Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2009 Proceedings

Americas Conference on Information Systems (AMCIS)

2009

User-Friendly Ontology Creation Methodologies - A Survey

Nikolai Dahlem
University of Oldenburg, dahlem@wi-ol.de

Axel Hahn
University of Oldenburg, hahn@wi-ol.de

User-Friendly Ontology Creation Methodologies - A Survey

Nikolai Dahlem
University of Oldenburg, Germany
dahlem@wi-ol.de

Axel Hahn
University of Oldenburg, Germany
hahn@wi-ol.de

ABSTRACT

The convergence of the semantic web and the social web to the social semantic web leads to new challenges in ontology engineering. As of today ontologies are created by highly specialized ontology engineers. In order to unite "the wisdom of crowds" and ontologies new approaches to ontology creation are necessary to bridge the "ontology gap" and enable novice users to create formalized knowledge. Numerous ontology development methodologies are available; in this paper we will briefly present 16 ontology development methodologies and evaluate them against criteria for their user-friendliness and their suitability for usage by novice users and domain experts. Our eventual goal is the identification of best practices and required research for user-friendly ontology design.

Keywords

Survey, Ontology development methodologies, Domain experts, Usability

INTRODUCTION

The success of the semantic web is highly dependent on the available of ontologies. As of today most ontologies are build by academics or in industrial projects. In order to bridge the “ontology gap” it is necessary that average users are enabled in contributing knowledge to the semantic web. User-friendliness is the major factor of success of Web 2.0 and in order to make the semantic web real a similar degree of user-friendliness in ontology development needs to be reached. Available manual ontology creation methodologies and tools are not usable by novice users and auto generation methodologies (Tho, Hui, Fong and Cao, 2006), which would make more rigorous ontology creation methods unnecessary are not yet mature enough. With this survey we access available ontology development methodologies and aim at pointing out potential enhancements. Our eventual goal is the creation of a user-friendly ontology creation methodology and a supporting tool integrating all phases of ontology development. In this paper we will give criteria that a user-friendly ontology development methodology needs to meet. Then we will briefly present a survey of 16 methodologies and evaluate them against our criteria. Eventually we will draw conclusions and give an outlook of our future research.

CRITERIA

An ontology development methodology should be efficient and easy to learn. The following section gives requirements for such a methodology in the form of criteria the methodology has to meet. These criteria are based on experiences made in the course of the STASIS project (http://www.stasis-project.net). This includes a survey with users from academia and industry. A more detailed description of these criteria has been given in our earlier work (Dahlem, Guo, Hahn and Reinelt, 2009). We believe that user-friendly methodologies need to be tool-supported, therefore the criteria consider features of methodologies as well as of tools and underlying languages.

Adequate terminology (C1): The methodology should use adequate terminology and metaphors the user is familiar with. No prior knowledge in the fields of ontology, knowledge or software engineering should be assumed.

Structure (C2): The steps of the ontology live cycle that are covered must be clearly identified. Additionally the life cycle (or life cycles) supported by the methodology should be explained to the users in order to provide guidance and orientation through the process.

Descriptiveness (C3): Every step of the life cycle should be described in detail. In addition every method and function applied in the methodology should be explained.

Transparency (C4): The purpose of each step, method and function must be explained. In order to motivate the users it must be clear why they are performed and at which goals they aim.

Error avoidance (C5): Novice users commit common errors. A methodology should try to detect patterns that hint at common error and inform the user.

Robustness (C6): The probability for errors committed by novice users is higher than compared to trained experts. Therefore a methodology targeted at them must be more robust to errors. The methodology must offer the possibility to correct errors made in previous steps.

Lookahead (C7): Novice users are not always able to foresee the consequences of changes. To avoid changes that contradict the meaning intended by the user, the methodology should point out the consequences of actions and should provide checks for consistency errors in the resulting conceptualization.

