From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment

Asunción Gómez-Pérez

Knowledge Systems Laboratory
Stanford University
701 Welch Road, Building C
Palo Alto, CA, 94304, USA
Tel: (415) 723-1867, Fax: (415) 725-5850
Email: gomez@hpp.stanford.edu, Asun@fi.upm.es

Abstract

There is no set of general guidelines to evaluate Knowledge Sharing Technology, specific ideas to evaluate user-independent ontologies in their own whole life cycle, whether their definitions are reused by KBS or they are shared among software agents. Instead of starting from the beginning, this paper discusses similarities and differences between knowledge bases and ontologies. The idea is to learn from Knowledge Base Systems' evaluation and assessment by picking up some successful ideas and adapting them to the domain of the ontologies. We also learn from its mistakes by avoiding them. The paper also describes how different agents that use ontologies with different aims have different concerns in the evaluation and assessment processes. Definitions of the terms: evaluation, verification, validation and assessment in the knowledge sharing domain are also given.

1. Introduction

The lack of mechanisms to evaluate Knowledge Sharing Technology (KST) is an obstacle to their use in commercial applications. While ontologies may provide for reusability, sharability or both, the evaluation of their definitions, software environments and documentation is critical to the success of the applications that reuse and share these definitions. For example, if wrong or incomplete definitions from the ontology coexist with specific knowledge formalized in the Knowledge Base (KB), the KBS may reach poor or wrong conclusions. If the ontology definitions and software environment have not been sufficiently evaluated, communication between software agents may not succeed.

The main problem appears to be: *there is no set of general guidelines to evaluate KST, specific ideas to evaluate user-independent ontologies in their own whole life cycle, whether their definitions are reused by KBS or they are shared among software agents*. This lack appears because of a vicious circle problem: almost nobody does evaluation of the KST because it is not still required; almost nobody requires evaluation of the KST because nobody knows how to perform it; and, finally, nobody performs evaluation of the KST because nobody did it before. To break this vicious circle, the following approach is taken in this work: If an ontology is like a KB, evaluation and assessment of the ontology should be done in the same way that KBS evaluation and assessment are performed. However, if they are different, a conceptual framework should be created to carry out these tasks in the knowledge sharing domain.

Since ontologies differ from KBs, the purpose of this paper is to take advantage of the evaluation, verification, validation and assessment in KBSs in order to learn from their successes and mistakes. The main idea is that it is not necessary to reinvent the wheel. We can learn from KBSs's successes by picking up some successful ideas and adapting them to the domain of the ontologies. We also can learn from KBS evaluation and assessment mistakes by avoiding them. The following sections describe the outcome of this research:

- In section Two, we summarize the differences and similarities between ontologies and KBs.
- Section Three discusses some KBS evaluation, verification, validation and assessment ideas.
- Section Four analyze what can be learn from KBSs's successes and mistakes.
- Section five describes three kinds of agents dealing with the evaluation of the knowledge sharing technology. It also covers: *What* each agent should evaluate; *When* and *Where* it should achieve its evaluation; and the benefits of the evaluation of the KST under different point of views.
- Finally, section Six proposes the definition of the terms: "evaluation", "verification", "validation" and "assessment" in the knowledge sharing domain.

2. KB vs. Ontologies

The word *ontology* is a fashionable word in the artificial intelligence community. Sometimes the term is used with the meaning of a KB because both define, represent, and gather knowledge about concepts and their relations in a machine readable language for an abstract or concrete domain. This section provides definitions of these concepts as well as differences and similarities between them.

2.1 Definitions

A KB is the knowledge module of a KBS. It contains abstract and specific knowledge of a particular subject in a machine-readable format. The knowledge in the KB can be declarative or procedural, shallow or deep. The inference engine works with the information gathered in the knowledge module to create intelligent behavior.

As libraries of domain-independent and abstract definitions that can be used for different purposes in different applications, Gruber [11] defines ontologies as an explicit *specification of a conceptualization* in Newell's conceptual level [23]. From the point of view of knowledge reusability, ontologies avoid the need to build a KB from scratch by allowing KBS developers to assemble reusable components [20]. Developing and implementing ontologies for reusing knowledge requires the development of methodologies, techniques and more powerful tools that integrate, in a coherent and consistent manner, their definitions into the entire life cycle of the KBS by developing technology that: (1) supports, checks and makes effective this integration; and (2) allows the integration of different solutions provided by different families of ontologies [10].

From the point of view of knowledge sharability, ontologies can be used by software agents [11, 13, 15, 20] that interoperate. From this point of view, many complementary definitions appear: Gruber [11] defines an ontology as the vocabulary with which queries and assertions are exchanged among agents; according to Gruber and Olsen [13], the ontology vocabulary defines the ontological commitments among agents that are agreements to use the shared vocabulary in a coherent and consistent manner; Guha and Lenat [15] view ontologies as foundational knowledge shared by agents to enable them to communicate their specialized knowledge; and Gómez-Pérez [10] sees ontologies as the platforms that reduce semantic differences among agents by establishing common vocabularies and semantic interpretations of terms.

