**AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES**

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

# The Domain Ontology driven Systems: Growth and Challenges

[1]Mehfooza. M, [2]Visalaxi. S, [3]Anitha. G, [4]Haroon Basha. I

*[1, 2,3]Department of Information Technology, Rajalakshmi Engineering College, Chennai, India*
*[4]Department of Management, Quaide Milleth College for Men, Chennai, India*

**Address For Correspondence:**
Mehfooza. M, Department of Information Technology, Rajalakshmi Engineering College, Chennai, India
E-mail: mehfoomsphd@gmail.com

**ARTICLE INFO**

***Article history:***
*Received 28 May 2016*
*Accepted 29 July 2016*
*Published 17 August 2016*

***Keywords:***
*Domain Ontology, Information System, Knowledge sharing system*

**ABSTRACT**

This paper aims at providing a comprehensive review on key approaches followed in the field of domain ontology evolution. The increased use of ontologies in several application fields makes it possible to observe requirements for their smooth integration within Information Systems. The increased usage of domain ontology in Information Systems and Knowledge Sharing Systems raises the importance of its maintenance. Domain ontology change management incorporates areas like domain ontology engineering, evolution versioning, merging, integration, and maintenance. As the experts develop a better understanding of the domain, changes are made to the body of knowledge. These properties are essential to enhance the performance of domain ontology-driven Systems in general and domain ontology-driven Information Extraction Systems in particular. The analysis reveals that different individual components have been developed yet a complete integrated system for automated domain ontology learning is not available. This paper introduces some unfolded challenges in the field of domain ontology evolution and learning, which must be tackled to complete the process automatically. Moreover, the new changes could affect the dependent data, applications, systems, and services. Therefore, this paper also discusses in detail why special attention must be paid to minimize the after effects of domain ontology evolution and learning and proposes some possible solutions to achieve this goal.

## INTRODUCTION

Ontologies, being explicit specifications of conceptualizations (Gruber, 1993), play a major role in many of today's Information Systems (ISs) as knowledge bearing artifacts. The impact ontology have on an IS, referred as in (Guarino. N, 2010) distinguishes between a temporal and a structural dimension. The temporal dimension describes whether an ontology is used at development time or at run-time, whereas the structural dimension describes in which way an ontology can affect the components of an IS (e.g., application programs, information resources, and user interfaces). This paper focus on the temporal dimension, more precisely on the use of ontologies at run time. Using ontology at run time can yield two forms of IS: ontology-aware IS and ontology-driven IS. An ontology-aware IS is a system that is just aware of the ontology and can use it whenever needed. An ontology-driven IS, on the other hand, is a system where the ontology is yet another component that co-operates with other components of the IS at run time, as in (Gruber, 1993). This paper also examines what are the requirements that have to be reconciled in order to enhance their smooth integration with Information Systems (ISs) in general and Information Extraction Systems (IESs) in particular.



**To Cite This Article:** Mehfooza. M, Visalaxi. S, Anitha. G, Haroon Basha. I., The Domain Ontology driven Systems: Growth and Challenges. ***Aust. J. Basic & Appl. Sci.,*** *10(12): 300-306, 2016*

### *2 Ontologies in Information Systems:*

To build an IS, some kind of domain and task knowledge should be provided with the IS. The IS cannot be expected as that, which can predict what an application want and just behaves like that. For example, in an application to compute graph drawings with as little edge crossings, the IS should be provided with the information such as what a graph is? What kind of graphs should be processed? (e.g. planar, non planar), How an edge crossing is defined? etc. All this information will be, in general, implicitly coded in the systems' architecture. This implies, that other people, who may want to build similar applications cannot make use of this implicit knowledge, unless they examine the code of the application, which can be a very tedious task. Ontologies can be used within ISs to make domain knowledge explicit, thus reusable. In addition to that, ontologies can contribute to the portability of an IS, as they could be replaced by other ontologies that represent a totally different domain, enabling the IS to work largely domain-independent. However, developers of ontology-driven systems are also confronted with a large set of obstacles, because the ontology life cycle comprises many phases and systems often have to deal with more than one concurrent phase (Noy. N *et al*., 2000).

