World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:4, No:2, 2010

The Spiral_OWL Model – Towards Spiral Knowledge Engineering

Hafizullah A. Hashim, and Aniza. A

***Abstract*—**The Spiral development model has been used successfully in many commercial systems and in a good number of defense systems. This is due to the fact that cost-effective incremental commitment of funds, via an analogy of the spiral model to stud poker and also can be used to develop hardware or integrate software, hardware, and systems. To support adaptive, semantic collaboration between domain experts and knowledge engineers, a new knowledge engineering process, called Spiral_OWL is proposed. This model is based on the idea of iterative refinement, annotation and structuring of knowledge base. The Spiral_OWL model is generated base on spiral model and knowledge engineering methodology. A central paradigm for Spiral_OWL model is the concentration on risk-driven determination of knowledge engineering process. The collaboration aspect comes into play during knowledge acquisition and knowledge validation phase. Design rationales for the Spiral_OWL model are to be easy-to-implement, well-organized, and iterative development cycle as an expanding spiral.

***Keywords*—**Domain Expert, Knowledge Base, Ontology, Software Process.

I. Introduction

A software process model is defined as a set of activities, methods, practices, and transformations that people used to develop and maintain software and its associated product [1]. It is viewed as a vehicle to improve software quality as well as productivity. The primary functions of a software process model are to determine the order of stages involved in software development and evolution and to establish the transition criteria from one stage to the next stage. These include completion criteria for the current stage plus choice criteria and entrance for the nest stage. Consequently, a process model differs from a software method (often called a methodology) in that a method's primary focus is on how to navigate through each phase (determining data, control, or "uses", hierarchies; partitioning functions; allocating requirements) and how to represent phase products (structure charts; stimulus-response threads; state transition diagrams).

This paper describes the Spiral_OWL model which is inspired by the Spiral development model for development of knowledge-base systems. The paper focuses on the designing Spiral_OWL which is and adoption of Spiral development process with knowledge engineering methodology. Its use ontology as knowledge base and shows the development of ontology based on Spiral_OWL.

Authors are with Faculty of Computer Science and Information Technology, University Malaya, 50603 Kuala Lumpur, Malaysia (e-mails: tringliserida@yahoo.com, noraniza@um.edu.my).

The reminder of this paper is organized as follows. Section II will firstly clarify the existing approach of knowledge engineering process and Spiral model. Section III gives a brief overview of a proposed Spiral_OWL. Section IV presents Spiral_OWL activities. Finally, Section V concludes the paper with additional comments and future work.

II. Existing Approaches

Related approaches can be roughly classified into two groups. Accompanied by the formation of knowledge engineering as an independent field of research, several knowledge engineering methodologies were developed. Most of them are much inspired by Software Engineering methodologies. In the Software Engineering domain, in the 1988, the Spiral development model is emerged. This software development is family of software development processes characterized by repeatedly iterating a set of elemental development processes and managing risk so that it is actively being reduced.

A. Knowledge Engineering

Knowledge engineering is the process of designing and producing knowledge-base system (KBS), and is so called to distinguish it from software engineering or the production of information systems. There are two approaches of Knowledge Engineering namely, transfer approach and modeling approach [2].

i. Knowledge Engineering as a Transfer Process

In early 1980s the development of KBS was seen as a transfer process of human knowledge into an implemented knowledge base. This transfer was based on the assumption that the knowledge which is required by the KBS already exists and just to be collected and implemented. Most often, the required knowledge was obtained by interviewing experts on how they solve specific tasks [3]. Typically, this knowledge was implemented in some kind of production rules which were executed by an associated rule interpreter. The transfer approach was only feasible for the development of small prototypical systems, but it failed to produce large, reliable and maintainable knowledge bases.

ii. Knowledge Engineering as a Modeling Process

Nowadays there exists an overall consensus that the process of building a KBS may be seen as a modeling activity. Building a KBS means building a computer model with the aim of realizing problem-solving capabilities comparable to a domain expert. It is not intended to create a cognitive adequate model, but to create a model which offers similar results in problem-solving for problems in the area of concern. While the expert may consciously articulate some parts of his or her knowledge, he or she will not be aware of a significant part of this knowledge since it is hidden in his or her skills. This knowledge is not directly accessible, but has to be built up and structures during the knowledge acquisition phase. Therefore, this knowledge acquisition process is no longer seen as a transfer of knowledge into an appropriate computer representation, but as a model construction process [4], [5]. There are three well-known modeling frameworks which have been developed in recent years namely, CommonKADS [6], MIKE [7] and Protégé-II [8].

