction represented by the symbol (^) and the disjunction represented by the symbol (∨).

Predicates in the condition are joined by an "and" operator when all of them are needed to achieve the conclusion; by an "or" operator when they are alternative predicates to achieve the conclusion. For example, to achieve the goal "Calculate the inheritance of a person" (calculateInheritance(PersonX, Inheritance)), it is necessary to satisfy all the goals "Identify the heirs of a person" (identifyHeirs(PersonX, PersonY)) and "Determine the inheritance of each heir" (determineInheritance(PersonY,Inheritance)). Fig. 11 shows the boolean operators used to join these two predicates.
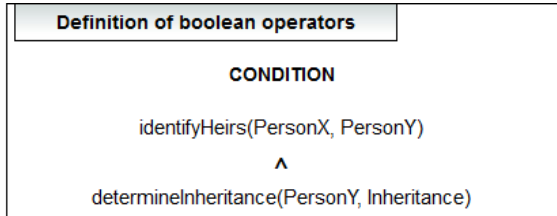

Definition of boolean operators

CONDITION

identifyHeirs(PersonX, PersonY)

∧

determineInheritance(PersonY, Inheritance)

Figure 11. The union of predicates with boolean operators

The sub-activity "Definition of quantifiers" defines the appropriate quantifiers associated to entities present in the axiom. Quantifiers can be universal ($\forall$) or existential ($\exists$). The first one is used to indicate that a predicate is true for all the elements of a given set while the last one is used to indicate that a predicate is true for at least one element in a given set. For instance, the variable "PersonX" refers to "at least one person who died" so the existential quantifier is associated to this entity. The variables "PersonY" and "Inheritance" follow the same principle, being set to the existential quantifier (Fig. 12).

Definition of quantifiers

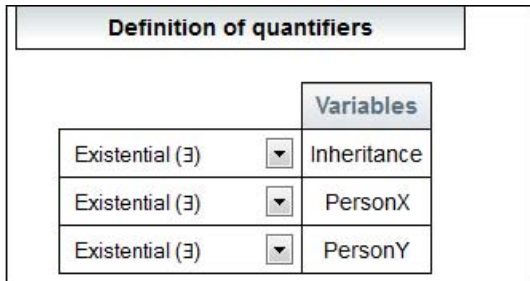|		Variables
Existential ($\exists$)	▼	Inheritance
Existential ($\exists$)	▼	PersonX
Existential ($\exists$)	▼	PersonY

Figure 12. Definition of the quantifiers for the variables of the predicates.

The sub-activity "Definition of implication or equivalence" takes as input a set of predicates like those in the example of Fig. 9 and determines whether the axiom to be created is an implication or an equivalence. The implication is used when the satisfaction of the condition leads to the conclusion. The equivalence occurs when there is a symmetry between condition and conclusion. Fig. 13 illustrates an implication used to form an axiom.

Definition of implication or equivalence

$\exists$ Inheritance, $\exists$ PersonX, $\exists$ PersonY
identifyHeirs(PersonX, PersonY) ∧ determineInheritance(PersonY, Inheritance)
Implication ($\Rightarrow$) ▼
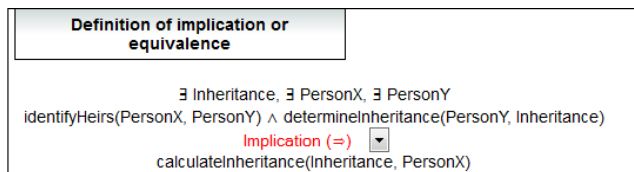calculateInheritance(Inheritance, PersonX)

Figure 13. Definition of the implication of the axiom.

After the execution of the sub-activities "Definition of the condition and conclusion", "Definition of boolean operators", "Definition of quantifiers" and "Definition of implication or equivalence" the axiom "$\exists$ PersonX, PersonY, Inheritance | identifyHeirs(PersonX, PersonY) ∧ determineInheritance (PersonY, Inheritance) $\Rightarrow$ calculateInheritance(PersonX, Inheritance)" is created and stored in the compartment "Axioms Developed", as shown in Fig. 14.