Consistency (C8): Novice users lack expert knowledge and hence can be easily confused by the usage of ambiguous terminology and synonyms. Therefore methodology should use clear, unambiguous and well-defined keywords and concepts.

Hiding formality (C9): Formality tends to increase the user's perception of complexity (Gruber, Westenthaler and Gahleitner, 2006). The goal must be to hide the formality form the users, but retain all information needed to generate a formal representation.

Expressiveness (C10): There is a trade-off between the expressiveness of the underlying ontology language and usability. Novice users are easily confused by the availability of too many options. The methodology should focus on a minimal usable set of features.

Conceptualization flexibility (C11): Deciding whether a concept should be a class or an individual depends on the potential applications of the ontology (Noy and McGuinness, 2001). Novice users often lack the experience to accurately foresee the needed granularity, therefore the methodology should support the transformation from individuals to classes and vice versa with as little overhead as possible.

Ontology assumptions (C12): In our evaluation it turned out, that novice users follow the closed-world and unique name assumptions, because it reflects their experiences with the information systems they are used to. The open-world assumption of ontologies and the possible non-uniqueness of names leads to confusion. An implementation of the methodology should point out the consequences of the open-world assumption and help users to convey their intended meaning.

Tool support (C13): As (Sure 2003) stated: 'The effectiveness and efficiency during the application of methodologies increase significantly through tool support.' Existing tools (Protégé, Altova SemanticWorks) are geared towards knowledge engineers or software engineers (Gruber et al., 2006). Ideally the tools supporting the methodology should be specifically targeted at novice users.

METHODOLOGIES

The following subsections briefly present methodologies and review them with regard to the aforementioned criteria. We considered knowledge engineering methodologies, as well as ontology development and ontology management methodologies representatively.

CommonKADS

CommonKADS (Wielinga, Schreiber and Breuker, 1992) is a methodology for building Knowledge Based Systems (KBS). Ontologies are an integral part of such system. The CommonKADS methodology addresses eliciting knowledge, interpreting the elicited data using a conceptual framework and formalizing the conceptualization

The central activity of a KBS is building the model of expertise. CommonKADS uses a four-layer model to distinguish several generic types of knowledge organized into the following layers:

- Static knowledge is the declarative theory of the domain.
- Infered knowledge are all inferences that can be made in this theory.
- Task knowledge are the rules on how to use this knowledge.
- Strategic knowledge is a higher level of control knowledge.

In this work only the static knowledge (or domain knowledge) is considered. In CommonKADS domain knowledge is captured in the following primitives: concepts, properties, two types of relations (is-a relationships between concepts and relationships between properties and values) and structures (used to represent complex objects).

CommonKADS uses standard knowledge engineering terminology, which is used concisely throughout the methodology. It exhibits a clear structure. The descriptiveness is low, it gives a vocabulary, but provides little support for the modeling process. Formality is hidden by using graphical representations. The expressiveness is on the level of frames. CommonKADS is supported by the “Shelley” workbench.

Cyc

The Cyc methodology (Lenat and Guha, 1990) is based on the experiences made building the Cyc Knowledge Base (KB), which contains nearly "two hundred thousand terms and several dozen hand-entered assertions about/involving each term" (http://cyc.com/cyc/technology/technology/whatiscyc_dir/whatsincyc). The aim of the Cyc project was to codify common sense knowledge not found in textbooks, dictionaries, magazines or encyclopedias, e.g. "You have to be awake to eat".

The phases to build the Cyc KB were:

1. Explicit and implicit common sense knowledge is manually coded in the Cyc representation language CycL and stored in the Cyc KB.
2. The codification of knowledge is aided by tools using the knowledge already available in the Cyc KB.
3. Knowledge is automatically extracted from textual knowledge sources. Human interaction is only needed for difficult text parts.

The first two phases have been successfully accomplished, the last goal has been at least partially attained as the Cyc KB automatically learns by means of a trivia game (cf. http://game.cyc.com/game.html).