The main goal of Knowledge Engineering is to structure the development and use of knowledge bases. For that purpose, the most widely known Knowledge Engineering approaches (e.g., CommonKADS [9]) are based on the ontology paradigm [10]. The development of both ontologies and adequate reasoning algorithms is supported by various methodologies, the phases and models of which resemble traditional Software Engineering approaches. The definition of Knowledge Engineering methodology is an agreement of how multiple people will work together. It defines a process in which domain experts and knowledge engineers will build a knowledge base. This knowledge base is represented in a knowledge representation language with suitable tools. Processes, languages and tools are based on knowledge representation paradigms. These Knowledge Engineering methodologies now also reveal similar problems to traditional Software Engineering approaches.

Significant initial efforts are needed to make the purpose of final ontology explicit and to deduce an appropriate model. It is often hard to estimate the required level of detail for the knowledge structuring a priori. Changes to the knowledge structuring are difficult and costly. For these reasons, methods from Knowledge Engineering are often too expensive to apply rarely used in practice [11]. However, for ontology construction the need of new approach that integrate Knowledge Engineering methodology and spiral model called Spiral_OWL.

B. *Spiral Model*

The spiral model of software development and evolution represents a risk-driven approach to software process analysis and structuring. This approach developed by Barry Boehm [12], incorporates elements of specification-driven, prototype-driven process methods, together with the classic software life cycle.

The model reflects the underlying concept that each cycle involves a progression that addresses the same sequence of steps, for each portion of the product and for each of its levels of elaboration, from overall concept of operation document down to the coding of each individual program. Each cycle involves traversing through the four quadrants. The first quadrant is to determine objectives, alternatives, and constraints for the cycle. The second quadrant is a risk analysis and evaluation of alternatives for the cycle. The third quadrant is to develop and verify the next level product. The fourth quadrant involves planning for the next phases. Each cycle of the spiral model iterates through these four quadrants. The number of cycles is project-specific, so the description of the activities in each quadrant is intended to be general enough so that they can be included in any cycle. The radial dimension in Fig. 1 represents the cumulative cost incurred in accomplishing the steps to date, and the angular dimension represents the progress made in completing each cycle of Spiral.

The goal of the spiral model is that the software process be risk driven, so that the risks within a given cycle are determined during the Analyze Risks quadrant. In order to manage these risks, certain additional project-specific activities may be planned to address the risks, such as Requirement Prototyping, if the risk analysis indicates that the software requirements are not clearly understood. These project-specific risks are termed process drivers.

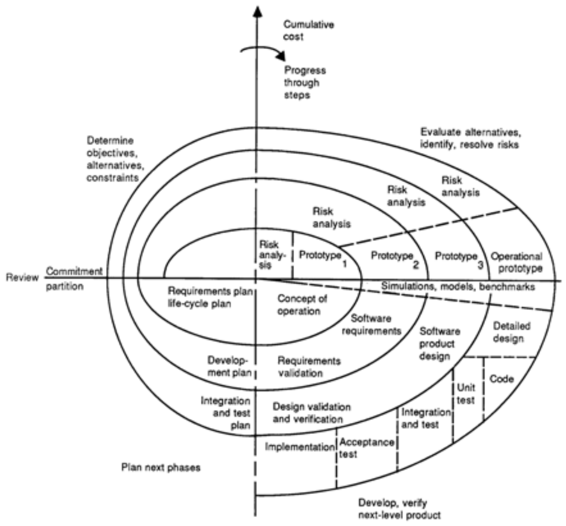

Fig. 1 The original spiral development diagram

For any process driver, one or more project-specific activities must be performed to manage the risk. Fig. 1 shows the original Spiral development diagram.