2.2. Similarities and Differences

Ontologies and KBs have in common that both gather information that evolves over time. The term "knowledge" refers to the information in a KB, while "definition" refers to that of an ontology.

First, the *generality* of the information is the most important difference between ontologies and KBs. The definitions of ontologies should be more general than the knowledge of a KB, so that they can be shared among KBS. Ontologies' definitions must be *independent of the agent* (both KBS and software agent) that will reuse or share them. So, abstraction and independence are *the* features that should enable ontologies to be used to solve a huge variety of different problems in several and specific domains.

Second, the KB of a KBS is separated from its inference engine. The inference engine decides when and how the information of the KB must be used to derive new conclusions. Ontologies differ from KBs because *they don't usually have reasoning methods* that take advantage of the ontologies' definitions and make them more understandable. However, ontologies written in CycL [19] can be used by a number of special-purpose inference schemas.

Third, the expressiveness and the semantics of the target machine-readable language used in the formalization of a KB influence the quantity and quality of the knowledge gathered in a KB. Ontologies should be written in an *expressive, declarative, portable, domain-independent, and semantically well-defined machine-readable language* that is independent of the final target machine-readable language of the application that will reuse or share the definitions in the application domain. Ontolingua's ontologies [11] written in KIF [9] provide these features, and Ontolingua also provides the software environment that transforms KIF definitions to many different target languages that can be used to build KBS.

Fourth, object and strategic domain knowledge can be represented together in the KB of a KBS. While the former defines knowledge about the problem, the second provides to the inference engine knowledge about its resolution. In this sense, Albert [3] recommends *separating the object and strategic knowledge of a given ontology in different ontologies*. There are two problems related to the strategic knowledge in ontologies. The first is the difficulty representing strategic knowledge declaratively. The second appears because ontologies don't usually have reasoning methods related with the use of any kind of knowledge.

Fifth, a KBS reasons with the knowledge stored in the KB to carry out tasks that are performed by experts in specific domains. The evaluation of a KBS's performance requires an understanding of how experts act and how their performance can be evaluated. How can we measure performance of ontologies without an inference engine to work with? A first approach consists of evaluating the performance of the ontologies in terms of the number of successful KBS that reuse their definitions, or by the amount of successful interoperation among agents. A set of competency questions [14] could be used for evaluating the performance of the ontologies used in different tasks in some domain.

Two important additional problems in KBS are the absence of a well-defined and well-structured set of requirements at the beginning of the development process, and the continuous changes in the requirements during the whole life cycle of the KBS. These problems don't exist so sharply in ontologies because before building ontologies for a given domain a high level of abstraction about the useful vocabulary in the domain is required. So, *ontologies' requirements should be more complete and precise than KBS requirements are*. The iterative specification process of the requirements of an ontology allows one to include or delete definitions at the time that the ontology is built. For example, the first requirements for a domain-independent ontology about numbers could be: a list of kinds of numbers, a list of operations allowed among numbers, a list of comparisons between numbers, and constants used in this domain. Consequently, it would be easy to express these known requirements in a formal language, yielding a set of formal specifications for the ontology.

Finally, any engineering development requires the definition and standardization of a life cycle that goes from requirement definition to maintenance of the finished product. Examples of methodologies for KBS are: Waterman [34], Parsaye and Chignell [27], ICOT methodology [33], IDEAL [21], KADS [35], Alberico and Mico [2], Harmon and Sawyer [17], etc. *Ontologies*, as any engineering development, *need the definition and standardization of a life* *cycle and methodologies and techniques that drive their development*.

Using ontologies to build KBs by picking up definitions from different families of ontologies allows us to conclude that: (1) a KB is not an instance or a subclass of an ontology because the KB might or might not gather different definitions from several ontologies; (2) an ontology is not an instance or a subclass of a KB either because ontologies are abstractions of KBs. So, we can say that there is no instance or subclass relationship between ontologies and KBs. However, we could say that *some knowledge of a knowledge base comes from some ontologies.*

3. KBS: Evaluation, Verification, Validation and Assessment

A study of the evaluation, verification, validation and assessment ideas in KBS preceded the evaluation, verification, validation and assessment of KST in order to learn from KBS' successes and mistakes.

Technical evaluation "versus" user's evaluation. The majority of the authors [5, 16, 29, 31] consulted say that it's necessary to distinguish between evaluation of the intrinsic properties of a KBS from the evaluation of its actual use and utility within a given organization. O'Leary [26] and Guida and Mauri [16] call the former "evaluation" and the latter "assesment".

Terminology and Definitions. The analysis of Table 1 allows us to say that there are a set of terms related to the judge of a KBS, and they are: "evaluation", "verification", "validation" and "assessment". Although the set of common terms is identified, there is no consensus on the definitions. Looking at Table 1, we find that sometimes the same term is used with different and complementary meanings so it is easy for authors to misunderstand each other. In this line, Hoppe and Meseguer [18] comment on several definitions of "evaluation", "verification" and "validation", and they also make a proposal for a common terminology in KBS.