### *2.1 Obstacles on the Way:*

Despite the benefits ontologies can offer, it is not yet a common approach amongst IS developers to integrate and use ontologies in their systems. The main reason behind this is, it takes more time for a developer to build an ontology-driven application than a usual application. To reduce the integration and run-time costs of ontologies, the ontology engineering process should be automated to a large extent and ontology management services have to be provided in form of an Ontology Management Module (OMM).

The following sections explore about the requirements needed for OMM w.r.t. different phases of the ontology life-cycle. These requirements are different from the known requirements for ontology management in the context of the Semantic Web, referred as in (Noy. N *et al*., 2000). In a scenario where an ontology is used to capture the domain knowledge needed for an IS, on the focus of portability and scalability, the requirements that the OMM has to reconcile are different. The requirements needed for OMM w.r.t different phases of the ontology life cycle are briefly analyzed here. The main challenge in this analyze is that most of these requirements have to be reconciled at the same time to provide the needed services, whereas in the Semantic Web context often only few of them are demanded (M. Klein *et al*., 2003).

#### *2.1.1 Ontology Generation:*

Ontologies, used as part of ISs in general are not that large. So, it should not be hard to generate ontology for a particular task specification at hand. Yet, it could be hard for someone who is not familiar with the particular ontology representation language or the domain. Further, changes in the task specification would require the adaptation of the ontology, else the generation of a new ontology from scratch (S. Bechhofer *et al*, 2001). Therefore, automated approaches for ontology generation are preferred. No matter how they are built, it is necessary to mark the components of the ontology with semantic knowledge regarding the level of confidence (property: confidence_level), which indicates how sure the ontology developer or an automated learning algorithm is about the existence of the component in the conceptualization.

#### *2.1.2 Ontology Integration:*

Ontologies can be generated using different representation languages, which are based on different knowledge representation paradigms (e.g. description logics, frame logics, etc.). To provide scalable and portable ISs, the OMM should be based on an abstract ontology model that can integrate, if not all, most of the ontological knowledge represented in different languages. This would make the system also more flexible to new-coming standards (T. Gabel *et al*, 2004). Further, depending on the task an IS has to perform, the OMM might also have to provide reasoning support for its abstract ontology model.

#### *2.1.3 Ontology Management:*

An ontology used in conjunction with an IS should not be considered as a static artifact, because the changes in the task specification or the domain have to be reflected on the ontology as well. To automate such necessary adaptations, the OMM should provide data-driven change detection. This can be achieved by supplying the OMM with a file corpus of relevant documents to the domain. Enriching components of the ontology with additional semantic knowledge indicating their estimated behavior over a period of time would further ease this process. In their proposed extended ontology model, referred as in (Tamma, V *et al*., 2003), propose an attribute (property: value change frequency) that indicates whether an ontological component is allowed to change its value over time or not.

Apart from the value change frequency property indicating the components' behavior over time, certain additional components are needed to make an IS adaptive, that is, to make them sensitive to changes in the domain.

• Source-link components: represent links between the ontological structures in the ontology and their respective occurrences in the file corpus. If documents are added to or removed from the file corpus, these links can be used to detect which components in the ontology are affected by the change.

• Change components: represent actual changes in the ontology. Every addition, deletion or edition can be represented in form of additional change instances, with appropriate properties about the kind of change, the date of change, etc. These change components also allow to keep track of the evolution of the ontology over time.

### *2.2 Ontology-Driven Information Extraction Systems:*

After examining the requirements for an Ontology Management Module (OMM) within ISs, consider the more specific case, where the IS is an Information Extraction System (IES). The general architecture of such a system is depicted in Figure. Information Retrival (IR) is defined as a form of natural language processing in which certain types of information must be recognized and extracted from text referred in (Riloff, E, 2002). It is an important and popular research field of the current time; for it tries to extract relevant information from the overwhelming amount of data available today. The question "what actually 'relevant information is? that crashes the individual mind immediately cannot be answered so easily. This difficulty in answering such a question is due to its dependency on the current task and domain. And it is even harder to communicate the answer to a computer. Ontologies can be used in that context to provide a specification of relevant information by representing parts of the domain.