III. The Spiral_owl Model

The Spiral_OWL model is presented in this document is an adoption of Spiral model. Fig. 2 summarizes the important ingredients of Spiral_OWL which is generated based on knowledge engineering methodology and spiral model. The radial dimension in Fig. 2 represents the cumulative cost incurred in accomplishing the steps to date, and the angular dimension of Spiral_OWL represents a Knowledge Engineering methodology.

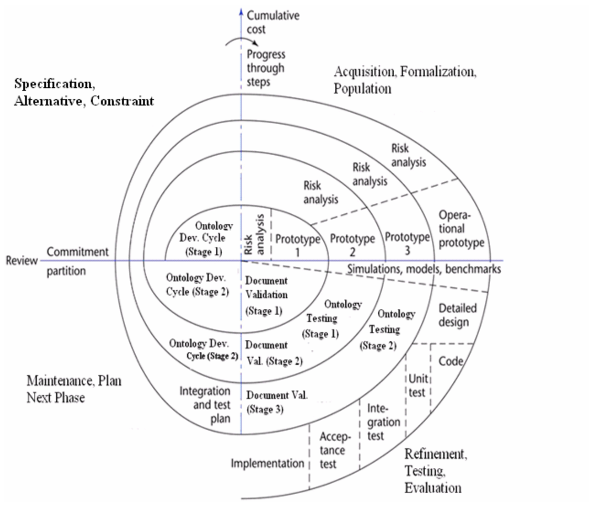
Cumulative cost
Progress through steps
Specification, Alternative, Constraint
Acquisition, Formalization, Population
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Ontology Dev. Cycle (Stage 1)
Prototype 1
Prototype 2
Prototype 3
Operational prototype
Review
Commitment partition
Simulations, models, benchmarks
Ontology Dev. Cycle (Stage 2)
Document Validation (Stage 1)
Ontology Testing (Stage 1)
Ontology Testing (Stage 2)
Detailed design
Ontology Dev. Cycle (Stage 2)
Document Val. (Stage 2)
Code
Unit test
Maintenance, Plan Next Phase
Integration and test plan
Document Val. (Stage 3)
Integration test
Acceptance test
Implementation
Refinement, Testing, Evaluation

Fig. 2 The spiral_OWL diagram

A. *A Typical Cycle of Spiral_OWL*

Each cycle of the spiral begins with the identification of specification of the ontology being developed (domain, functionality, target user, etc.), the alternative means of implementing this ontology (design T-Box, design A-Box, Special Type Property, Complex Class, ontology reuse, etc.), and the constraints imposed on the application of the alternatives (cost, schedule, resources, etc.)

The next step is to evaluate the alternatives relative to the specification and constraints. Frequently, this process will identify domain expert for knowledge acquisition, formalize knowledge being captured, and populate the knowledge into ontology form that are significant sources of project risk. If so, the next step should involve the formulation of a cost-effective strategy for resolving the sources of risk. This may involve administering domain experts' questionnaires, interviews subject matter experts, prototyping, and benchmarking.

Once the risks are evaluated, the next step is determined by the relative remaining risks. If risk of knowledge for certain domain strongly dominate program development and design of ontology risks, the next step may be an evolutionary development one: a minimal effort to specify the overall nature of the product, a plan for the next level of prototyping, and the development of a more detailed prototype to continue to resolve the major risk issues. If this prototype is operationally useful and robust enough to serve as low-risk base for future product evolution, the subsequent risk-driven steps would be the evolving series of evolutionary prototypes going toward the right in Fig. 2. In this case, the option of writing specifications would be addressed but not exercised. Thus, risk considerations can lead to a project implementing only a subset of all the potential steps in the model.

