Pre-conceptual Schema: a Conceptual-Graph-like Knowledge Representation for Requirements Elicitation*

Carlos Mario Zapata Jaramillo,[1] Alexander Gelbukh,[2] and Fernando Arango Isaza[1]

[1] Universidad Nacional de Colombia, Facultad de Minas, Escuela de Sistemas
Carrera 80 No. 65-223 Of. M8-113, Medellín, Colombia
cmzapata@unal.edu.co, farango@unal.edu.co

[2] Computing Research Center (CIC), National Polytechnic Institute,
Col. Zacatenco, 07738, DF, Mexico
www.Gelbukh.com

Abstract. A simple representation framework for ontological knowledge with dynamic and deontic characteristics is presented. It represents structural relationships (*is-a*, *part/whole*), dynamic relationships (actions such as *register*, *pay*, etc.), and conditional relationships (*if-then-else*). As a case study, we apply our representation language to the task of requirements elicitation in software engineering. We show how our pre-conceptual schemas can be obtained from controlled natural language discourse and how these diagrams can be then converted into standard UML diagrams. Thus our representation framework is shown to be a useful intermediate step for obtaining UML diagrams from natural language discourse.

1 Introduction

Knowledge Representation (KR) has been applied in software development, in tasks such as requirements elicitation, formal specification, etc. [1]. In requirements elicitation, the Stakeholder's discourse is transformed in software specifications by means of a process that involves intervention of the Analyst. Some works in KR have been made in representation of requirements, but there are still problems in Stakeholder validation and dynamic features of the paradigms used for this goal.

Several paradigms have been used for KR, such as semantic networks, frames, production rules, and predicate logic [1, 2]. In particular, Conceptual Graphs (CG) [3] have been used for KR because of its logic formalism.

In this paper we present Pre-conceptual Schemas, a simple CG-like KR framework motivated by the Requirements Elicitation task. On the one hand, these schemas can be obtained from a controlled natural language discourse. On the other hand, we show how to transform them to UML diagrams. Thus these schemas can be used for automatic conversion of natural language discourse into UML diagrams.

* Work done under partial support of Mexican Government (SIP-IPN 20061299 and CONACyT R420219-A, 50206) for the second author.

The paper is organized as follows. Section 2 presents an overview of KR. Section 3 discusses previous KR applications to Requirements Elicitation. Section 4 introduces the Pre-conceptual Schemas as a KR framework. Section 5 presents a case study based on Pre-conceptual Schemas in order to automatically acquire UML diagrams and compares Pre-conceptual Schemas with CGs. Section 6 concludes the paper.

2 Overview of Knowledge Representation

Sowa [1] defined KR as "the application of logic and ontology to the task of constructing computable models for some domain". In KR, a major concern is computational tractability of knowledge to reach automation and inference. The field of KR is usually called "Knowledge Representation and Reasoning", because KR formalisms are useless without the ability to reason on them [1].

A comprehensive description of KR can be found in [1]; a discussion of relationships between KR and Ontologies, in [2]. The major paradigms in KR are as follows.

- *Semantic networks* are used as a graphical paradigm, equivalent to some logical paradigms. They are useful for hierarchical representation. Nowadays, a number of graphs formalisms are based on the syntax of semantic networks, for example Conceptual Graphs [3]. Semantic networks are unstructured.
- *Frames* are templates or structured arrays to be filled with information. Frames can be considered as "structured" semantic networks, because they use data structures to store all structural knowledge about a specific object in one place. Object-oriented descriptions and class-subclass taxonomies are examples of frames.
- *Production rules* are hybrid procedural-declarative representations used in expert systems; many declarative languages are based on this paradigm. The reasoning process can be automatically traced in a controlled natural language [19].
- *Predicate logic* is based on mathematics and can be used as a reasoning mechanism for checking the correctness of a group of expressions. Programming languages such as PROLOG are logic-based.

Sowa [1] discusses major KR formalisms such as rules, frames, semantic networks, object-oriented languages (for example, Java), Prolog, SQL, Petri networks, and the Knowledge Interchange Format (KIF). All these representations are based on one or several of the mentioned paradigms. In particular, KIF has emerged as a standard model for sharing information among knowledge-based applications. KIF is a language designed to be used in knowledge exchange among disparate computer systems (created by different programmers, at different times, in different languages, etc.) [4].