Cyc uses standard knowledge engineering terminology in a consistent manner. The methodology is undetailed and proposal for a life cycle is made. Formality is high in early steps and lower in later steps, because of added tool support.

DILIGENT

The DILIGENT methodology (Vrandecic, Pinto, Sure and Tempich, 2005) in contrast to other methodologies focuses on the user-centric evolution of ontologies.

The DILIGENT methodology is not constrained to certain ontology types or languages. It implements a process for ontology evolution reflecting changing user needs and recognized knowledge as a moving target. The stakeholders considered in the process are domain experts, knowledge engineers, ontology engineers and users.

The five main process steps in the DILIGENT methodology are:

1. Build: The first step is building an initial core ontology. This ontology doesn't need to be complete with respect to the domain and can be built by a small team.
2. Local adaption: Users work the core ontology and locally adapt it to their own needs. The core ontology can not be changed directly, instead change requests and local adaptions are collected by a control board composed of representatives of the various stakeholders.
3. Analysis: The sum of change requests and local adaptions are analyzed by the control board in order to find similarities. The outcome of this process is a new version of the core ontology.
4. Revise: The control board should revise the core ontology on a regular basis so that the divergence to the local ontologies does not become too great. The goal is to implement obvious user needs and to gain a higher acceptance of the core ontology.
5. Local update: User can update the local ontologies to use the new version of the core ontology.

DILIGENT can be understood as a meta-methodology, therefore some criteria are not applicable. It uses standard ontology engineering terminology consistently throughout the methodology. It has a clear structure and defined life cycle. The expressiveness is dependent on the underlying ontology methodology and language. It is supported by a tool for structured discussion and versioning of ontologies.

DOGMA

DOGMA is a database-inspired approach for engineering ontologies (Jarrar and Meersman, 2002). Its main characteristic is the so-called "double articulation" of ontologies, which addresses the conflict in ontology engineering between possible reuse and complete specification of rules and constraints. In the DOGMA approach ontologies are decomposed into the ontology base and ontological commitments in order to reach semantical independence between ontologies and applications using them. The ontology base consists of intuitively "plausible" domain facts in the form of context-specific binary facts, which are called lexons. These lexons can be grouped together by context identifiers in order to eliminate ambiguities caused by polysemy. The ontological commitment mediates between the ontology base and applications using it. It represents the choice of and adherence to a set of rules and constraints. This architecture leads to the possibility to easily reuse the ontology base, while each application can choose its own ontological commitment. By the comparison of commitments the interoperability of different applications can be judged.

DOGMA uses its own terminology. The overall structure of the method is unclear and little detail is given. The level of formality is medium, because DOGMA uses textual, but complex rules. The expressiveness is on the level of heavy-weight ontologies. It is supported by the DOGMA Server and Modeler tools.

DynamOnt

The DynamOnt aims at cooperatively creating ontologies. It focuses on supporting domain experts in creation formal knowledge models (Gruber et al., 2006). The knowledge represented in DynamOnt can evolve from glossaries over taxonomies to more structured knowledge models. Upper-level ontologies like DOLCE and SUMO are utilized to guide users in the refinement process by asking questions. DynamOnt uses a semantic wiki called IkeWiki (Schaffert, 2006), thus lowering the barrier for non-technical users by hiding some of the complexity and allowing for evolution from textual knowledge to more structured representations.

DynamOnt consistently uses standard ontology engineering terminology. The structure of the methodology is flexible, there is no fixed life cycle. The description of the individual steps lacks detail. To avoid errors DynamOnt uses upper-level ontologies to guide users and the wiki-approach allows for constant evolution of the ontology and hides formality. The expressiveness is on the level of light-weight ontologies (a subset of OWL). DynamOnt is implemented in IkeWiki.