Criteria to evaluate, verify, validate and assess. Tables 2, 3, 4, and 5 allow us to say that the majority of the authors consulted give its criteria and methods to evaluate, verify, validate and assess KBS, some authors only offer criteria or methods (rather than both), and a few of them provide neither criteria nor methods. For those authors that offer criteria attached to a term, we can notice that often the criteria are almost the same (see evaluation, verification and assessment criteria at Table 2, 3, and 5), and sometimes they are complementary (see validation criteria at Table 4).

There are a large number of qualitative and quantitative criteria to evaluate, verify, validate and assess KBS. The majority of the authors define these criteria as independence of the language used to formalize the KB and as independence of the inference engine that works with it. Others make definitions depending on the formalism used to express knowledge in the KB. Dealing with evaluation, verification, validation and assessment criteria in KBS, the main dilemma is the absence of agreement in: (1) identification of the criteria attached to a given term; (2) definitions of the criteria; and (3) the phase of the KBS life cycle in which they should be applied. For instance, Guida and Mauri [16] propose to evaluate the dimensions of performance and quality of KBS by defining a collection of criteria related in a tree of components. Sharma and Conrath [31] analyze quality in the socio-technical system [7] and propose 39 dimensions of quality gathered in four categories: task, technology, people and organization.

Methods to evaluate, verify, validate and assess. For those authors that provide methods attached to a given criterion, we can say that the methods shown in Tables 2, 4 and 5 are complementary and independent of the formalism used to build the KB. However, the majority of the verification methods only works for KB formalized in rules, as it is shown in Table 3.

Author	Evaluation	Verification	Validation	Assessment
O'keefe, Balci, Smith (1987) [25]		Verification refers to building the system right, that is, substantiating that a system correctly implements its specifications.	Validation refers to building the right system, that is, substantiating that a system performs with an acceptable level of accuracy.	
Benbasat, Dhaliwal (1989) [4]	Evaluation by experts is to determine both the quality of the KB and of the advice that the KBS provide	Verification is the demonstration of consistency, completeness and correctness of the software at each stage and between each stage of the software development life cycle.	Validation should unfold as a sequence of stages paralleling the different stages of the KBS development life-cycle: .- *Conceptual Val.* .- *Elicitation Val.* .- *Implementation Val.* .- *Representational Val.* .- *Functional Evaluation.*	Evaluation by users is associated with both the usefulness and the usability of the KBS
Suen, Grogono, Shinghal, Coallier (1990) [32]	Evaluation is divided in two components: verification and validation.	Verification confirms that the expert system is logically consistent but does not guarantee that its domain - dependent knowledge agrees with that of the human expert.	The process of evaluating an Expert System during and after the development process to ensure compliance with the initial requirements.	The expert system must be acceptable to its intended users.
Preece (Nov., 1990) [29]	Evaluation covers methods for system verification, validation and user acceptance testing.	Determines the internal self-consistency and completeness of the system.	The system satisfactorily performs the real-world tasks for which it was created.	User acceptance tends to encapsulate ergonomic and organizational aspects of the system
Guida, Mauri (April, 1993) [16]	Evaluation of the intrinsic properties of a KBS	Determine whether the implemented KBS completely satisfies its specifications.	Determine whether the KBS satisfactorily performs the real-word tasks for which it was created	Evaluation of its actual use and utility within a given organization
Hope, Meseguer (June, 1993) [18]	Assesses or measures a KBS's quantitative and qualitative characteristics and compares them with expected or desired values. We can to evaluate the KB structure, inference engine features, etc.	Checks the well-defined properties of a KBS against its specification for particular KBS aspects like: .- KB .- Inference Engine .- User interface .- I/O behavior	Checks whether a KBS corresponds to the system it is supposed to represent in on particular KBS aspects like: .- KB .- Inference Engine .- User interface .- I/O behavior	

Table 1. Definitions in KBS

Methodologies and Tools. Evaluation of a KBS is an iterative process [5, 16, 25, 29] that is performed during all the different stages of its life cycle [4]. Suen and colleagues [32] restrict this iteration to the validation process. The lack of a complete, consistent, and precise definition of the KBS requirements forces Knowledge Engineer (KE) to carry it out from the beginning of the KBS development until the maintenance of the finished product. For this reason, evaluation must follow an order, it has to be planned, and it must be controlled to reduce the cost (time and money) of the final system.

Examples of integrated methodologies that evaluate KBS are: Geissman and Schult's iterative methodology [8] for formal validation and verification of expert systems based on the spiral model [6] of software engineering; Benbasat and Dhaliwal [4] provides a framework where validation evolves with the different stages of the KBS development life cycle; and Guida and Mauri [16] describe a general methodology for measuring performance and quality of a KBS. A tool called KVAT [22] integrates knowledge validation in the knowledge acquisition process, and UVT [28] is a unification-based tool for knowledge base verification written in rules.