An IES utilizes extraction patterns (rules), with which it can decide whether a part of a document is relevant or not. There are two approaches in rule generation: the knowledge engineering approach and the automatically training approach. In the knowledge engineering approach, a knowledge engineer generates the rules for the information extraction process by hand, using his domain and task knowledge. In the automatically training approach, where the aim is to decrease human intervention, a set of documents needs to be annotated manually, whereas the annotations represent relevant parts and can be utilized by the system to learn patterns in order to extract relevant information and also from unseen documents as in (D. Rogozan *et al*., 2005). Ontology can be used in conjunction with both approaches as an artifact, representing a shared conceptualization of the domain to which the knowledge engineer can commit to while generating extraction rules and the annotator can commit to while annotating the file corpus.

Because our focus is on facilitating portable and scalable IES, we will concentrate on the case where the rule generation process is fully automated, that is where the rules are generated automatically using a given ontology. In the following we will take a look at the additional requirements to the ontology w.r.t. to the IE task.

### *2.3 Requirements w.r.t. Information Extraction Systems:*

During the development of an ontology-driven IES, we encountered several requirements for the smooth integration of ontologies within such systems. These requirements should be considered as additions to the one pointed out for the use of ontologies in ISs in general (ref. Section 2.1). The components in the input ontology should contain a few additional properties, which are essential for the Rule Generation Module (RGM) to produce accurate rules (P. Plessers *et al*., 2015).

• Quality Properties: As mentioned before it is important for ISs to have knowledge about the confidence level of the components in the ontology (property: confidence_level). In the case of IESs this is absolutely mandatory, because those levels are needed in order to compute the confidence levels of the rules themselves (Stojanovic *et al*., 2014). These computed levels are used to choose among rules, when more than one rule can be applied to a certain part of text.

• Value Constraint Properties: Value constraints are used to restrict property values such as the data type or cardinality. It is already possible to state this kind of knowledge in ontology representation languages such as OWL (Grigoris, A *et al*., 2004). If there are value constraints on the components of a conceptualization, they should be implemented in the ontology, because the more constraints known the more fine-granulated rules can be generated, which in turn enhances the performance of an IES.

• Temporal Properties: In many settings the components of an ontology have to be marked with temporal values such as the transaction time (property: transaction_time), or valid time (property: valid_time_begin and valid_time_end) of the component. These properties are especially useful in connection with changing ontologies where out-of-date components are not deleted from the ontology but marked as such. In a common scenario where the IES wants to extract information from new and relatively old data alike, a completely up-to-date ontology would not serve the purpose (H. Liu *et al*., 2006).

### *2.4 How to Implement Additional Semantic Knowledge:*

To include additional semantic knowledge into ontology there are two ways: first, to extend an existing ontology representation language with the needed modeling primitives; second, to use already defined modeling primitives to add the semantic knowledge in form of additional properties to components. One always has to

think thoroughly before deciding to take the first way, because any addition to a language not only increases its expressive power also its complexity (G. A. Miller *et al*., 1990). The increase in complexity cannot be evaluated in prior, so it is not clear whether the benefits would justify the additional costs caused. Furthermore, often such additional knowledge is needed only by a particular group and not by the whole community, hindering the wide acceptance of the extension.

As in (Kushmerick, N *et al*., 2005), agree to some extent with the objection that this kind of additional knowledge does not represent ontological knowledge and therefore its presence in an ontology is questionable. But they also state that ontologies must contain additional information in particular settings, especially when complex and accurate services are demanded (e.g., multi-agent systems). This proposal also go in hand with (A. M. Khattak *et al*., 2010), and analyze that in the context of IESs it is much easier to build an ontology and to implement some additional knowledge in it, than to build and refine extracting rules by hand in order to increase the performance of the system.