3 State-of-the-Art in KR-based Requirements Elicitation

According to Leite [5], "*Requirements analysis is a process in which 'what is to be done' is elicited and modelled. This process has to deal with different viewpoints, and it uses a combination of methods, tools, and actors. The product of this process is a* *model, from which a document, called requirements, is produced.*" Requirements Elicitation (RE) is a difficult step in the software development process. Viewpoints reported by Leite are associated with several Stakeholders—people with some concern in software development—and are difficult to collect for Analysts: Stakeholders are committed with domain discourse, while Analysts are concerned with modelling languages and technical knowledge. This is a cause for many miscommunication problems.

Some RE projects have used KR for solving such miscommunication problems:

- *Frames* were employed by Cook *et al.* [6] for gathering information about RE problems. The frames were used for communication purposes and Stakeholder validation, but did not contribute to further automation in software development.
- *Logical languages* were used in ERAE [7, 8], RML [9, 10], Telos [11, 12], FRORL [13], and PML [14] projects. These languages require technical training for their use and elaboration, which Stakeholders do not have. Furthermore, KR languages are used only for representation and inference. They are not used for conversion to other standard specification formalisms, such as UML diagrams.
- *Controlled English* was used in CPE [15], ACE [16], and CLIE [17] projects. Again, it is not converted to other standard specification formalisms.
- *Conceptual Graphs* were used by Delugach and Lampkin [18] to describe requirement specifications. However, they use technical terminology (like "object" or "constraint") that the Stakeholder usually misunderstands.

As far as CGs are concerned as KR language, there are other problems:

- *They represent the specifications phrase-by-phrase.* This can lead to the repetition of a concept many times. To solve this problem, CG standard has proposed co-reference lines, but complex concepts can be spread across many CGs, and their behaviour can be difficult to validate.
- *Their syntax can be ambiguous.* Concepts can be either nouns or verbs or even entire graphs. Relationships can be either thematic roles or comparison operators. This can lead to multiple representations of the same phrase.
- *They represent mainly structural information.* For better expressiveness, we need a schema capable of representing both structural and dynamic properties.

4 Pre-conceptual Schemas: CG-like Framework for Knowledge Representation

Design Goals In order to solve the problems related to CGs mentioned in Section 3, a KR approach to obtaining UML diagrams should meet the following conditions:

- *Unambiguous rules* must be provided. Precise rules may map words to only one element of each resulting diagram.
- *Automated translation* from controlled language into a KR language and into UML Diagrams is to be possible.
- *Applicability to any domain* is expected, no matter how specific the domain is.

– *No pre-classification ontologies* are to be used.
– *Several UML diagrams* should be obtainable from the same source.
– *Use of a common KR formalism*, no matter what the target diagram is.
– *The KR formalism must be an integration* of all the target diagrams.

Pre-conceptual Schemas are proposed as a KR formalism for automatically obtaining of UML Diagrams that is aimed at satisfying these requirements.

The Term *Pre-conceptual* This term was coined by Heidegger [20], referring to a previous knowledge about a concept. Piaget [21], in his Stage Theory, distinguishes a pre-conceptual stage, at which children have a certain understanding of class membership and can divide their internal representations into classes.

In software development, Analyst builds Conceptual Schemas based on the Stakeholder discourse. Analyst performs an *analysis* to find the ideas behind the discourse and internally in his or her mind depicts something like a *pre-concept* of the Conceptual Schema. Following this idea, the proposed framework may build a KR description of the Stakeholder discourse. Thus the term *Pre-conceptual Schema*.

Syntax and Semantics Pre-conceptual Schemas (PS) use a notation reminiscent of that of Conceptual Graphs (CG), with certain additional symbols representing dynamic properties. A Pre-conceptual Schema is a (not necessarily connected) labelled digraph without loops and multiple arcs, composed of the nodes of four types connected by the arcs of two types shown in Figure 1, with the following restrictions:

Topology

– A *connection* arc connects a concept to a relationship or vice versa.
– An *implication* arc connects a dynamic relationship or conditional to a dynamic relationship.
– Every *concept* has an incident arc (which is of connection type, going to or from a relationship).
– A *dynamic relationship* has exactly one incoming and one outgoing connection arcs (incident to concepts; it can have any number of incident implication arcs).
– A *structural relationship* has exactly one incoming and one or more outgoing arcs (of connection type, incident to concepts).
– A *conditional* has no incoming arcs and one or more outgoing arcs (of implication type, going into dynamic relationships).