Axioms Developed

✖ ✚ $\exists$ Inheritance, $\exists$ PersonX, $\exists$ PersonY | identifyHeirs(PersonX, PersonY) ∧ det
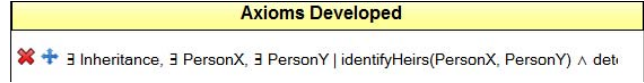
Figure 14. Example of an axiom developed

As illustrated in Fig. 1, the GAODT activities are executed iteratively. Therefore, in order to construct new axioms, each one of the predicates in the condition of the current axiom is submitted to the "Selection of Goals and Facts" activity where items in the list of the goals and facts (Fig. 2) that satisfy this condition will be selected.

For instance, the predicate "identifyHeirs(PersonX, PersonY)" which is part of the condition of the axiom in Fig. 14 is submitted to the "Selection of Goals and Facts" and the domain specialist informs that the four goals "Identify the descendants of a person", "Identify the spouse of a person", "Identify the ascendants of a person" and "Identify the collaterals of a person" (Fig. 2) satisfy it. These four goals in natural language are then given as input to the activity "Representation of Predicates in FOL" to be represented in FOL. Finally, the goals are given as input to the activity "Specification of Axioms in FOL" which generate the new axiom: identifyDescendants(PersonX, PersonY) ∨ identify Ascendants(PersonX, PersonY) ∨ identifySpouse(PersonX, PersonY) ∨ identifyCollaterals $\Rightarrow$ identifyHeirs(PersonX, PersonY).

The process is then recursively executed for each one of the subgoals, until all the goals given as input to the technique (as the ones illustrated in Fig. 2) have been satisfied. The product of this activity is a set of axioms specified in predicates in FOL. A sub-set of the axioms generated from the activity "Specification of Axioms in FOL" is shown in Fig. 15.

Axioms Developed

✖ ✚ $\exists$ PersonX, $\exists$ PersonY | hasAscendants(PersonX,PersonY) $\Rightarrow$ verifyExistenc
✖ ✚ $\exists$ PersonX, $\exists$ PersonY | hasDescendants(PersonX,PersonY) $\Rightarrow$ verifyExisten
✖ ✚ $\exists$ PersonX, $\exists$ PersonY | hasSpouse(PersonX,PersonY) $\Rightarrow$ verifyExistenceSp
✖ ✚ $\exists$ PersonX, $\exists$ PersonY | hasCollaterals(PersonX,PersonY) $\Rightarrow$ verifyExistence
✖ ✚ $\exists$ Properties, $\exists$ PersonX, $\exists$ PersonY | sumProperties(Properties,PersonX) $\Rightarrow$
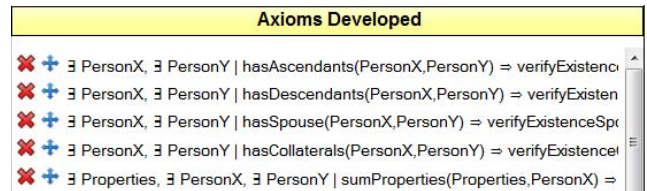
Figure 15. A sub-set of axioms generated from the activity "Specification of Axioms in FOL"

E. Specification/Extension of the Application Ontology

The constituent elements of the axioms specified in the previous activity are extracted for the construction of the application ontology. This activity consists of six sub-activities: "Translation of axioms", "Definition of classes", "Definition of non-taxonomic relationships", "Definition of taxonomic relationships", "Definition of properties" and "Retrieval of application ontologies".

The sub-activity "Translation of axioms" converts the axioms defined in the previous activity expressed in FOL into rules expressed in an ontology rule based language, like RuleML [4]. The experiences conducted to evaluate GAODT use RuleML because of its expressiveness.

To perform this translation, the following heuristics are applied. First, regular expressions [7] are used to extract the premises and the conclusions of the axioms. For example, for the rule "∃ PersonX, PersonY, Inheritance | identifyHeirs(PersonX, PersonY) ^ determineInheritance (PersonY, Inheritance) ⇒ calculateInheritance(PersonX, Inheritance)", the following regular expression was used "^(\w+\(.*\)) ^ (\w+\(.*\)) ⇒ (\w+\(.*\))". Then, the premises and conclusion are specified in POSL [8] and finally automatically translated to RuleML axioms (Fig. 16).

```xml
<Implies>
    <And>
        <Atom>
            <Rel>identifyHeirs</Rel>
            <Var>PersonX</Var>
            <Var>PersonY</Var>
        </Atom>
        <Atom>
            <Rel>determineInheritance</Rel>
            <Var>PersonY</Var>
            <Var>Inheritance</Var>
        </Atom>
    </And>
    <Atom>
        <Rel>calculateInheritance</Rel>
        <Var>PersonX</Var>
        <Var>Inheritance</Var>
    </Atom>
</Implies>
```

Figure 16. Example of an axiom represented in RuleML.