### *3 Survey on related work:*

One can observe that in many cases additional knowledge about components in the ontology is needed to perform the task in hand more accurately. Often researchers use an abstract ontology model to integrate knowledge in existing ontologies and to enrich them with their proposed additional knowledge.

For the case of ontology learning from text documents, (G. A. Miller *et al*., 2001) argued in a similar way and proposed their Probabilistic Ontology Model (POM). In this model they save the results of their learning system by attaching a probability (confidence level) to them. Doing this, they aim to enhance the interaction with the user by presenting the learned structures ranked according to their confidence level or by presenting the only results above a certain confidence threshold. Furthermore, their POM also contains links of the structures to corresponding documents from which they were derived; allowing the user to understand the context of a particular structure and allowing the system do react to changes in the document corpus (Kushmerick, N *et al*., 2005). We think that both of these additions to the components of an ontology are essential for the use of ontologies in ISs.

**Table 1:** Comparisons between various systems

| SYSTEM | CONTRIBUTION | LIMITATION | EVOLUTION |
|---|---|---|---|
| Protégé | • Mostly used for ontology creation.<br>• Often used for evolution and maintenance.<br>• Provides Merging, Integration and Comparison.<br>• SparQL queries support. | • Weak ontology change management.<br>• No facility for ontology recovery.<br>• Use third party services for consistency checking | • Manual evolution support. |
| KAON | • Provides ontology editing services like protégé.<br>• Provides environment for pre-evolution strategy making, avoid conflicts using deduce changes.<br>• Supports automatic evolution, redo and undo.<br>• Provides collaborative editing facility. | • Complex systems<br>• Slow in response.<br>• Needs ontology engineering for conflict resolution. | • Pre-defined strategy based evolution support. |
| OilED | • Used for ontology engineering.<br>• Disallows inconsistency in ontology.<br>• Support semi-automated<br>• ontology evolution. | • No change logging facility.<br>• No facility for ontology recovery.<br>• Strict in its operation. | • Semi automatic support. |
| OntoEdit | • Used for ontology editing.<br>• More options than KAON for strategy making.<br>• Allows collaborative editing environment. | • Provides less operation than Kaon.<br>• To avoid side effects of conflict,it involves ontology engineer. | • Strategy based evolution support. |

As in reference (A.M. Khattak *et al*., 2010), motivated an extended ontology model to characterize precisely the concepts ‘properties and expected ambiguities, including which properties are prototypical of a concept and which are exceptional, the expected behavior of properties over time and the degree of applicability of properties to sub concepts. The authors claim that this enriched semantics can prove useful to describe what agents know in a multiagent system. As our focus is on IESs, we are not concentrating on all of the proposed meta properties. The properties describing the components’ expected behaviour over time alone are used, these properties can help during the ontology management phase when the ontology has to be adapted to changes in the domain. At present different tools were used for ontology editing and they are classified based on their

contribution, evolution and limitation of each systems. Table. 1 gives the comparison between various systems. Various evolution approaches were introduced for ontology based system. They were categorized based on change request, change representation, conflict resolution, change implementation, propagation and working. Table 2. Summary of ontology evolution approaches. Last column represent maturity level of the approach in terms of automation.