Grüninger and Fox

The methodology of Grüninger and Fox (Grüninger and Fox, 1995) is used to create new ontologies from scratch or extend existing ones and follows an approach to move from informal descriptions to formalized first-order logic representations. It consists of six steps:

1. Motivating scenarios are given in the form of story problems or examples which are not addressed appropriately by existing ontologies.
2. Informal competency questions can be regarded as requirements in the form of questions. They arise out of the motivating scenarios and are later used to evaluate the ontology.
3. Terminology describing the domain is extracted from the competency questions and then specified in first-order logic. This terminology must be able to express the answers to the competency questions.
4. Formal competency questions are created from the informal ones using the formal terminology defined in the previous step.
5. Axioms define the definition of terms in the ontology and constraints on their usage. They are defined in first-order sentences.
6. Completeness theorems are eventually constructed from the formal competency questions, the ontology axioms and instances. They define the conditions under which the solutions to the competency questions are complete.

The methodology of Grüninger and Fox uses its own terminology. It has a clear structure but makes no recommendations for a life cycle. The overall detail of the methodology is low. The expressiveness is very high (First Order Logic). There is no tool support for the methodology.

HCOME

HCOME is a methodology for the development and evaluation of living ontologies (Kotis and Vouros, 2006). HCOME aims at integrating managing ontologies into the day-to-day-activities of knowledge workers.

Therefore it builds on HCONE, a user-centered ontology management environment focused towards collaboration. In HCONE the knowledge space is structured into three parts, the:

1. Personal Space in which users can create their own ontologies, import ontology versions from the shared space and generalize/specialize them. Additionally users can compare and merge versions, add documentation and reuse generic top ontologies.
2. Shared Space in which users can publish their ontologies and collaboratively work on them using a model for structured discussions. Additionally users can compare others` versions and create new versions by incorporating suggested changes.
3. Agreed Space in which agreed ontology versions are stored for browsing and exploitation in applications.

The HCOME methodology is closely linked to this notion of separated knowledge spaces. The three main phases of the methodology are:

1. Specification of the ontology requirements and identification of collaborators.

2. Conceptualization of the ontology and publication to the shared space.
3. Exploitation of the agreed ontology.

HCOME uses standard terminology. It has a clear structure and life cycle. The methodology is described in a very detailed way. It is supported by the HCONE tool, which allows for evolution of ontologies. The expressiveness is on the level of heavy-weight ontologies.

KACTUS

The KACTUS project (Schreiber, Wielinga and Jansweijer, 1995) investigated the feasibility of knowledge reuse in complex technical systems. It proposed an application-driven ontology development by reusing existing application ontologies. The KACTUS toolkit provides a library of available application ontologies and includes support for browsing, editing visualizing, checking and querying them. The steps to perform in order to build an ontology each time an application is build are:

1. Specification of the application: In this step one specifies the context of the application creates a list of terms and tasks and models the components of the application.
2. Preliminary design: Based on the information gathered in the previous step one searches for top-level ontologies developed for other applications which can be refined and restructured.
3. Ontology refinement and structuring: The ontologies found in the previous step are refined and structured to arrive at the final design.

KACTUS uses its own terminology. The structure of the methodology is unclear and there is no detailed description. The expressiveness is on the level of frames with restrictions. The methodology is supported by the KACTUS toolkit.

KBSI IDEF5

KBSI IDEF5 is a methodology that guides and assists domain experts and knowledge engineers in the construction of both small and large reusable ontologies. The IDEF5 methodology was designed to be usable by people with varying skill levels and backgrounds. IDEF5 regards the ontology creation process as open-ended and thus does not adopt a "cookbook" approach, but recommends a general procedure and guidelines instead. The methodology consists of the following steps:

1. Organizing and Scoping: This activity involves establishing the purpose, viewpoint and context for the ontology development project and assigning roles to the team members.
2. Data Collection: This activity involves acquiring the raw data needed for ontology development.
3. Data Analysis: This activity involves analyzing the data to facilitate ontology extraction.
4. Initial Ontology Development: This activity involves developing a preliminary ontology from the acquired data.
5. Ontology Refinement and Validation: This activity involves refining and validating the ontology to complete the development process.