The sub-activity "Definition of classes" extracts the variables of the axioms of the "Specification of Axioms in FOL" activity illustrated in Fig. 15. For example, the predicate "identifyHeirs(PersonX, PersonY)" has the variables "PersonX" and "PersonY" both referring to the class "Person". Fig. 17 shows the extracted classes.

Definition of classes				
Classes				
Person	Properties	Inheritance	MaritalProperties	Pre-MaritalProperties

Figure 17. Classes of the application ontology

The sub-activity "Definition of non-taxonomic relationships" extracts non-taxonomic relationships of the ontology from the predicates in the list of axioms outputted from the activity "Specification of Axioms in FOL" (Fig. 15). For example, in relation to the predicate "calculateInheritance(PersonX, Inheritance)" the non-taxonomic relationship identified is the predicate "calculateInheritance" which defines the relationship between the classes "Person" and "Inheritance". Fig. 18 shows a partial view of the non-taxonomic relationships identified.

	Domain	Relation	Range
☑	Person ▾	hasAscendants	Person ▾
☑	Person ▾	hasDescendants	Person ▾
☑	Person ▾	hasPre-MaritalProperties	Pre-MaritalProperties ▾
☑	Person ▾	calculateInheritance	Inheritance ▾

Figure 18. Partial view of the non-taxonomic relationships identified

The sub-activity "Definition of taxonomic relationships" extracts a set of taxonomic relationships based on the hierarqical relation between the classes outputted from the sub-activity "Definition of classes" (Fig. 17). For example, there is a hierarchy between the classes, "Properties", "MaritalProperties" and "Pre-MaritalProperties". Fig. 19 shows the taxonomic relationships identified.

Definition of taxonomic relationships			
	Classe		Classe
❌	MaritalProperties	Subclasse of	Properties
❌	Pre-MaritalProperties	Subclasse of	Properties

Figure 19. Taxonomic relationships of the ontology

The sub-activity "Definition of properties" extracts from the axioms the predicates describing attributes of the classes. For example, "hasValue(MaritalProperties, Value)" and "hasValue(Pre-MaritalProperties, Value)" describe that the classes "MaritalProperties" and "Pre-MaritalProperties" has the property "hasValue". Fig. 20 shows a partial view of the properties identified.

	Domain	Property	Range
☑	Person ▾	sumProperties	float ▾
☑	Person ▾	sumMaritalProperties	float ▾
☑	Pessoa ▾	determineQuantityDescendants	int ▾
☑	Person ▾	verifyExistenceRegime	boolean ▾
☑	MaritalProperties ▾	hasValue	float ▾
☑	Pre-MaritalProperties ▾	hasValue	float ▾

Figure 20. Properties identified

If there is a need to extend the application ontology developed, the sub-activity "Retrieval of application ontologies" performs a semantic search for reusable ontologies in a repository. Several similarity measures [20] [1] can be used to rank the ontologies retrieved.

Finally, a file containing the developed OWL application ontology is generated. Fig. 21 presents the ontology taxonomy and a partial view of the non-taxonomic relations and properties of the ontology in the graphic environment of Protègè.
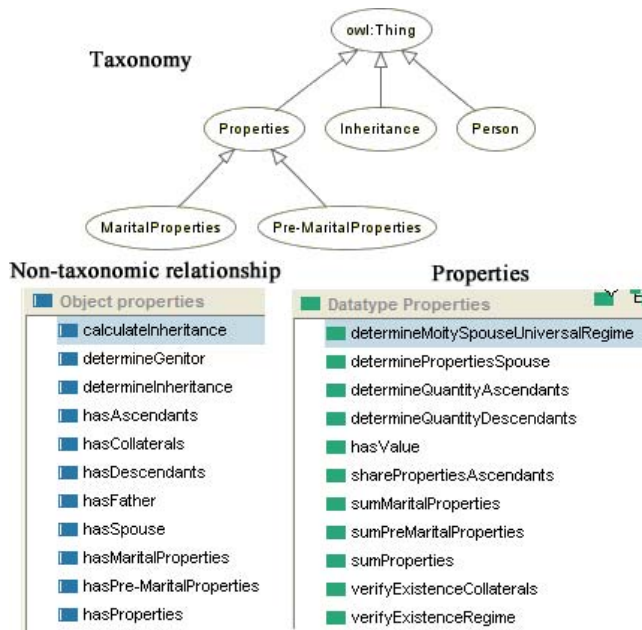
Figure 21. Application ontology developed in the Intestate Succession domain.