**Table 2:** Comparison on ontology evolution approaches

| Approaches | Change request | Change Representation | Conflict resolution | Change Implementation | Change Propagation | Working |
|---|---|---|---|---|---|---|
| L. Stojanovic, A. Madche, B. Motik, and N.Stojanovic | The complete change request is represented in formal representational format. These changes (due to business requirement)are specified by ontology engineer. | | Ontology engineer resolves all the inconsistencies due to requested changes by incorporating deduced challenges. | The requested changes (including deduced changes) are applied to source ontology. | Applied changes are propogated to dependant data, applications and services. Out dated instances are replaced. | User interventions required for system working. |
| M.Klein, NNoy et,al. | Specified by ontology engineer. | Developed change and Annotation ontology (CHAO) to represent change request. | Ontology engineer involvement. | Suggested that tools should provide interface for user interaction | Consistent propogation of changes to distributed instances of ontology. | User interventions required for system working. |
| T. Gabel, Y. Sure, and J. Voelker | Specified by ontology engineer. | Formal representation of changes. | Predefined strategies for conflict resolution. | Provide interface for user interaction and also logs the changes. | Propogation of changes to depend artifacts. | Most part of the system are working. |
| P.Plessers and O. de Troyer | Different versions of ontologies are used in this approach. Changes among different versions are represented formally. | | After change implementation, it checks for inconsistencies and implement change recovery. | First it implement change request and then checks for any conflicts. | It does not support change propogation as it works on versions. | User interventions required for system working. |
| D.oberle.et.al | Changes detected among two versions by using prompt Diff and Onto view and a complete change request is compiled. | Formally represented using their developed semantic structure. | Ontology engineer resolves inconsistencies by introducing deduced changes. | With change implementation, all the changes are also logged for undo/redo purpose. | It does not support change propogation as it works on versions | User interventions required for system working. |
| P.Plessers and O. de Troyer | Use top down manual and bottom up automatic approach for change detection. | Suggestions for the use of formal representation i.e using the change log representation. | Involve ontology for engineering for resolving conflicts. | Manual implementation of these changes. | It does not support change propogation | User interventions required for system working. |
| H. Liu, C. Lutz, M. Milicic, and F. Wolter | Changes are suggested by end user and are assumed to be represented at atomic level. | | For all conflicts ,the resulting | Changes are implemented ;no log is | Suggestions for consistent propogations are | User interventions required |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | solution are calculated using DL assertions. | maintained for this. | made. | for system working. |
| S. Castano, A. Ferrara, and S. Montanelli | Changes are recogonized automatically by analyzing domain artifacts.H.Match and wordNetare used for change detection. | Changes are then formally represented. | Inconsistencies are resolved by ontology engineer. | Changes are made by ontology engineer. | Changes propogation is not a focus. | Semi automatic. |
| A. M. Khattak, K. Latif, S. Y. Lee, Y. K. Lee, and T. Rashee | New changes such as (change in single concept ,group of concepts and concepts in a hierarchical structure) are detected automatically using H-Match[27]and wordnet.Change representation is provided by change history Ontology(CHO)[21]. | | For conflict resolution KAON API is used with some suggested extensions. | Changes are implemented automatically and after every change implementation these are logged in CHL. At the end all changes are validated against the change request. | Change propogation is not handled in this approach. | This approach provides suggestion toward automation on the process. |
| 49. F. Zablith | Changes can be specified by user and detected automatically. They also use Word Net for new change detection. Then these changes are formally represented using different representation techniques followed in their overall Neon Toolkit. | | A new developed algorithm for conflict resolution strategy is partially implemented. | Changes are implemented and verified. | Change propogation is the focus for the 2nd phase with conflict resolution. | This approach is a step towards automatic evolution. |

### *4 Conclusions and Future Work:*

Originally, ontologies have been proposed to be used as the backbone of the Semantic Web. Consequently, most of the research that has been done in the field of ontology engineering has found Semantic Web as their application field. But often, Information Systems (ISs) do not share the characteristics of the Semantic Web. So we can say that other requirements are demanded from ontologies when used in different settings. These requirements have to be analyzed before making a decision about using ontology in an IS. Hence, the main contribution of this paper is, an in-depth analysis of the changed requirements regarding Ontology Engineering (OE) when used in ISs in general and Information Extraction Systems (IES) in particular. This paper also explained in detail about the obstacles on the way to integrate ontologies in ISs and IESs. Therefore, every developer has to analyze whether the benefits justify the additional costs, which can arise because of the ontology usage. The requirements mentioned in this paper have to be reconciled in order to faster the wide acceptance of ontology-driven IS development.

## REFERENCES

Khattak, A.M., K. Latif, S.Y. Lee, Y.K. Lee and T. Rasheed, 2009. “Building an integrated framework for ontology evolution management,” in Proceedings of the 12th Conference on Creating Global Economies through Innovation and Knowledge Management, pp: 55-60.