The methodology uses its own terminology; it has a clear structure and life cycle. The methodology is described in high detail. The expressiveness is on the level of heavy-weight ontologies. There is no tool-support for the methodology. Error avoidance is supported through a library of commonly used relations. Formality is hidden by using graphical representations.

METHONTOLOGY

METHONTOLOGY (Fernández, Gómez-Pérez and Juristo, 1997) enables construction of ontologies from scratch at the knowledge level. It was inspired by the main activities performed in the software development process. The methodology follows a life cycle consisting of the following steps:

1. Specification: In this phase the intended use, use scenarios, end-users, level of formality and scope are defined. The authors recommend the use of the middle-out approach for gathering concepts and building a glossary of terms. Additionally it lists techniques for knowledge acquisition: brainstorming, interviews, text analysis and inspection of other, similar ontologies.
2. Conceptualization: In this phase the gathered knowledge is structured in a conceptual model. Therefore the glossary of terms is grouped as concepts and verbs and concepts classification trees and verb diagrams are build. Later more structured, intermediate representations are created that eventually result in a table of formulas and a table of rules.
3. Formalization: In this phase the conceptual model is transformed into a formal one.
4. Integration: In this phase the reuse of existing ontologies from libraries is evaluated. The result of this phase is an integration document specifying the terms reused from other ontologies.

5. Implementation: In this phase the formal model is implemented in it's final form in a suitable language, such as LOOM, Ontolingua or Prolog.
6. Maintenance: In this phase the ontology is constantly evaluated against and adjusted to users needs.

METHONTOLOGY consistently uses standard ontology engineering terminology. It has a clear structure and well-defined life cycle. The descriptiveness is high, knowledge acquisitions is described in detail. Robustness is considered through evaluation and evolution capabilities. There is no tool-support for the methodology.

ONIONS

ONIONS is a methodology for conceptual analysis and ontological integration (Gangemi, Pisanelli and Steve, 1999). Its aim is to coherently work out domain ontologies for various sources as a basis for an integrated model. ONIONS works with a wide range of sources, e.g. classifications, nomenclatures, coding systems and thesauri. The ONIONS methodology is structured into five steps:

1. Represent all concepts, relations, templates, rules and axioms from sources ontologies in the ONIONS formalism (Loom).
2. Analyze and axiomatize available plain text descriptions.
3. Infer distinctions between concepts reusing relations and concepts already available in the ONIONS library and build an inheritance hierarchy. Additionally polysemy and synonymy are handled by introduction of new and merging of concepts.
4. The library of generic, intermediate and domain ontologies is stratified, i.e. intermediate ontologies are used to abstract between generic and domain ontologies.
5. The source ontologies are mapped to the integrated ontology by "equivalent" or "coarser equivalent" mappings.

ONIONS uses its own terminology and has a clear structure, but specifies no life-cycle. There is no hiding of formality. The expressiveness is on the level of light-weight taxonomies. There is no tool-support.

Ontology Development 101

Ontology development 101 (Noy and McGuinness, 2001) is a methodology targeted beginners in the field of ontologies. It takes the form of a guide and builds simple example ontology in the domain of fine wines. The authors of the guide also work on Protégé ontology development environment and thus it uses Protégé as an example. The methodology consists of the following steps:

1. Determine domain and scope: In this step the domain, as well as the envisioned usage of the ontology are determined. Competency questions that the ontology should be able to answer are formulated and the future users and maintainers of the ontology are considered.
2. Consider reuse: Reusing and extending existing ontologies should be considered, because it eases implementation and enhances interoperability with other applications.
3. Enumerate important terms: In this steps terms relevant for the domain, as well as properties of those terms are collected.
4. Define classes and class hierarchy: In this step terms with an independent existence are chosen as classes. Then the class hierarchy is build according to either the top-down, bottom-up or middle-out approach. The appropriate approach depends on ones view on the domain.
5. Define properties: The remaining terms constitute the properties. The authors differentiate between intrinsic, extrinsic, part-of and relationships properties.
6. Define restrictions: In this step restrictions on the properties are formulated, i.e. restricting the value type, allowed values and cardinality or defining domain and range of a property.
7. Create instances: In the final step the ontology is populated with instances.