III. CONCLUSION AND FURTHER WORK

This article described GAODT, a technique for building application ontologies through a goal-oriented development cycle. The technique also provides the developer, a well-defined way to translate the knowledge expressed in natural language to a computational representation. This feature is not covered by any other technique of the state of the art presented in this paper.

To evaluate GAODT a case study was developed, which consisted in the construction of an application ontology to be used in a KBS to support decision making in the Intestate Succession domain. GAODTool, a software tool which provides support to all GAODT activities and automates the creation of rules in RuleML and the OWL file containing the application ontology was also presented.

Building reusable ontologies is a costly process. Among the four types of ontologies defined by Guarino [14], application ontologies are the less reusable once they are developed for specific software applications. However, they are generally easier and faster to develop. Building application ontologies and then generalizing its elements to domain and task ontologies is a good alternative approach for developing these reusable artifacts. In this context, GAODT consists of a first step in this direction by defining a systematized way for building application ontologies.

Further improvements of GAODT include a technique to perform the semantic search for ontologies to be reused. GAODT will also be integrated into a knowledge based process for the development of multi-agent systems [2] [17]. The main objective of GAODT in this context is to construct the knowledge bases of deliberative agents of KBS developed with this process.

REFERENCES

[1] A. Claudia, N. Fanizzi and F. Esposito. A semantic similarity measure for expressive description logics. In Proceedings of Italian Conference on Computational Logic, Roma, 2005.

[2] A. Costa. MADAE-Pro A knowledge-based process for Domain and Application Engineering. Master thesis - Federal University of Maranhão, 2009. (In Portuguese).

[3] A. Pérez, M. F. Lopez and O. Corcho. Ontological Engineering: with examples from the areas of knowledge management, e-commerce and the semantic web. London. Springer-Verlag, 2004.

[4] B. Harold. The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformations. In Proc. 14th International Conference on Applications of Prolog, 2001.

[5] C. Gonçalves. Brazilian Civil Law: Inheritance law. São Paulo, Saraiva, 2009. (In Portuguese).

[6] F. Caliari. DERONTO: Method for Building Ontologies from Entity-Relationship Diagrams. Master thesis - Federal Technological University of Paraná, 2007. (In Portuguese)

[7] F. Jeffrey. Mastering Regular Expressions. O'Reilly Media, 3rd Edition, 2006.

[8] H. Boley. POSL: An Integrated Positional-Slotted Language for Semantic Web Knowledge. W3C, 2004.

[9] K. Bontcheva and H. Cunningham. The Semantic Web: A New Opportunity and Challenge for Human Language Technology, In Proceedings of Workshop on Human Language Technology for the Semantic Web and Web Services, 2nd International Semantic Web Conference, Sanibel Island, 2003.

[10] M. Fernández, A. Pérez and N. Juristo. Methontology: From Ontological Art Towards Ontological Engineering. Spring Symposium Series. Stanford, 1997.

[11] M. Gruninger and M. Fox. Methodology for the design and evaluation of ontologies. In: IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.

[12] M. Uschold and M. King. Towards a Methodology for Building Ontologies. In IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, 1995.

[13] N. Guarino, C. Masolo and C. Vetere. Ontoseek: Content-based Access to the web. IEEE Intelligent Systems, vol. 14 (3), 1999, pp. 70-80.

[14] N. Guarino. Formal Ontology in Information Systems. Proceedings of the 1st International Conference, Trento, Italy, IOS Press, 1998.

[15] N. Noy and D. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology, 2001.

[16] R. Dale, H. Moisl and H. Somers. Handbook of natural language processing, CRC, 2000.

[17] R. Girardi and A. Leite. Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications, Hershey: IGI Global, 2011 (in press).

[18] R. Girardi. Guiding Ontology Learning and Population by Knowledge System Goals. In: Proceedings of the International Conference on Knowledge Engineering and Ontology Development, Ed. INSTIIC, Valence, 2010, pp. 480 – 484.

[19] S. Russel and P. Norvig. Artificial Intelligence. Rio de Janeiro: Ed. Campus, 2004. (In Portuguese).

[20] W. Lee, N. Shah, K. Sundlass and M. Musen. Comparison of ontology-based semantic-similarity measures. AMIA Symposium, 2008, pp. 384–388.