Labels

– A *connection* arc has no labels.
– An *implication* arc has a label *yes* or *no* (if omitted, *yes* is assumed).
– A *concept* is labelled with a noun representing an entity of the modelled world. Different concepts nodes have different labels.

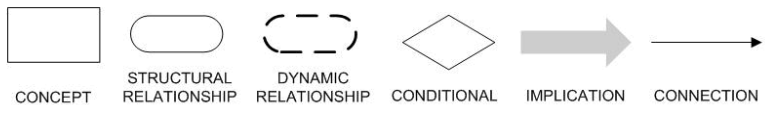

Fig. 1. Syntactic elements of Pre-conceptual Schemas

- A *dynamic relationship* is labelled with an action verb, e.g., *pay*, *register*. Different dynamic relationship nodes can have the same label.
- A *structural relationship* is labelled with a verb *is* or *has*.
- A *conditional* is labelled with a logical condition on values of certain concepts, e.g., *score* > *3*. A description of the formal language for expressing such conditions is beyond the scope of this paper.

Semantics

- *Connections* express argument structure of relationships: roughly speaking, the subject and the object of the corresponding verb. A concept can participate in various relationships (*secretary* → *prints* → *report*, *secretary* → *calls* → *client*, *director* → *employs* → *secretary*; here *secretary* is the same node).
- *Concepts* represent people (*employee*, *secretary*), things (*document*, *bill*), and properties (*address*, *phone*). In requirement elicitation, one can very roughly imagine them as what later might become dialog boxes shown to the user of the given category or representing the given thing, or as fields in such boxes.
- *Structural relationships* express class hierarchy (*secretary* is an *employee*), properties (*employee* has *phone*), part-whole relationships (*car* has *motor*), etc. One relationship node can only have one subject and one object (*secretary* → *prints* → *report*, *accountant* → *prints* → *bill*; these are two different *print* nodes). In requirement elicitation, one can roughly imagine the properties as text field or links on the dialog boxes corresponding to their owners.
- *Dynamic relationships* express actions that people can perform (*secretary* can *register* the *bill*). In requirement elicitation, one can roughly imagine them as buttons that the users of the software can press to perform the corresponding actions.
- *Conditionals* represent prerequisites to perform an action. In requirements elicitation, one can roughly imagine them as enabling or disabling the corresponding buttons, depending on whether a condition is true (accountant can pay a bill after the bill has been registered) or some another action has been performed (*accountant* can *pay* a *bill* only if secretary has *registered* the *bill*).
- *Implications* arc has a label *yes* or *no* (if omitted, *yes* is assumed). It represents logical implication between events.

Comparison of Pre-conceptual Schemas and Conceptual Graphs While the syntax and semantics of Pre-conceptual schemes strongly resemble those of Conceptual Graphs, there are some important differences.

- PS concepts differ from CG concepts in that CG concepts can be nouns, verbs or graphs. PS concepts are restricted to nouns from the Stakeholder's discourse.
- PS relationships differ from CG relationships in that the latter can be nouns (for example, thematic roles), attributes, and operators. PS relationships are restricted to verbs from the Stakeholder's discourse. There are two kinds of PS relationships:
  - Structural relationships (denoted by a solid line) correspond to structural verbs or permanent relationships between concepts, such as *to be* and *to have*.
  - Dynamic relationships (denoted by a dotted line) correspond to dynamic verbs or temporal relationships between concepts, such as *to register*, *to pay*, etc.
- PS implications are cause-and-effect relationships between dynamic relationships.

– PS conditionals are preconditions—expressed in terms of concepts—that trigger some dynamic relationship.
– PS connections are used in a similar way to CG connections: they can connect a concept with a relationship and vice versa. Furthermore, PS connections can connect a conditional with a dynamic relationship.