The methodology uses standard ontology engineering terminology. It has a clear structure and well-defined life cycle. It gives guidelines to help users avoid common errors. The quality of the ontology is accessed by checks and resulting actions are proposed. Formality is not hidden and the ontology is directly encoded in OWL.

On-To-Knowledge

The On-To-Knowledge methodology (Sure, 2003) aims at introducing and maintaining ontology based Knowledge Management (KM) applications into enterprises. The methodology is structured into five main steps:

1. Feasibility study: Problems and opportunities, the focus of the KM application and the involved people are identified.
2. Kickoff: Requirements for the ontology are captured and a semi-formal description is created.
3. Refinement: The semi-formal description is refined and is formalized into a prototype ontology.
4. Evaluation: The generated ontology is evaluated with regards to technological, user-specific and ontological considerations.
5. Application & Evolution: The ontology is applied in the application and maintenance and evolution are supported.

The OTK methodology uses standard ontology engineering terminology. It has a clear structure and well-defined life cycle. The descriptiveness of the methodology is very detailed. The ontology is directly encoded on the level of first-order logic or OIL. OTK is supported by the OntoEdit tool.

SENSUS

SENSUS (Swartout, Patil, Knight and Russ, 1996) is a top-down approach to semi-automatically derive domain specific ontologies from large-scale ontologies. The SENSUS ontology is natural language based and was build merging using existing electronic resources: the PENMAN Upper Model, ONTOS and WordNet were merged with English, Spanish and Japanese dictionaries. SENSUS follows the notion of a broad coverage ontology from which domain ontologies are derived. The SENSUS methodology consists of five steps:

1. Seed terms for the domain are identified by domain experts.
2. The seed terms are linked to SENSUS by hand.
3. Concepts that are in the path to the root of the SENSUS ontology are included in resulting domain ontology.
4. Relevant domain terms that are not yet covered in the domain ontology are added by hand.
5. For nodes with many paths through them, entire subtrees under them may be included in the domain ontology, because other concepts in the subtree are likely to be relevant in the domain. This step is carried out manually with assistance by domain experts.

The methodology uses it own terminology. It has a clear structure, but doesn't propose a life cycle. The descriptiveness is low. The ontology is directly encoded as an taxonomy. There is no tool-support.

UPON

UPON is a use-case driven, iterative and incremental methodology for creating ontologies (De Nicola, Missikoff and Navigli, 2005). It is derived from the well-known Unified Software Development Process and uses the Unified Modeling Language (UML) for blueprinting ontologies. UML diagrams are used to structure, design and evaluate the ontology. The aim of UPON is producing ontologies usable for both humans and automated systems (semantic web services, intelligent agents, etc.). Following the Unified Process UPON has cycles, phases, iterations and workflows. The result of a cycle is a new version of the ontology and a cycle consists of four phases. Each phase itself is subdivided into iterations. In an iteration five workflows take place:

1. Requirements: The goal of the requirements workflow is to determine the domain of interest and scope of the ontology and to define its purpose. To achieve this a storyboard is written and an application lexicon is created. Additionally competency questions for the evaluation of the ontology are identified. Eventually use-cases are identified and prioritized.
2. Analysis: During analysis the reuse of existing resources is considered and the application scenario is modeled using UML enabling domain experts with knowledge of UML to perform this task without the help of ontology engineers. Then glossary of domain concepts is build.
3. Design: During the design workflow concepts are categorized and then the concepts and their relations are further refined.
4. Implementation: During implementation the ontology is formalized in an ontology language. The previously created UML diagrams are used to structure the ontology into parts.
5. Test: During the test workflow the ontology is evaluated for completeness against the use-cases by semantically annotating the UML diagrams with ontology concepts. Additionally the ontology is evaluated against the competency questions created during the requirements workflow.