Measures of the results. The evaluation of the results from empirical evaluation [29] can be done in an informal and qualitative manner, or by using quantitative methods like: the Kappa statistic [30], paired-t test [25], confusion matrices [27] and ROC curves [1]. While for O'keefe the paired-t test is a validation method, for Preece this method allows evaluation of the results of the evaluation process. So, there doesn't exist an agreement among the methods to evaluate the results of the evaluation process.

Author	Definition	Criteria	Methods
Benbasat, Dhaliwal (1989) [4]	Evaluation by experts is to determine both the quality of the knowledge-base and of the advice that the KBS provide.	*Quality*	
Suen, Grogono, Shinghal,Coallier (1990) [32]	Evaluation is divided in two components: verification and validation.		
Preece (Nov., 1990) [29]	Evaluation covers methods for system verification, validation and user acceptance testing.		
Berry, Hart (Nov. 1990) [5]	The success of a system does not just depend on the system matching user needs and supporting users in their tasks, but also on the match between the system and the social and political factors within the host organization.		*Usability* .- Interviews .- Questionnaires .- System walkthough .- Formal observation .- User diaries .- System Logging .- Simple Experiments
Guida, Mauri (April, 1993) [16]	Evaluation of the intrinsic properties of a KBS	*Performance and Quality*	P&Q factor (PQF) .- Evaluation function .- Measure function .- Combination function
Hope, Meseguer (June, 1993) [18]	Assesses or measures a KBS's quantitative and qualitative characteristics and compares them with expected or desired values. We can to evaluate the KB structure, inference engine features, and so on.		

Table 2. Evaluation definitions, criteria and methods

Author	Definition	Criteria	Methods
Nguyen, Perkins, Laffey, Pecora (1985) [24]		*Consistency* *Completeness*	*Consistency* .-Redundant Rules .- Conflicting Rules .- Subsumed Rules .- Circular Rules Chain *Completeness* .- Missing Rules .- Unreachable Clauses .- Dead-end Clauses
O'keefe, Balci, Smith (1987) [25]	Verification refers to building the system right, that is, substantiating that a system correctly implements its specifications.		
Benbasat, Dhaliwal (1989) [4]	Verification is the demonstration of consistency, completeness and correctness of the software at each stage and between each stage of the software development life cycle.	*Consistency* *Correctness* *Completeness*	
Suen, Grogono, Shinghal, Coallier (1990) [32]	Verification confirms that the expert system is logically consistent but does not guarantee that its domain-dependent knowledge agrees with that of the human expert.	*Logically consistent*	*Structure Checking* .- Inconsistency .- Redundancy .- Subsumption .- Cyclic dependencies *Semantic Checking* .- Range errors .- Cardinality Errors .- Illegal Values .- Incorrectness rules
Preece (Nov., 1990) [29]	Determines the internal self-consistence and completeness of the system.	*Consistency* *Completeness*	*Logical Methods* .- Conflictive, subsumed, and Redundant Rules .- Conflictive, subsumed, and Redundant Inferences Chain .- Cyclic inference chain .- Missing Rules, values and goals .- Useless Rules .- Dead-end Rules .- Unsatisfiable Conditions .- Unobtainable data items .- Data types classes
Guida, Mauri (April, 1993) [16]	Determine whether the implemented KBS completely satisfies its specifications.	*Ontology* .- Structure .- Content .- Soft. component .- System software .- Hard. system	
Hope, Meseguer (June, 1993) [18]	Checks the well-defined properties of a KBS against its specification in on particular KBS aspects like: .- KB .- Inference Engine .- User interface .- I/O behavior		