Some differences between PS and CG can be noted from Figures 3 and 4 below:

– The Conceptual Graph in Figure 4 is one of the possible CGs that can be obtained. The syntax of CG can derive more than one representation. In contrast, PS in Figure 3 is the only possible representation that can be obtained from the given UN-Lencep specification.
– Concepts are repeated in CG because every CG tries to represent a sentence. In PS, a concept is unique and it is possible to find all the relationships it participates in.
– In CG, there is no difference between the concepts like *assess* and *grade_mark*, because representation is the same in both cases. In PS, *assess* is a dynamic relationship, while *grade_mark* is a concept.
– In CG, verbs such as *have* and *assess* have the same representation (an agent and a theme). In PS these verbs have different representations: *have* is a structural relationship and *assess* is a dynamic relationship.
– Stakeholder validation of the obtained PS is easier than CG validation, because relationships like *agent* and *theme* are not present in UN-Lencep specification.
– If we use CG for representing Stakeholder's discourse as in [18], we need words like *attribute* and *constraint*, which belong to software discourse. In PS, we only need words from the UN-Lencep specification.

UN-Lencep Language A subset of natural language, the Controlled Language called UN-Lencep (acronym of a Spanish phrase for *National University of Colombia—Controlled Language for Pre-conceptual Schema Specification*) is defined in such a way that simplifies automatic obtaining of Pre-conceptual Schemas from a discourse. Unrestricted natural language is very complex and has many linguistic irregularities and phenomena difficult to tackle computationally—such as anaphora, syntactic ambiguities, etc.—that make it difficult to obtain PS elements from a text. However, if the Stakeholder is capable to express his or her ideas in a simpler subset of natural language, PS can be directly obtained from such a discourse.

Figure 2 shows the basic syntax of UN-Lencep, and Table 1 shows equivalences for the basic specification of UN-Lencep. In the table, the left-hand side column shows the formal elements expressed by the controlled natural language expressions shown in the right-hand side.

Rules for Obtaining UML Diagrams PS can be mapped in three UML diagrams: Class, Communication, and State Machine diagrams. To achieve this goal, we define 14 rules based on PS elements. Space limitations do not allow us to discuss or even list here all those rules, but following are some examples of such rules:

– A source concept from a HAS/HAVE relationship is a candidate class.
– The source set of concepts and relationships from an implication connection is a candidate guard condition.

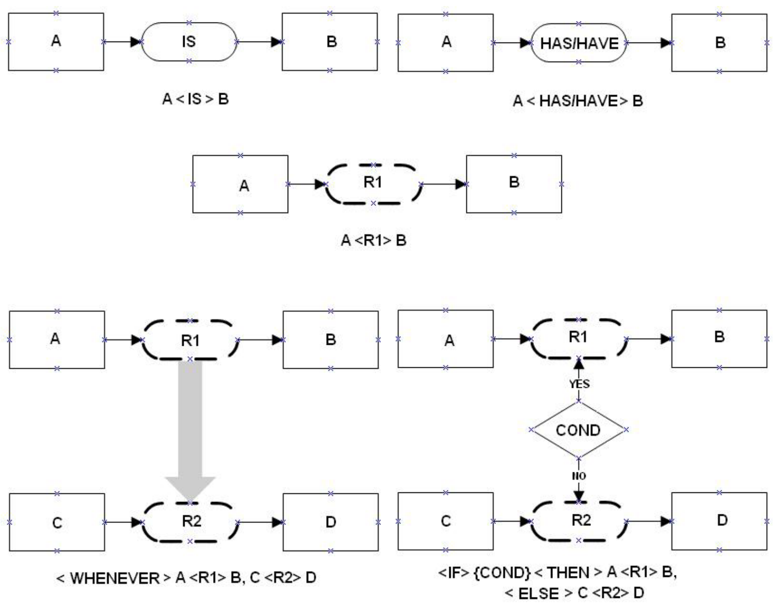

Fig. 2. Basic Syntax of UN–Lencep

Table 1. Equivalences for basic specification of UN–Lencep

Formal construction	*Controlled natural language expression*	
A <IS> B	A *is kind of* B	A *is a sort of* B
	A *is a type of* B	B *is divided into* A
A <HAS/HAVE> B	A *includes* B	B *is part of* A
	A *contains* B	B *is included in* A
	A *possesses* B	B *is contained in* A
	A *is composed by* B	B *is an element of* A
	A *is formed by* B	B *is a subset of* A
	B *belongs to* A	
<WHENEVER> A <R1> B, C <R2> D	*if* A <R1> B *then* C <R2> D	
	since A <R1> B, C <R2> D	
	after A <R1> B, C <R2> D	

– Messages identified in communication diagrams—expressed in past participle—are candidate states for target object class.