On the other hand, if previous prototyping efforts have already resolve all of the designing T-Box (classes, properties, relations) and A-Box (instances) for particular stage, the next step follows the basic waterfall approach (refinement and testing ontology for every stages) modified as appropriate to incorporate incremental development. Each stage of ontology testing in the figure is then followed by a validation step and preparation of plans for the succeeding cycle. In this case, the options to interview domain expert, prototyping, questionnaires, etc. are addressed but not exercised, leading to the use of different subset of steps.

This risk-driven subsetting of the Spiral_OWL model steps allows the model to accommodate any appropriate mixture of a specification-oriented/domain-oriented, prototype-oriented, questionnaires-oriented, and interviews-oriented to approach to ontology development. In such cases, the appropriate mixed strategy is chosen by considering the relative magnitude of the program risks and the relative effectiveness of the various techniques in resolving the risks. In a similar way, risk management considerations can determine the amount of time and effort that should be devoted to such other ontology activities as planning, designing, quality assurance, formal verification, and testing. In particular, risk-driven specifications can have varying degree of completeness, formality, and granularity, depending on the relative risks of doing too little or too much ontology specification.

An important feature of the Spiral_OWL model, as with most other knowledge engineering models, is that each cycle is completed by a review involving the Knowledge Engineer and Subject Matter Expert concerned with the ontology. This review covers all ontologies developed during each cycle, including the plans for the next cycle and the resources required to carry them out. The review's major objective is to ensure that all the knowledge is populated properly according to result of interviews. At the same time to ensure that all concerned parties are mutually committed to the approach for the next phase.

B. *Initiating and Terminating the Spiral_OWL*

Four fundamental questions arise in considering this presentation of the Spiral_OWL model:

1. How does the Spiral_OWL ever get started?
2. How do you get off the Spiral_OWL when it is appropriate to terminate project early?
3. Why does the Spiral_OWL end so abruptly?
4. What happen to ontology enhancement (or maintenance)?

The answer to these questions involves and observation that the Spiral_OWL model applies equally well to design and development of ontology or maintenance efforts. In either case, the Spiral_OWL gets started by a hypothesis that operational mission (e.g., an ontology development process is an iterative means) could be improved by a software effort.

The Spiral_OWL process then involves a test of this hypothesis: at any time, if the hypothesis fails the test (for example, if broken knowledge cause incorrect output to miss its market window, or if ontology is being developed becomes available), the Spiral_OWL is terminated. Usually, experience with the operational mission leads to further hypothesis about ontology improvements, and a new maintenance Spiral_OWL is initiated to test the hypothesis. Initiation, termination, and iteration of the tasks and ontologies of previous cycles are thus implicitly defined in the Spiral_OWL model (although they are not included in Fig. 2 to simplify its presentation)

IV. Spiral_owl Activities

The remainder of this paper expands the activities of Spiral_OWL by describing five essential aspects that every proper Spiral_OWL process must exhibit. The essentials are sketched in Fig. 2.

A. Spiral_OWL Essential 1 (Ontology Specification, Alternatives, Constraints)

The first step of knowledge engineering process is ontology specification. The ontology specification is needed to specify what is expected of the ontology. This specification determines the purpose and the domain of the ontology. The purpose of ontology can be set by listing typical query that the ontology has to answer or by describing a usage scenario. Several questions need to be answer before developing ontology so as to ensure the right ontology develops to the right application that will deliver to the right user. Such questions are i) why is the ontology being built? ii) what is its intended used iii) who are its users? Besides answering all the questions, we have to identify who is going to be a domain expert in the project.

B. Spiral_OWL Essential 2 (Acquisition, Formalization, Population)

Most of the ontology development methods conduct the ontology acquisition in a subjective manner. They generate concepts either by brainstorming (i.e., randomly enumerating a list of terms and then figuring out how they are related to each other), or by interviewing with experts. The first approach may be effective in creating ontologies for simple domains with shallow knowledge. However, it is not feasible in developing broader as well as complex knowledge. The second approach may be appropriate if the ontology is built based upon the knowledge in a small domain, such as company. However, the content of the ontology may be skewed and limited.