Table 3. Verification definitions, criteria and methods

Author	Definition	Criteria	Methods
O'keefe, Balci, Smith (1987) [25]	Validation refers to building the right system, that is, substantiating that a system performs with an acceptable level of accuracy.	*Performance*	*Qualitative Validation* .- Face Validation .- Predictive Validation .- Turing Test .- Field Test .- Subsystem Validation .- Sensitivity Analysis .- Visual Interaction *Quantitative Validation* .- Paired t-tests .- Hotelling's one-sample T^2 test .- Simultaneous Confidence Intervals .- Consistency measures
Benbasat, Dhaliwal (1989) [4]	Validation should unfold as a sequence of stages paralleling the different stages of the KBS development life-cycle: .- *Conceptual Validation* .- *Elicitation Validation* .- *Implementation Validation* .- *Representational Validation* .- *Functional Evaluation*	*Conceptual Validation* .- Quality *Elicitation Validation* .- Completeness .- Correctness *Implementation Validation* .- Quality *Representational Validation* .- Structural Match .- Behavioral Match *Functional Evaluation* .- Performance	*Conceptual Validation* .- Structured Walkthroughs .- Test Cases *Elicitation Validation* .- Domain and construct correspondence - Test for range correspondence .- Protocol Analysis .- Structured and logical walkthroughs .- Enumeration of cognitive-primitives .- Test Cases .- Inter-Expert validation .- Formal Checking *Implementation Validation* .- Turing Test .- Sensitive Analysis .- Formal Checking .- Degenerate Tests .- Extreme-value test .- Model-components test
Suen, Grogono, Shinghal, Coallier (1990) [32]	The process of evaluating an Expert System during and after the development process to ensure compliance with the initial requirements.	*Accurate* *Suitable*	*Test problems* *Questionnaires* *Sensitive Analysis*
Preece (Nov., 1990) [29]	The system satisfactorily performs the real-world tasks for which it was created.	*Accuracy*	*Empirical Method* .- Test cases .- Agreement Method .- Linear Model .- Turing Test
Guida, Mauri (April, 1993) [16]	Determine whether the KBS satisfactorily performs the real-word tasks for which it was created	*Static Behavior* .- Appropriateness .- Adequacy .- Reliability *Dynamic Behavior*	
Hope, Meseguer (June, 1993) [18]	Checks whether a KBS corresponds to the system it is supposed to represent in on particular KBS aspects like: .- KB .- Inference Engine .- User interface .- I/O behavior		*Testing* .- Pure testing .- Experimentation

Table 4. Validation definitions, criteria and methods

Author	Definition	Criteria	Methods
Benbasat, Dhaliwal (1989) [4]	Evaluation by users is associated with both the usefulness and the usability of the KBS	*Usable* *Useful*	
Suen, Grogono, Shinghal, Coallier (1990) [32]	The expert system must be acceptable to its intended users.	*Acceptability*	
Preece (Nov., 1990) [29]	User acceptance tends to encapsulate ergonomic and organizational aspects of the system	*Usability*	*Empirical Method* .- Test cases .- Questionnaires

Table 5. Assessment definitions, criteria and methods

4. What can we learn from KBSs's successes and mistakes?

Technical evaluation "versus" user's evaluation. A characterization of KST' users is needed in order to figure out what each one of them attempts to do with this technology. Section 5 identify three kind of agents dealing with KST.

Terminology and Definitions. We have to identify a set of terms in the KST domain and provide a standard definition for these terms. As a point of departure, Section 6 provides definitions of the terms "evaluation", "verification", "validation" and "assessment" in KST.

Criteria to evaluate, verify, validate and assess. A set of criteria and definitions of these criteria for KST will provide some guidance in performing its evaluation and assessment. For each term, we need to develop a set of criteria that allow us to identify several kinds of mistakes in KST. Section 6 offers some criteria attached to the previous terminology.

Methods to evaluate, verify, validate and assess. For each criterion, we recommend to establish a set of methods that allow us detect mistakes. Different methods for ontologies written in different languages should be provided.

Methodologies and Tools. For KST evaluators, the main idea to be taken from KBS evaluation, verification, validation and assessment is the need for developing and integrating methods and tools that allow one to perform an iterative evaluation and assessment of the KST during their whole life cycle.

Measures of the results. At the same time that ontologies are evaluated, measures of the results should be required to estimate the efficiency of the methods used.

5. Who evaluates Knowledge Sharing Technology?

Who evaluates knowledge sharing technology is tightly related to *what* can be evaluated, *when* to evaluate and *where* the activity is carried out. At the same time we are answering these questions, we discuss *why it* is important to evaluate this technology whatever its uses are.

Knowledge sharing technology can be evaluated by the ontology development team and end users (i.e., KBS development team and software agents development team). While the ontology development team evaluates technical properties of the ontologies, software and documentation, end users assess their actual utility and usability within a given organization or by other software agents.

5.1. Ontology development team evaluation

Before knowledge sharing technology leaves the academic or industrial lab, the ontology development team must guarantee to the end users the correctness and completeness of the ontologies' definitions, software and documentation. So, the ontology development team must evaluate:

1. Any intermediate definition or axiom, final definition or axiom, and collection of definitions or axioms belonging to one or more ontologies.

2. The software environment required to build, share and reuse definitions and axioms. It includes: the graphic and text user interface used to browse and edit definitions, parsers that check the syntax of the definitions and some basic semantics, translators for some target knowledge representation languages, software required to install the environment, and so on.

3. Documentation includes documentation of the ontologies, documentation of the software environments, tutorials, and examples of real applications built by using this technology.

While definitions and axioms should be evaluated against the ontology's requirements [10] or its competence questions [14], the software environments should be evaluated against their specifications. In order to detect as soon as possible wrong, incomplete, or missed definitions or axioms, or wrong, incomplete, or missed functionalities in the software environment and in its documentation, development team evaluation should iteratively perform this *technical* evaluation during the whole ontology life cycle and during the software environment's life cycle. Evaluation of other expert development teams would help to ensure the quality, the abstraction and the domain-independence of the definitions and axioms.