5 Automatically Obtaining UML Diagrams from UN-Lencep Specifications using Pre-conceptual Schemas

In the following example, we define a UN-Lencep specification and construct the Pre-conceptual Schema (Figure 3) and the Conceptual Graph representing the same dis-

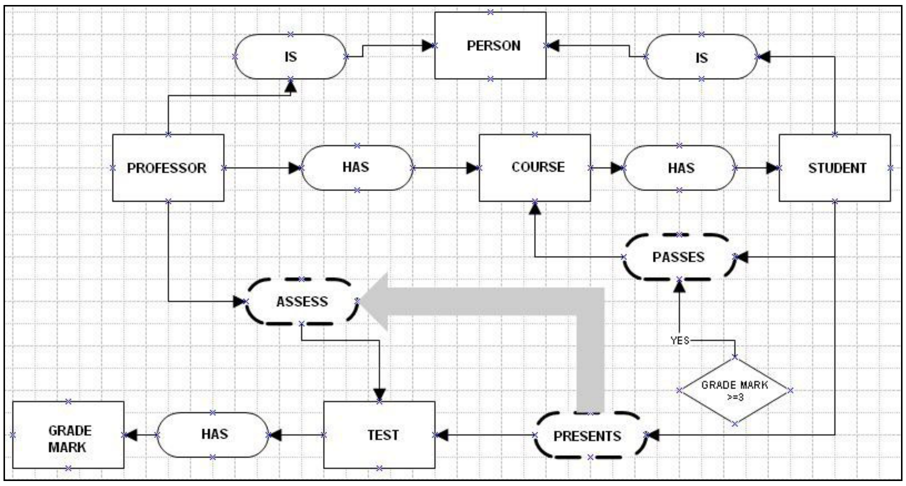

Fig. 3. PS of the example discourse.

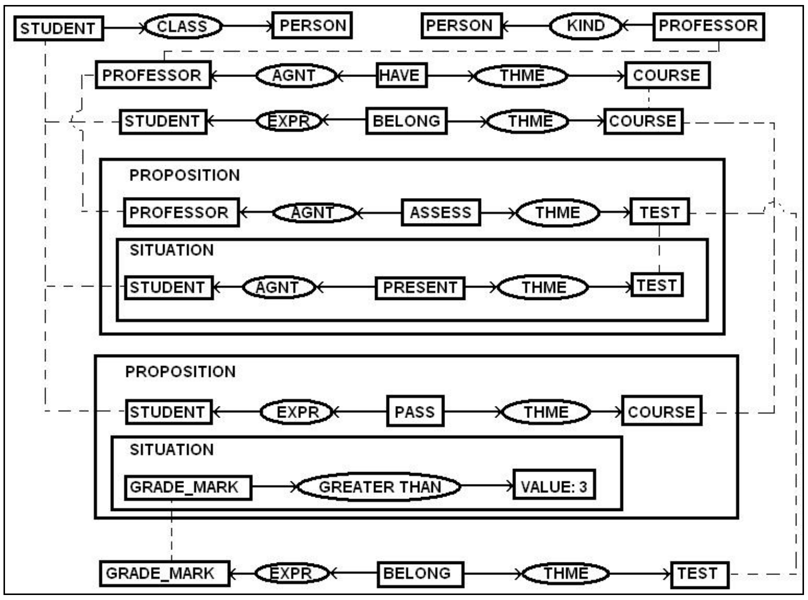

Fig. 4. Conceptual Graph of the example discourse.
Agnt stands for Agent, *Thme* for Theme, Expr for Experiencer.

course (Figure 4). Then we apply the rules described in Section 4.4 for obtaining three different UML diagrams (Figures 5 to 7). Here is an example of the discourse:

Student is a type of person.
Professor is a kind of person.
Professor has course.
Student belongs to course.