UPON uses software and ontology engineering terminology. It has a clear structure, well-defined life cycle and is described in high, detail. Formality is hidden by using UML diagrams. The result is a light-weight ontology encoded in a subset of OWL. There is no direct tool-support, but UML tools can be utilized.

Uschold and King

The methodology proposed by Uschold and King (Uschold and King, 1995) is based on the experience of developing the Enterprise ontology. It consists of five main steps:

1. Identify purpose: In this step the reason for building the ontology and the intended uses are defined.
2. Building the ontology: This step consists of three sub-steps:
    a. Ontology capture is the process of identifying the concepts and relationships of the domain and defining them by precise unambiguous textual definitions. The authors propose a middle-out approach for the gathering of relevant concepts.
    b. Coding: In this step the gathered knowledge is expressed in a formal language.
    c. Integration of existing ontologies is considered and existing ontologies or parts are reused if applicable.
3. Evaluation of the ontology against its specification, competency questions and/or the real world.
4. Documentation of the ontology should include all important assumptions made during the development regarding the concepts in the ontology, as well as the language in which it is expressed.

The methodology by Uschold and King uses standard ontology engineering terminology. It has a clear structure and life cycle. The methodology is described in medium detail. It doesn't hide formality and produces a heavy-weight ontology as an result. There is no tool support.

CONCLUSION

In our survey we reviewed the relevant literature and accessed the methodologies from a novice user's point of view. Figure 1 gives an overview over our results and shows that available ontology development methodologies are not suitable for novice users. We did not aim at fine-grained results, but wanted to give a general overview and point out areas where existing methodologies are lacking, thus a plus signifies that a criterion is fulfilled and a minus that it is not. Empty cells mean that a criterion wasn't applicable.

Methodology	Criterion												
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
CommonKADS	−	+	−	−	−	−	−	+	+	+	−	−	+
Cyc	−	−	−	−	−	−	−	+	+	−	−	−	+
DILIGENT	−	+			−	+	−	+					+
DOGMA	−	−	−	−	−	−	−	+	+	−	−	−	+
DynamOnt	−	−	−	−	+	+	−	+	+	+	−	−	+
Grüninger & Fox	−	+	−	−	−	−	−	+	−	−	−	−	−
HCOME	−	+	+	−	−	+	−	+	−	−	−	−	+
KACTUS	−	−	−	−	−	−	−	+	−	−	−	−	−
KBSI IDEF5	−	+	+	−	+	−	−	+	+	−	−	−	−
METHONTOLOGY	−	+	+	−	−	+	−	+	−	−	−	−	−
ONIONS	−	+	−	−	−	−	−	+	−	+	−	−	−
101	−	+	+	−	+	+	−	+	−	+	−	−	+
OTK	−	+	+	−	−	+	−	+	−	−	−	−	+
SENSUS	−	+	−	−	−	−	−	+	−	+	−	−	−
UPON	−	+	+	−	−	−	−	+	−	−	−	−	−
Uschold & King	−	+	+	−	−	+	−	+	−	−	−	−	−

Figure 1. Survey result

On the positive side we discover that all methodologies are consistent (C8) and most are well structured (C2). About 50% are described in sufficient detail (C3), but don't make their steps transparent for the users (C4). Only three methodologies implement error-avoidance techniques (C5), while half of them provide some robustness against error through evaluation and revision loopbacks (C6). Five methodologies manage to hide some formality (C9) and five (partly different) methodologies provide adequate expressiveness (C10). Half of the methodologies are supported by tools (C13). User-adequate terminology (C1), lookahead features (C7), conceptualization flexibility (C11) and ontology assumptions (C12) are not considered in any methodology.