5.2. Final users assessment

The knowledge factory idea allows different people (whether they are related or not to the development team) not only to build their own ontologies but to put them to work in real applications. The number of successful applications that use knowledge sharing technology will be the measure of the success of this technology.

The opinions of the KBS and software agents development teams in favour or against the knowledge sharing technology will be the key factor in the perception that companies have about it. Developing and implementing ontologies for reusing and sharing knowledge requires assessment by the final users: KBS development team and software agent development team.

A) KBS Development team assessment

When KE decide to reuse components to build a KBS, they are looking for some definitions that simplify the development process of the new system and reduce its cost (in terms of time and money). Before reusing definitions and axioms from ontologies, the KE tean assessment is focused on:

1. The understanding, usability, abstraction, quality, granularity, and portability of the definitions and axioms given by the ontology.

2. Features of the software environment that help the team to understand as well as locate the definitions in the set of ontologies. Perhaps software agents like *Ontology-Brokers* will in the future help the KE to find, understand and reuse a particular definition.

3. Documentation (tutorial, case studies, etc.) that reduces the cost required to learn and assimilate this new technology.

4. Methodologies and tools that make possible the integration of the ontologies' definitions with the whole life cycle of the KBS.

5. Have the ontologies been technically evaluated?

Since the programming team makes effective the integration of the definitions and axioms with the knowledge of the KB, programming team assessment is a part of the assessment process. The opinion of the KBS development team in favor or against the knowledge sharing technology can be one of the key factors in the perception that companies have about this technology. So, *although a technically well-evaluated ontology won't guarantee the absence of errors in the integration of its definitions with the KB of a KBS, it will make the process easier.*

B) Software Agents Development Team Assessment

Since ontologies' definitions are the platforms that enable the sharing of knowledge among agents, inferences are always performed by the agent making the query and by the agent giving the answers. Given a technically well evaluated ontology, Software agents development team assessment is focused on: (1) the correctness of the protocols used in the communication between agents and the ontologies; and (2) how the software used in the communication between agents and ontologies solves inconsistencies, ambiguities, omissions, and errors between human or agent requests/answers and ontologies' definitions.

6. Definitions

This section proposes definitions of the terms: "evaluation", "verification", "validation", and "assessment" in the domain of KST.

Evaluation means to judge technically the features of KST with respect to a frame of reference[1] during each phase and between phases of its life cycle. In this paper, the term "evaluation" subsumes the terms "verification" and "validation", and it refers to the technical activity performed by the KST development team. The activities that the evaluation gathers are:

1. Evaluation of the ontologies' definitions and axioms, which includes:
   - Evaluation of each individual definition or axiom.
   - Evaluation of the set of definitions and axioms gathered in the ontology.
   - Evaluation of the definitions and axioms that are imported from other theories. The goal is to guarantee that these definitions don't alter the set of well-evaluated properties in the current theory.
2. Evaluation of the software environment, which includes software used to build, reuse, or share ontologies' definitions and axioms.
3. Evaluation of the documentation, which includes documentation of the ontologies and documentation of the software environments.

Verification refers to the iterative process that guarantees (during each phase and between phases of the life cycle) the *correctness* of an ontology, its associated software environments, and documentation with respect to their requirements[2] or competency questions [14] that specify the problem and what constitutes a good solution to the problem.

Ontologies Verification refers to building the ontology right, that is, insuring that the ontology correctly implements its requirements or its competence questions. Verification has to be donethroughout the ontology life cycle and has three goals.

1. To determine the *correctness of definitions and axioms* by figuring out what explicitly the ontology defines, doesn't define or defines incorrectly.
2. To determine the *scope of definitions and axioms* by figuring out what can be inferred, can not be inferred, or can be inferred incorrectly.

[1] A frame of reference can be: requirements specifications, competency questions [14], and the real-world.

[2] The requirements of an ontology include a list of terms and basic ontological commitments

3. The demonstration of a set of *well-defined attributes in definitions and axioms* [10] is related to: the coherence between the structure of a given definition and the design criteria of the environment in which it is included, the syntax and semantic of the definitions, and a set of logical properties like coherence, completeness, consistency, and conciseness.

Software Verification refers to building the software right, which means that the software that builds, reuses and shares definitions and axioms correctly and completely implements its requirements. Software engineering methodologies, techniques and tools provide the appropriate framework to verify the KST software in each stage and between stages of its life cycle.

Documentation Verification refers to building the documents right. It's aim is to guarantee that all the required documents are written, that nothing has been forgotten in each document, and that they evolve in step with definitions and software environments in each phase and between phases of their life cycle. If WWW documents are indexed automatically by using a program, the verification of the documentation must guarantee that there are no semantic inconsistencies, context and morphological mistakes in the indexes.

Validation refers to the iterative process that guarantees that the ontologies, the software development environment, and the documentation correspond to the systems that they are supposed to represent. So, validation compares characteristics of the KST with a frame of reference. Validation is a technical evaluation performed by the ontology development team in each stage and between stages of the KST life cycle.