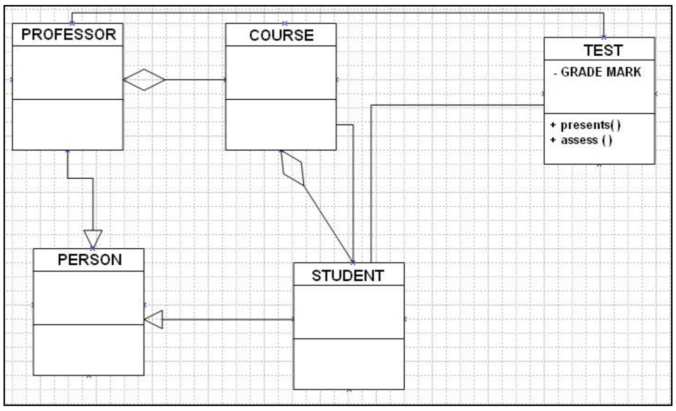

Fig. 5. Class Diagram obtained from PS

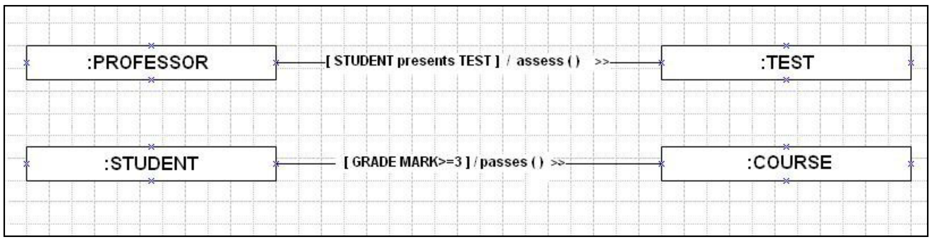
:PROFESSOR
[STUDENT presents TEST] / assess ( ) >>
:TEST
:STUDENT
[GRADE MARK>=3] / passes ( ) >>
:COURSE

Fig. 6. Communication Diagram obtained from PS

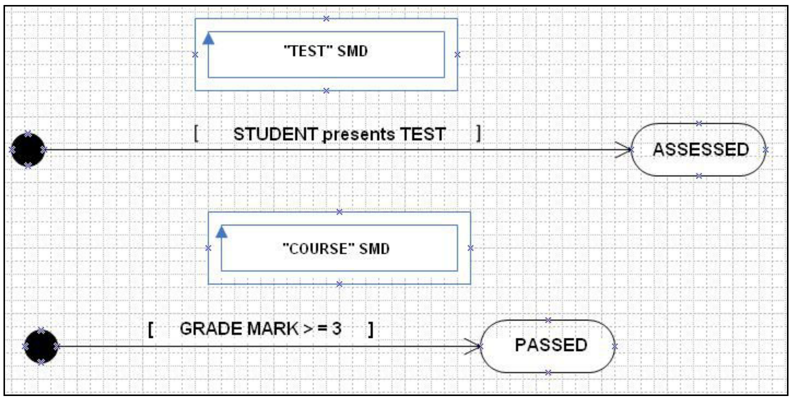

Fig. 7. State Machine Diagrams obtained from PS

After student presents test, professor assess test.
If grade mark is greater than 3 then student passes course.
Grade mark belongs to test.

We have developed a CASE Tool named UNC-Diagrammer for constructing Pre-conceptual Schemas and transforming them into Class, Communication, and State Machine UML diagrams.

6 Conclusions and Future Work

We have presented a framework based in Pre-conceptual Schemas, a Conceptual-Graph-like Knowledge Representation for automatically acquiring UML Diagrams from controlled natural language discourse. Namely, PSs are obtained from UN-Lencep, a controlled language for the specification of Pre-conceptual Schemas. We have shown the use of this framework with an example. The obtained UML diagrams are consistent with respect to each other because they are obtained from the same PS that represents the Stakeholder's discourse expressed in UN-Lencep.

In comparison with Conceptual Graphs, Pre-conceptual Schemas have many advantages: unambiguous syntax, integration of concepts, dynamic elements, and proximity to the Stakeholder language. Compared with other KR languages for requirements elicitation, PS are superior in that they do not require technical training from the Stakeholder and there is a framework for automatically building UML diagrams.