Berners-Lee, T., 1999. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. Harper San Francisco.

Cimiano andVölker (Cimiano & Völker, 2005) Ontology Learning and Population from Text: Algorithms, Evaluation Page no 238.

Cimiano, P. and J. Völker, 2005. Text2Onto – a Framework for Ontology Learning and Data-driven Change Discovery. In Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB’2005), 227-238.

Oberle, D., R. Volz, B. Motik and S. Staab, 2004. “An extensible ontology software environment,” in Handbook on Ontologies (Series of International Handbooks on Information Systems), Springer, pp: 311-333.

Rogozan, D. and G. Paquette, 2005. “Managing ontology changes on the semantic web,” in Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence, pp: 430-433.

Zablith, F., 2009. "Ontology evolution: A practical approach," poster, in Proceedings of Workshop on Matching and Meaning at Artificial Intelligence and Simulation of Behavior.

Miller, G.A., 1990. "WordNet: An on-line lexical database," International Journal of Lexicography, 3: 235-312.

Grigoris, A. and F. van Harmelen, 2004. A Semantic Web Primer. Cambridge: The MIT Press.

Guarino, N., 2010. Formal Ontology and Information Systems. In Proceedings of the First International Conference on Formal Ontologies in Information Systems (FOIS), 3-15.

Liu, H., C. Lutz, M. Milicic and F. Wolter, 2006. "Updating description logic ABoxes" in Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning, pp: 46-56.

Kushmerick, N. and B. Thomas, 2002. Adaptive Information Extraction: Core Technologies for Information Agents. In Intelligent Information Agents: The AgentLink Perspective. Berlin/Heidelberg: Springer, pp: 79-103.

Stojanovic, L., A. Madche, B. Motik and N. Stojanovic, 2014. "User driven ontology evolution management," in Proceedings of European Conference on Knowledge Engineering and Management, pp: 285-300.

Klein, M. and N.F. Noy, 2003. "A component-based framework for ontology evolution," in Proceedings of the Workshop on Ontologies and Distributed Systems, CEUR-WS, 71.

Klein, M., 2004. "Change management for distributed ontologies," Ph.D. Thesis, Department of Computer Science, Vrije University, Amsterdam.

Noy, N.F., A. Chugh, W. Liu and M.A. Musen, 2006. "A framework for ontology evolution in collaborative environments," in Proceedings of International Semantic Web Conference, pp: 544-558.

Noy, N., R. Fergerson and M. Musen, 2000. "The knowledge model of Protégé Combining interoperability and flexibility" in Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management: Methods, Models, and Tools, pp: 17-32.

Plessers, P. and O. de Troyer, 2015. "Ontology change detection using a versioning log," in Proceedings of the 4th International Semantic Web Conference, pp: 578-592.

Riloff, E., 2002. Information Extraction as a Stepping Stone Toward Story Understanding. Understanding Language Understanding: Computational models of Reading, pp: 435-460.

Bechhofer, S., I. Horrocks, C. Goble and R. Stevens, 2001. "OilEd: A reasonable ontology editor for the semantic web," in Proceedings of the 24th German/9th Austrian Conference on Artificial Intelligence, pp: 396-408.

Castano, S., A. Ferrara and S. Montanelli, 2006. "Matching ontologies in open networked systems," Techniques and Applications, Journal on Data Semantics, 7: 25-63.

Gabel, T., Y. Sure and J. Voelker, 2004. "KAON – ontology management infrastructure," D3.1.1.a, SEKT Project: Semantically Enabled Knowledge Technologies.

Tamma, V. and T. Bench-Capon, 2002. An Ontology Model to Facilitate Knowledge-Sharing in Multi-agent Systems. The Knowledge Engineering Review, 17(1): 41-60.

Sure, Y., J. Angele and S. Staab, 2003. "OntoEdit: Multifaceted inferencing for ontology engineering" Journal on Data Semantics, 1: 128-152.