Ontologies Validation refers to whether the *meaning* of the ontologies' definitions really represent the real world for which it was created. The validation of the ontologies against the frame of reference provides information about whether the ontology definitions are necessary and sufficient to represent the tasks and their solutions for different uses.

Software Validation refers to whether the *behavior* of the software environment adequately performs the tasks given in its requirements with an acceptable level of accuracy. Software engineering methodologies, techniques and tools provide the appropriate framework to validate the KST software in each stage and between stages of its life cycle.

Documentation Validation refers to whether the natural language documentation of the ontologies and the natural language documentation of the software environment have the *same meaning* that the meaning of the ontologies and software environments. Validation of the documentation must be performed in step with validation of the definitions and validation of the software environments during their life cycles.

Assessment refers to usability and utility of the ontologies, software environments, and their documentation when they are used within a given organization or by software agents.

Ontologies assessment refers to the understanding, usability, generality, granularity, quality, well-defined (both, logically and syntactically) properties, portability, incrementalism, maintainability and uniformity of the definitions and axioms given by an ontology.

Software assessment refers to the robustness, accuracy, portability or transferability, extendibility, reliability, computational efficiency of the software environments used to build, reuse and share ontologies.

Documentation assessment refers to whether the natural language documents are self-explanatory, that is, if they provide precisely and sufficiently information to learn and use efficiently this technology

Conclusions

The analysis of KBS evaluation and assessment allowed us to identify the main points in the evaluation and assessment of the knowledge sharing technology: technical evaluation versus user's evaluation, the need to provide a terminology and definitions, the need of identifying some criteria to perform the evaluation and assessment, the need of performing an iterative evaluation during the whole life cycle of the KST, and the need to build methods, methodologies and tools that support this process.

As KST is used for several agents, diverse kinds of evaluation are required. This paper described three kinds of agents dealing with knowledge sharing technology and how each agent has different concerns in the evaluation process; *what* should be evaluated by each agent; and *when* and *where* they should achieve their own evaluation.

We also proposed a set of terms and their definitions. While the term "evaluation" subsumes the terms "verification" and "validation" and refers to the activity performed by the ontology development team, the term "assessment" refers to the final user evaluation.

Future work

This study identified the need for evaluating ontologies and for a basic vocabulary and definitions. The paper also points out some future work:

- *To develop a set of criteria to evaluate (both verify and validate) and assess ontologies.* These criteria are tightly related to the person, place, time, and mistakes and omissions that we look for in the ontologies.

- *A set of iterative methods that are integrated with the ontologies' life cycle are required to establish how and when evaluation activities should be performed during their life cycle.* Although the iteration of the evaluation process is key idea in order to detect mistakes as soon as possible, the main question is: how can they perform an iterative evaluation during the whole ontologies life cycle if the ontologies' life cycle is not defined yet?

- *A set of evaluation tools is necessary to increase the performance of the evaluation process.*

- *Some recommendations and ontology-brokers that help end users to choose the best definitions for their systems are necessary.* So, some criteria that compare different definitions and axioms and that characterize the end user are required before performing these recommendations.

Acknowledgment

This work is support by the "Ministerio de Educacion y Ciencia" in Spain. Thanks to Bob Engelmore and Lee Brownston for their comments.

References

[1] Adlassing, K.P. *The Application of ROC Curves to the Evaluation of Medical Expert Systems*. In **Proc. 7th International Conference of the European Federation for Medical Informatics**. Rome, September 1987. Volume I. Edizioni Luigi Pozzi SRL. Rome. pp: 951-956.

[2] Alberico, R.; and Miko, M. **Expert Systems for Reference and Information RetrieValidation** UK: Mockler Corporation. 1990.

[3] Albert, M. **YMIR: an Engineering Design Ontology**. Doctoral Dissertation, University of Twente. Germany. 1993.

[4] Benbasat I; Dhaliwal, J.S. *A framework for the Validation of Knowledge Acquisition*. **Knowledge Acquisition**. Vol. 1, No. 2, 1989, pp: 215-233.

[5] Berry, C.D.; Hart, A. E. *Evaluating Expert Systems*. **Expert Systems**. Vol. 7, No. 4, November , 1990, pp: 199-207.

[6] Boehm, B. *A Spiral Model of Software Development and Enhancement*. **Computer**, 21 (5). 1988. pp: 61-72.

[7] Emery, F.E. *Characteristics of social-technical systems,* **The emergence of a New Paradigm of work.** Australian National University, Canberra. 1978.

[8] Geissman, J.R.; Schultz, R.D. *Verification and Validation of Expert Systems*. **AI Expert**. February. 1988. pp: 26-33.

[9] Genesereth, M.R.; Fikes, R.E. **Knowledge Interchange Format**. Version 3.0. Reference Manual. Report Logic-92-1. Computer Science Department. Stanford University. Stanford, CA, 94305. 1992.