Some work is still to be done to improve this KR formalism:

- Improvements to completeness of the rules to build more types of diagrams and more elements of the existing diagrams;
- Integration of UN-Lencep into the UNC-Diagrammer CASE tool;
- Enrichment of UN-Lencep in order to make it closer to unrestricted natural language;
- Enrichment of Pre-conceptual Schema syntax for including other linguistic elements, such as articles.

References

[1] Sowa, J.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks/Cole, Pacific Grove (2000).

[2] Brewster, Ch., O'Hara, K., Fuller, S., Wilks, Y., Franconi, E., Musen, M., Ellman, J., and Shum, S.: Knowledge Representation with Ontologies: The Present and Future. IEEE Intelligent Systems, vol. 19, No. 1 (2004) 72–81.

[3] Sowa, J. F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Publishing Co., Reading (1984).

[4] Knowledge Interchange Format. Draft proposed American National Standard (dpANS) NCITS.T2/98-004. logic.stanford.edu/kif/dpans.html

[5] Leite, J.: A survey on requirements analysis, Advanced Software Engineering Project. Technical Report RTP-071, Department of Information and Computer Science, University of California at Irvine (1987).

[6] Cook S. C., Kasser J.E., and Asenstorfer J.: A Frame-Based Approach to Requirements Engineering. Proc. of 11th International Symposium of the INCOSE, Melbourne (2001).

[7] Dubois, E., Hagelstein, J., Lahou, E., Ponsaert F., and Rifaut, A.: A Knowledge Representation Language for Requirements Engineering. Proceedings of the IEEE, 74, 10 (1986) 1431–1444.

[8] Hagelstein, J.: Declarative Approach to Information Systems Requirements. Knowledge Based Systems 1, 4 (1988) 211–220.

[9] Greenspan. S.: Requirements Modeling: A Knowledge Representation Approach to Software Requirements Definition. PhD thesis, Dept. of Computer Science, University of Toronto (1984).

[10] Greenspan, S., Mylopoulos, J., and Borgida, A.: On Formal Requirements Modelling Languages: RML Revisited. IEEE Computer Society Press, Proceedings of the Sixteenth Intl. Conf. on Software Engineering, Sorrento (1994) 135–148.

[11] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M.: Telos: Representing Knowledge about Information Systems. Transactions on Information Systems 8, No. 4 (1990). 325–362.

[12] Jeusfeld, M.: Change Control in Deductive Object Bases. INFIX Pub, Bad Honnef (1992).

[13] Tsai, J., Weigert, Th., and Jang, H.: A Hybrid Knowledge Representation as a Basis of Requirement Specification and Specification Analysis. IEEE Transactions on Software Engineering, 18, No. 12 (1992). 1076–1100.

[14] Ramos, J. J. PML–A modeling language for physical knowledge representation. Ph.D. Thesis, Universitat Autònoma de Barcelona (2003).

[15] Pulman, S.: Controlled Language for Knowledge Representation. Proceedings of the First International Workshop on Controlled Language Applications, Leuven (1996) 233–242.

[16] Fuchs, N. E. and Schwitter, R.: Attempto Controlled English (ACE). Proceedings of the First International Workshop on Controlled Language Applications, Leuven (1996).

[17] Polajnar, T., Cunningham, H., Tablan, V. and Bontcheva, K.: Controlled Language IE Components Version 1. EU–IST Integrated Project (IP) IST–2003–506826 SEKT, D2.2.1 Report, Sheffield (2006).

[18] Delugach, H. and Lampkin, B.: Acquiring Software Requirements As Conceptual Graphs. Proceedings of the Fifth International Symposium on Requirements Engineering, Los Alamitos (2001).

[19] Alonso-Lavernia, M., A. De-la-Cruz-Rivera, G. Sidorov. *Generation of Natural Language Explanations of Rules in an Expert System.* LNCS N 3878, Springer, 2006, 311-314.

[20] Heidegger. M.: Protokoll zu einem Seminar über den Vortrag "Zeit und Sein". Zur Sache des Denkens, Tübingen (1976) 34.

[21] Piaget, J.: The origins of intelligence in children (2nd ed.). New York: International Universities Press (1952).