2018

Ontology in Software Engineering

Salvatore F. Pileggi
School of Information, Systems and Modelling, University of Technology Sydney,
SalvatoreFlavio.Pileggi@uts.edu.au

Antonio A Lopez-Lorca
School of Computing and Information Systems, University of Melbourne, antonio.lopez@unimelb.edu.au

Ghassan Beydoun
School of Information, Systems and Modelling, University of Technology Sydney,
Ghassan.Beydoun@uts.edu.au

Ontology in Software Engineering

Salvatore F. Pileggi
School of Information, Systems and Modelling
University of Technology Sydney
Australia
Email: SalvatoreFlavio.Pileggi@uts.edu.au

Antonio A Lopez-Lorca
School of Computing and Information Systems
University of Melbourne
Australia
Email: antonio.lopez@unimelb.edu.au

Ghassan Beydoun
School of Information, Systems and Modelling
University of Technology Sydney
Australia
Email: Ghassan.Beydoun@uts.edu.au

Abstract

During the past years, ontological thinking and design have become more and more popular in the field of Artificial Intelligence (AI). More recently, Software Engineering (SE) has evolved towards more conceptual approaches based on the extensive adoption of models and meta-models.

This paper briefly discusses the role of ontologies in SE according to a perspective that closely matches the theoretical life-cycle. These roles vary considerably across the development lifecycle. The use of ontologies to improve SE development activities is still relatively new (2000 onward), but it is definitely no more a novelty. Indeed, the role of such structures is well consolidated in certain SE aspects, such as requirement engineering. On the other hand, despite their well-known potential as knowledge representation mechanisms, ontologies are not completely exploited in the area of SE.

We first (i) proposes a brief overview of ontologies and their current understanding within the Semantic Web with a focus on the benefits provided; then, the role that ontologies play in the more specific context of SE is addressed (ii); finally, we deal with (iii) some brief considerations looking at specific types of software architecture, such as Multi-Agent Systems (MAS) and Service-Oriented Architecture (SOA).

The main limitation of our research is that we are focusing on traditional developments, where phases occur mostly sequentially. However, industry has fully embraced agile developments. It is unclear that agile practitioners are willing to adopt ontologies as a tool, unless we ensure that they can provide a clear benefit and they be used in a lean way, without introducing significant overhead to the agile development process.

Keywords Ontology, Software Engineering.

1 Introduction

During the past years, ontological thinking and design have become more and more popular in the field of Artificial Intelligence (AI). More recently, Software Engineering (SE) has evolved towards more conceptual approaches based on the extensive adoption of models and meta-models (Henderson-Sellers 2011). A recent survey on applications of ontologies for SE (Bathia et al. 2016), provides a broad analysis of current issues and challenges for the years to come. The authors define a fine-grained classification of the ontologies based on their type and scope. The main categories considered within the survey are upper ontology, software process ontology, domain ontology, requirement ontology, architecture and design ontology, pattern ontology, implementation ontology, documentation ontology, quality ontology, maintenance ontology and technology ontology.

In this paper we reflect on the adoption of ontologies in the field of SE according to a role-based perspective that maps the SE life-cycle much closely than in Bathia et al. (2016). We focus our discussion on Requirements Engineering (RE). Furthermore, we briefly discuss the use of ontologies in the context of Multi-Agent Systems (Van der Hoek and Wooldridge 2008) and Service-Oriented Architecture (Stantchev and Malek 2010) development. As far as the authors know, there are no surveys or papers on the topic adopting a similar approach. However, we have identified a number of works that focus vertically on specific aspects or sub-problems. For instance, Gigante et al. (2015) deal with the role of semantics in requirements verification, while Ding et al. (2014) review knowledge-based approaches in software documentation. These contributions are valuable and informative, however, they lack the big picture view that holistic approaches can provide.

This paper first proposes a brief overview of ontologies and their current understanding within the Semantic Web (Berners-Lee et al. 2001) with a focus on the benefits provided. Then, the role that ontologies play in the more specific context of SE is addressed. We focus on the RE activities while maintaining a holistic view of the SE life-cycle. Finally, the paper briefly considers the cases of Multi-Agent Systems (MAS) and Service-Oriented Architecture (SOA).