Here, there are three different stage of acquisition process: i) determining the scope of the ontology; ii) selecting a method to capture the ontology and iii) defining the concepts in the ontology. Determining the concept involves identifying all the key concepts and relationships. This can be achieved by sketching a list of questions named competency questions that a knowledge base based on the ontology should be able to answer. The competency questions will differ depending on which type of ontology being built. For domain ontologies, the competency questions are formulated so that they can be used to check at each stage of ontology construction whether the correct relationships have been created between the concepts, and whether the relationships created sufficiently describe the domain. To define competency questions, some pre-conceptions about which concepts are core to describing the domain are required.

When most of the knowledge has been acquired, it is unstructured and needs to be organized and structure by using representations that both computers and humans can understand. Such representations are named “knowledge worksheet” [13]. They are formatted templates and independent of the ontology engineering tools or implementation languages used. Basically, there are two types of worksheets: taxonomy worksheets and relationship worksheets. Each of taxonomy corresponds to a taxonomy worksheet while each concept, in general has a relationship worksheet. The taxonomy worksheet is used in organizing the unstructured results from the concept acquisition into a hierarchical structure. In our experience, this is the most challenging step of the overall development process. For example, different knowledge may classify the same taxonomy or concept from different perspectives and therefore have to be merged carefully. For instance, in agriculture field, we have to determine the growing stage of crops. Here, there are several stages need to classify either those stages is a concept or instance. In order to classify it, we have to describe it as a big picture not at the particular level. However, performance is one of the main issues that need to consider so as classifying the knowledge or resources.

While formalize the ontology, knowledge engineer is encourage to record the linguistic definition of term (e.g. noun, verb) as an intermediate step to identifying which terms are concepts in the ontology and which are relationships terms. The nouns are more likely to be concepts and verbs are most likely to be relationships terms. Note that naming conventions are applied in order to the ontology more readable and make each concepts, relationships, and instances unambiguous. The naming conventions require that each concept or class name use capitals for beginning class names and there can be a CamelCase brand name like “CropStage”, while properties or relationship start with lower case letters like “hasCropStage”. In fact, UpperCamelCase be used for a classes and instances, and lowerCamelCase be used for properties. Table I and II show more details of taxonomy worksheet and relationship worksheet respectively.

TABLE I
TAXONOMY WORKSHEET

Taxonomies	Number of Concepts	Example of Concepts	Acquisition Resources	Examples of Acquisition Resources
Disorder	1	PhysicalDisorder	Interview with domain expert	Questionnaires
Environemt Factor	3	SoilFactor, WaterFactor, WeatherFactor	Agriculture Handbook	[21][31]
Management Control	2	DiseaseControl, HormoneControl	Same as Disorder taxonomy	Same as Disorder taxonomy
Crop Stage	2	SeedStage, Vegetative Stage	Online resources	www.crop.com.my

Population is a process to construct a concept network from knowledge captured in acquisition process that describes the domain in question. A concept visualizes an ontology as nodes (concepts) and links (relationships between concepts). This is much more than Gomez-Perez's "Concept Classification Trees" [14] which organize domain concepts in taxonomies.

TABLE II
RELATIONSHIP WORKSHEET

Relationship	(Concept*,	Filler Concept)	Definition of relationship	Examples
crops:controlBy	crops:Pest, crops:Disease	crops:DiseaseControl, crops:PestControl	Describe on how to overcome disease and pest for tomato	**crops:controlBy** (crops:RussetMites, crops:SulfurOrCalciumPolysulphide)
crops:hasBotanicalInformation	crops:CropName	DP	Describe the type of crop like cultivar, variety, species, and general information	**crops:hasBotanicalInformation** (crops:Tomato, crops:RedRock)
crops:hasDisease	crops:CropName	crops:Disease	Describe the potential disease that may attack the tomato crop.	**crops:hasDisease** (crops:Tomato, crops:BacterialCancer)
crops:hasDisorder	crops:CropName	crops:Disorder	Describe the disorder that affect the functioning of the plant system	**crops:hasDisorder** (crops:Tomato, crops:LeafTipScorching)
crops:hasExpectedResult	crops:SystemSelection	crops:SystemExpectedResult	Describe the expected result of the selected system	**crops:hasExpectedResult** (crops:SoilessCulture-Fertigation, crops:HighYield)