We conclude that there is an evident need for user-friendly ontology development methodologies and supporting tools. In our future work we aim at developing such a methodology and a supporting tool integrating all phases of ontology development. To validate our results we will then compare conceptualizations created by novice users with our methodology with conceptualizations created by ontology engineers with the reviewed methodologies and judge quality and completeness.

ACKNOWLEDGMENTS

Research work for this paper was partly funded by the EU IST FP6 project STASIS (http://www.stasis-project.net).

REFERENCES

1. Dahlem, N., Guo, J., Hahn, A. and Reinelt M. (2009) Towards an user-friendly ontology design methodology, to be published in *Proceedings of the I-ESA 2009*.
2. De Nicola, A., Missikoff, M. and Navigli, R. (2005) A Proposal for a Unified Process for Ontology Building: UPON, Database and Expert Systems Applications, 655-664.
3. Gangemi, A., Pisanelli, D. and Steve, G. (1999) An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies, *Data & Knowledge Engineering*, 31, 2, 183-220.
4. Gruber, A., Westenthaler, R. and Gahleitner, E. (2006) Supporting domain experts in creating formal knowledge models (ontologies), in: *Proceedings of I-KNOW'06. 6th International Conference on knowledge management*, Graz, Austria, 252-260.
5. Grüninger, M. and Fox, M. (1995) Methodology for the design and evaluation of ontologies, in: *Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada.
6. Fernández, M., Gómez-Pérez, A. and Juristo, N. (1997) METHONTOLOGY: from Ontological Art towards Ontological Engineering, in *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, March, Stanford, USA, 33-40.
7. Jarrar, M. and Meersman, R. (2002) Formal Ontology Engineering in the DOGMA Approach, in *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, Springer, 1238-1254.
8. Kotis, K. and Vouros, G. (2006) Human-centered ontology engineering: The HCOME methodology, *Knowledge and Information Systems*, 10, 1, 109-131.
9. Lenat, D. and Guha, R. (1990) Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project, Addison-Wesley Publishing Company Inc.
10. Noy, N. and McGuinness, D. (2001) Ontology Development 101: A Guide to creating your first Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.
11. Schaffert, S. (2006) IkeWiki: A Semantic Wiki for Collaborative Knowledge Management, in *1st International Workshop on Semantic Technologies in Collaborative Applications, STICA*, 2006.
12. Schreiber, A., Wielinga, B. and Jansweijer, W. (1995) The KACTUS view on the 'O' word, in *Proceedings 7th Dutch National Conference on Artificial Intelligence NAIC'95*, Erasmus University Rotterdam, The Netherlands, 159-168.
13. Sure, Y. (2003) A Tool-supported Methodology for Ontology-based Knowledge Management, York Sure, in The Ontology and Modelling of Real Estate Transactions, Ashgate.
14. Swartout, W., Patil, R., Knight, K. and Russ, T. (1996) Toward Distributed Use of Large-Scale Ontologies, in *Proceedings of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems*.
15. Tho, Q.T., Hui, S.C., Fong, A.C.M., Cao, T. H. (2006) Automatic fuzzy ontology generation for semantic Web, *IEEE Transactions on Knowledge and Data Engineering*, 18, 6, 842-856.
16. Uschold, M. and King, M. (1995) Towards a methodology for building ontologies, in *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*.
17. Vrandecic D., Pinto, H., Sure, Y. and Tempich, C. (2005) The DILIGENT Knowledge Processes, *Journal of Knowledge Management*, 9, 5, 85-96.
18. Wielinga, B., Schreiber, A. and Breuker, J. (1992) KADS: a modelling approach to knowledge engineering, *Knowledge Acquisition*, 4, 1, 5-53.