2 Ontology and Web Semantics

The term "ontology" was originally coined in ancient Greece in metaphysics, a major branch of philosophy at the time, to mean "the study of what is there". This included for example, the question of whether or not there is a God (Hofweber 2018). AI community adopted the term in the 1980's to describe a central component of knowledge-based systems (Akerkar and Sajja 2010), to conceptualise and to theorise a modelled world (Gruber 2009). In the early 1990's Gruber (1993) defined ontology as an "explicit specification of a conceptualisation", a formalization of the definition suitable to many systems.

More recently, ontology has become one of the key concepts underpinning the Semantic Web model (Chandrasekaran et al. 1999) where, unlike other data models, ontology is supposed to address the "meaning" of the target data, information or knowledge. Indeed, in a Web context, ontology is understood as a rich data model, developed upon the Web infrastructure, able to represent complex resources and the relations among them. As resources are univocally identified worldwide by adopting IRIs (Web resources), ontologies work according to an enhanced model of interoperability, commonly referred to as Semantic Interoperability (Obrst 2003). Furthermore, ontology overcomes the most basic understanding of the Semantic Web (e.g. Linked Data (Bizer et al. 2011)) by providing capabilities for standard reasoning, normally based on the specification of inference rules.

Ontologies may have different scopes and purposes. They are currently adopted in a wide range of domains and discipline. In the context of this work, we mostly refer to domain ontologies which, specify a given-domain by defining its objects, the relationships among them and axioms that govern the domain. The main goal of a domain ontologies is to represent a shared view of a domain. Generally speaking, the model of a domain assumes all parties involved in their use agree on the represented conceptualisation, meaning that the concepts adopted are expected to be well-known and accepted.

Ontologies can be defined as Web ontologies using the OWL language (McGuinness and Van Harmelen 2004). This formalisation relies on Axioms which define facts involving Web resources and literals that are supposed to be always true in the considered context. Web ontologies operate under the assumption of an open world, the falsehood of unknown facts is not assumed. Key elements of OWL ontologies are:

- *Classes and class hierarchy* which determine the type of concepts defined in the ontology, i.e. that exist in the domain. For example, a generic class "person" belonging to a given domain can assume two sub-classes, e.g. "worker" and "student", defining a hierarchy.
- *Individuals* are normally instances of classes, although it is possible to define un-typed individuals.

- *Properties* establish the relationships among concepts in a given domain. For example, an employee could be related to his employer by a given property. Properties are also adopted to specify the attributes of an individual, i.e. the age or the address of an employee.
- *Inference constructs and rules* are extensively used within ontologies to infer further knowledge (referred to as inferred knowledge) from available facts. Inference is a powerful and key mechanism to define semantic patterns in knowledge.

2.1 Benefits of Ontologies

The benefits of ontologies are implicitly defined by their own purpose: by providing the "meaning" of the information addressed, ontologies are knowledge structures richer and more expressive than other data models. Additionally, despite their intrinsic complexity, ontologies can be "understood" (at least in theory) by both machines and humans that can so have exactly the same view of a given system. That is in contrast with most models that normally target humans or computers but very rarely both them. Furthermore, ontologies are descriptive models by definition as they focus on the analysis and the description of the domain. So they support very well descriptive approaches, unlike other models that are naturally oriented to prescriptive solutions (Henderson-Sellers 2011).

Potential benefits of ontologies have been extensively discussed in literature (e.g. by Bürger and Simperl (2008)). First of all, because of the high expressiveness, adopting ontologies provides an opportunity for automatic reasoning to infer implicit knowledge. On the other hand, other data models lack the necessary underlying semantics. Furthermore, the standardisation and the development upon the Web infrastructure significantly increase the capabilities in terms of information exchange, sharing and re-use. In very few words, the potential benefits provided by ontological frameworks overcome the simple, yet relevant, facilitations for data aggregation from heterogeneous data-sources. For instance, in SE inconsistencies in the modelled knowledge can be detected to prevent propagating errors to later phases of software development. Focusing on a domain perspective, they do not only provide the structure for a data container, but define a formalised domain theory both with its implementation. In more general terms, the benefits of using ontologies have been concisely discussed by Gruninger and Lee (2002), who have identified three main areas:

- *Communication*. Ontologies can assist to ensure interoperability among software entities at multiple layers (e.g. data and process level). The specification of formal semantics contributes to avoid ambiguous definitions as well as unwanted interpretations. Moreover, by providing semantically consistent links, it facilitates knowledge engineering, consolidation and transfer.
- *Reuse*. Ontologies can be used to develop systematic widely accepted knowledge environments. Once an ontology is available in a certain domain, it can be reused for similar developments, as well as it can be used in a different or wider scope. This avoids the expensive ad-hoc development and may increase the quality of the final product. Linking concepts from different ontology provides enormous, still largely unexplored, capabilities. Finally, ontologies are intrinsically extendible.
- *Inference and automatic reasoning*. Ontologies typically adopt logic-based languages (e.g. Description Logics) to define inference rules. These structures permit deriving implicit facts from the explicit stated knowledge. Inference rules are solved by specialized software components, known as reasoners. Inference is a key and critical feature: increasing the capability in terms of inference results in a more sophisticated technology.

3 Role of Ontologies in Software Development: Requirement Engineering

Most software development processes share a core set of phases (Tsui 2009) (Figure 1, left):

- The *analysis phase* is aimed at extracting the system requirements from the customer and at building the models to understand the problem.
- In the *design phase*, designers think in terms of the solution to define the architecture of the software without concerning themselves with low-level operational detail.
- Within the *codification phase*, programmers materialize the architecture defined in the previous step by using a concrete programming language.

- During the *testing phase*, the different artefacts produced during the software development are verified and validated to assure their quality and compliance with the original requirements.
- Finally, once the product has been delivered to the client, it starts the last phase, *maintenance*, where errors are identified and fixed. Furthermore, new requirements may be eventually added to the system.

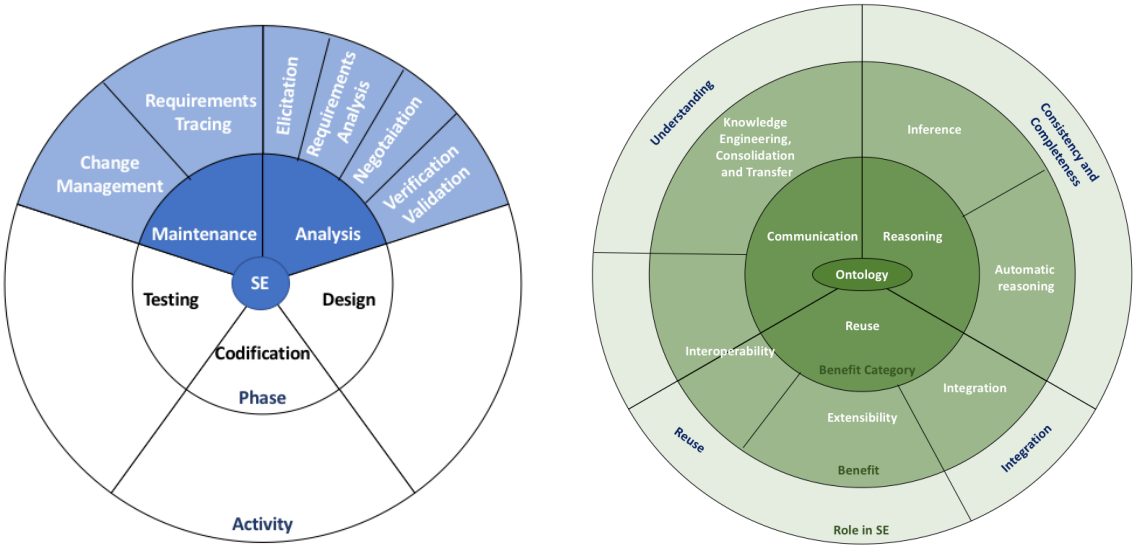

Figure 1: Paper scope in the context of SE (left) and identified ontology roles (right).