[10] Gómez-Pérez, A. *Some Ideas and Examples to Evaluate Ontologies*. Technical Report **KSL-94-65.** Knowledge Systems Laboratory. Stanford University. CA. 1994.

[11] Gruber, T. *A Translation Approach to Portable Ontology Specifications*. **Knowledge Acquisition**. Vol. 5. 1993. pp: 199-220.

[12] Gruber, T. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*. **Technical Report KSL 93-04**. Knowledge Systems Laboratory. Stanford University. CA. 1993.

[13] Gruber, T; Olsen, G. *An Ontology for Engineering Mathematics*. In Jon Doyle, Piero Torasso, & Erik Sandewall, Ed., **Fourth International Conference on Principles of Knowledge Representation and Reasoning**, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann, 1994.

[14] Gruninger, M.; Fox, M.S. *The role of Competency Questions in Enterprise Engineering*. **IFIP WG5.7 Workshop on Benchmarking. Theory and Practice**. Trondheim, Norway, 1994.

[15] Guha, R.V.; Lenat, D. *Enabling Agents to work Together*. **Communications of the ACM.** July 1994. Vol. 37. N0 7. pp: 127-142.

[16] Guida, G.; Mauri, G. *Evaluating Performance and Quality of Knowledge-Based Systems: Foundation and Methodology*. **IEEE Transactions on Knowledge and Data Engineering**. Vol. 5, No. 2, April, 1993. pp: 204-224.

[17] Harmon, P.; King, D. **Expert Systems: Artificial Intelligence in Business and Industry**. New York. NY. John Wiley & Sons. 1990.

[18] Hope, T.; Meseguer, P. *VVT Terminology: A Proposal*. **IEEE Expert**. Vol. 8, No. 3, June. 1993. pp: 48-55

[19] Lenat, D.B., Guha, R.V.; **Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.** Addison-Wesley Publishing Company, Inc. CA. 1990.

[20] Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W.R. *Enabling Technology for Knowledge Sharing*. **AI Magazine**. Winter 1991. pp: 36-56.

[21] Mate, J.L.; Pazos, J. **Ingenieria del Conocimiento**. Cordoba, Argentina. 1988.

[22] Mengshoel, O.J.; Delab, S. *Knowledge Validation: Principles and Practice*. **IEEE Expert**. Vol. 8, No. 3, June. 1993, pp: 62-68.

[23] Newell, A. *The knowledge Level*. **Artificial Intelligence**. Vol. 18. 1982. pp: 87-127.

[24] Nguyen, T.A.; Perkins, W.A..; Laffey, T.J.; Pecora, D. *Checking an Expert Systems knowledge Base for Inconsistency and completeness*. **Proc. of the International Joint Conference of Artificial Intelligence Applications.** AAAI. 1985. pp: 374-378.

[25] O'keefe, R. M.; Balci, O.; Smith E.P. *Validating Expert System Performance*. **IEEE Expert.** Winter. 1987. pp: 81-89.

[26] O'Leary, D.E. *Validation of Expert Systems-With Applications to Auditing and Accounting Expert Systems.* **Decision Science**. Vol. 18. No. 3. 1987. pp: 468-486.

[27] Parsaye, K.; Chignell, M. *Expert Systems for Experts*. In **Measuring Expert Systems Performance.** John Wiley. 1988. pp: 365-374.

[28] Polat, F. Guvenir, H. A. *UVT: A Unification-Based Tool for Knowledge Base Verification*. **IEEE Expert**. Vol. 8, No. 3, June. 1993, pp: 69-75.

[29] Preece, A.D. *Towards a Methodology for Evaluation Expert Systems*. **Expert Systems**. Vol. 7, No. 4, Nov. 1990, pp: 215-223.

[30] Reggia, J.A. *Evaluation of Medical Expert Systems: A case of Study in Performance Assessment*. In **Proc. 9th Annual Symposium on Computer Applications in Medical Care**. (SCAMC 85). 1985. pp: 287-291.

[31] Sharma, R. S.; Conrath, D.W. *Evaluating Expert Systems: the Socio-Technical Dimension of Quality*. **Expert Systems**, Vol. 9, No. 3, August, 1992. pp: 125-137.

[32] Suen, C.Y.; Grogono, P.D.; Shinghal, R.; Coallier, F. *Verifying, Validating, and Measuring the Performance of Expert Systems*. **Expert Systems with Applications**. Vol. 1. 1990. pp: 93-102.

[33] Taki, I. *Expert Model for Knowledge Acquisition in the ICOT*. **Technical Presentation at Computer Science Department**. Carnegie Mellon University, Pittsburgh, PA. 1986.

[34] Waterman, D.A. **A Guide to Expert Systems**. Reading MA: Addison-Wesley. 1986.

[35] Wiellinga, B.J.; Schreiber, A.T.; Breuker, J.A. *KADS: A Modeling Approach to Knowledge Engineering*. **KADS-II/T1.1/PP/UVA/008/1.0.ESPRIT-KADSII**. Amsterdam University. 1991.