C. Spiral_OWL Essential 3 (Refinement, Testing, and Evaluation)

Refinement process consists of two phase's namely intra-coding refinement, and extra-coding refinement. Intra-coding refinement involves the refinement done during the coding phase. As the code is being developed, if either some errors are discovered or new requirements come up, the code is refined to correct the errors or fulfill the new requirements. Extra-coding refinement refers to the changes done to overcome the errors that are uncovered during testing, and enhancements carried out during maintenance. Forms can be customized to form a refinement knowledge-acquisition tool; further design problems in the original ontology may surface.

In testing process, it uncovers defects in functional logic and implementation, and is carried out at all stages of development. Once the knowledge base has been created, end-user tests should be carried out to uncover defects in the ontology. Depending upon the problems encountered, appropriate changes need to be carried out to the ontology. In addition, full application will be test with end-users. This step can lead to further revisions to the ontology and knowledge acquisition form (questionnaires).

In evaluation process, knowledge engineer should firstly check whether all information captured during interview with domain expert has been captured as triples or restrictions or constraints in the concept network, or has been recorded as information loss. Secondly, they should check that the information captured in during interview session has been captured in black and white (paper). If there is information missing from the paper, further checks should be made against with domain expert in term of scope and purpose. Knowledge engineer can now evaluate their conceptual ontology against the following criteria:

- Logical consistency: Checks are made for repetition and missing triples. The competency questions can be used to identify core concepts, relationships, and triples that have not been captured.
- Conceptual accuracy: The domain expert should agree with the information that has been captured as triples, in that it represents his/her interpretation of the domain, task or application.
- Minimal ontological commitment: Only those relationships suited to the purpose and within scope have been created, i.e. the core concepts are well defined by their explicit relationships to other concepts and relations to their characteristics. Secondary concepts have only been used in the ontology to describe the core concepts.
- Clear differentiation between ontologies: The concepts and relationships captured should be suited to the ontology type created (i.e. domain ontology does not contain concepts more suitable to ontology task).

Vagueness has been handled well: Knowledge engineer has attempted to capture probability, possibility, uncertainty and fuzziness within the conceptual ontology.

D. Spiral_OWL Essential 4 (Maintenance and Plan for the Next Phase)

Maintenance process consists of three types namely, corrective, adaptive, and perfective [15]. Corrective maintenance involves considering the problem faced by the users while querying the ontology and correcting the ontology to overcome these problems. Adaptive maintenance involves modifying the ontology to fulfill new requirements in the future. Perfective maintenance involves improving the ontology, to further refine it.

Based on information given by domain expert and user, we plan for the next phase. The future ontology might be improved with more thus the system will be more intelligent. This would happen if knowledge engineer keep enrich the knowledge and keep finding new knowledge from several people who work in the same domain.

E. Spiral_OWL Essential 5 (Document Validation and Ontology Validation)

There are two types of validation will be conduct during design and development ontology. Document validation is vital so as to validate the knowledge that has been captured during interview sessions. This document contains all structured knowledge in form of table format. Domain expert has to validate the document base on questionnaires and its respective answers. Normally, the document is validated after knowledge engineer reorganizes the knowledge and before they start populates the knowledge into ontology form.

Ontology validation on other hand is validating the structure of ontology by senior knowledge engineer. This kind of validation called code inspection. During the validation, all the T-Box and A-Box is verified. T-Box design must be in line with the core knowledge acquired from domain expert whereas A-Box must be inserted to the respective class. Ontology validation is significant in order to reduce inconsistency data thus will ensure the knowledge is accurate and meet the user need.

V. Conclusion

The knowledge engineering process is susceptible to risks, and knowledge engineers wish to mitigate those risks by selecting the most appropriate process model for a project. The Spiral_OWL model has been proposed for ontology construction. The model is an adoption of Spiral model so that suitable for knowledge engineering environment.

The Spiral_OWL is a risk-driven process model, which depending on specific risks associated with a given project, may be tailored to create a project specific process model.

This paper presented an iterative-cyclic knowledge engineering process for building ontology which integrates solution proposals developed to date for overcoming the specification, communication and optimization barriers based on the notion of an optimization cycle. The optimization cycle can be subdivided into four regions: the region where the specification is determined (Quadrant I), the region of acquisition, formalization and population (Quadrant II), the region of refinement, testing and evaluation (Quadrant III) and the region of maintenance and plan for the next phase (Quadrant IV).

Spiral_OWL is essential for development of ontology as it's an iteration process. Furthermore, the knowledge is strictly relying on domain expert or in other word, Subject Matter Expert (SME). So, by all means, we need the iteration process model as knowledge comes into phase by phase. Further worthwhile research has been carrying out with focus on how to fully utilize this process model so as to produce a comprehensive ontology.

References

[1] A. Gomez-Perez, M. Fernandez-Lopez, and M. De Vicente, Towards a Method to Conceptualize Domain Ontologies. *In working notes of the workshop on Ontological Engineering, ECAI'96*, pp. 41-52, ECCAI 1996.

[2] A. R. Puerta, J.W. Egar, S. W. Tu, M. A. Musen, A Multiple-Method Knowledge Acquisition Shell for the Automatic Generation of Knowledge Acquisition Tools, *Knowledge Acquisition 4* (1992), pp. 171-196

[3] A. T. Schreiber, B. J. Wielinga, R. de Hoog, H. Akkermans, W. Van de Velde, CommonKads: A Comprehensive Methodology for KBS Development, *IEEE Expert* (December 1994), pp. 28-37

[4] B. Boehm, A Spiral Model of Software Development and Enhancement, *Computer*, May 1988, pp. 61-72

[5] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. V. de Velde, and B. J. Wielinga, Knowledge Engineering and Management: The CommonKADS Methodology, *MITpress*, 2000

[6] H. Knublauch, An Agile Development Methodology for Knowledge-Based Systems, *PhD thesis*, University of Ulm, 2002

[7] J. Angele, D. Fensel, R. Studer, Developing Knowledge-Based Systems with MIKE, *Journal of Automated Software Engineering*, in press

[8] J. M. David, J. P. Krivine, R. Simmons (eds.), *Second Generation Expert Systems* (Springer-Verlag, Berlin, 1993)

[9] K. Morik, Underlying Assumptions of Knowledge Acquisition as a Process of Model Refinement, *Knowledge Acquisition* 2(1), March 1990, pp. 21-49.

[10] M. A. Musen, An Overview of Knowledge Acquisition, in J.M. David et al. (eds.), *Second Generation Expert Systems* (Springer-Verlag, 1993)

[11] Paulk M. C., Curtis, B., Chrissis, M. B., Weber, C. V.(eds.): CMM Capability Maturity ModelSM for Software. *Version 1.1, Technical Report*, CMU/SEI (1993)

[12] R.S. Pressman, Software Engineering: A Practitioner's Approach, *3rd Ed., McGraw-Hill*, New York, NY, 1992

[13] T. R. Gruber, A Translation Approach to Portable Ontologies, *Knowledge Acquisition* 5(2), pp. 199-220, June 1993

[14] W. J. Clancey, The Knowledge Level Reinterpreted: Modeling How System Interact, *Machine Learning 4* (1989), pp. 285-291

[15] Zhanjun Li, Victor Raskin, and Karthik Ramani. (2007), A *Methodology of Engineering Ontology Development for Information Retrieval, International Conference on Engineering Design, ICED'07*. Paris, France. 28-31 August 2007.