The analysis phase is commonly known as requirement analysis phase and, because of its importance, is often object of special attention. It can be further divided in sub-tasks, each one characterised by the following activities (Sadraei et al. 2007):

- *Elicitation* identifies high level goals of the target system, requirements for different groups of users, and the tasks to be accomplished, along with system boundaries. The normal outcome is the early requirements document.
- *Requirement Analysis* analyses the requirements to uncover conflicts, ambiguities, missing or duplicate requirements in order to identify alternatives and convert them into a structured and unambiguous representation. The analysis outcome is an early requirements model.
- *Negotiation* addresses key trade-offs to achieve agreement between stakeholders.
- *Verification and Validation* examine requirements to find any deficiencies in consistency, accuracy and adequacy. It may also include a feasibility analysis to verify the cost of development. After verification and validation, the requirements model should be understood by systems analysts.

Concerning the maintenance phase, two commonly accepted sub-phases may be identified:

- *Change Management* recognizes changes through continuous requirements elicitation, re-evaluation of risk and evaluation of the system in its operational environment (Nuseibeh and Easterbrook 2000), to assure that all relevant information for each change is collected.
- *Requirement Tracing* manages the evolution of requirements, maintaining traces about its history to track the origins of each requirement, so that if a change has to be made to a design component, the original requirement can be located (Davis 1993).

Looking at the typical SE life-cycle with a focus on analysis and maintenance (Figure 1, left), we have identified four different major roles (Figure 1, right) as follows:

- *Consistency and completeness checking*. Ontologies define formal semantics based on some logic. It contributes and can facilitate the automation of important modelling tasks, such as consistency and completeness checking. Consistency checks avoid inconsistent use of knowledge, i.e. to assert truth and falsehood of a given fact at the same time. Automated consistency checking enables the

detection of such assertions through a chain of systematic logical steps. During consistency checking implicit knowledge can also be explicated and modelled. Ontologies can be used this way to complete (Kaiya and Saeki 2005) models acting as reference frames to identify modelling gaps or lacks.

- *Understanding*. By formally defining relations in a domain, ontologies can facilitate the understanding of the target system (e.g. Graja et al. 2011). An ontological approach allows to deconstruct the problem, splitting it into fragments that are easier to understand and analyze. Furthermore, these fragments can be classified, which helps comprehending the underlying structure and eventually defining further relations. Annotation can then be used to augment the concepts of an ontology with metadata that describe them and give further information. These annotations combined with formal relations and structural knowledge, make possible to address a subsequent complexity, suitable for humans and machines both. Semantic query (Ray 2009) provides a powerful interface that enables flexible interaction with ontologies.
- *Integration*. The features of semantic technologies intrinsically enable further capabilities in terms of integration (e.g. Paulheim 2009) within heterogeneous systems, i.e. facilitating that different systems to interoperate and share knowledge. Ontologies are technology-independent knowledge modelling artefacts and can be used in heterogeneous contexts in a wide range of technologies.
- *Reuse*. As previously discussed, ontologies holistically define knowledge by relating atomic concepts. In the context of SE, ontologies are usual to separate domain knowledge from operational knowledge. This facilitates the reuse (Uschold et al. 1998) at two levels: first of all, the ontology schema can be reused as a shared knowledge conceptualization; moreover, it can facilitate the reuse of the concrete artefacts that they describe.

3.1 Ontology-driven MAS and SOA Development

The effectiveness of ontology-driven development techniques becomes evident in the moment in which more specific paradigms are considered. Figure 2 shows the theoretical life-cycle for MAS (Van der Hoek and Wooldridge 2008) and SOA (Stantchev and Malek 2010). An exhaustive analysis is out of the scope of the paper. However, it is possible to identify a number of contributions in literature, that adopt an ontology-driven approach for the development of a specific kind of software architectures.

There are cases in which ontologies are applied to generic aspects. That is, for example, the case of the methodology adopted by Cossentino et al. (2010) which adopts an ontological approach in early stages of analysis to describe the problem domain concepts and to complement the requirements description in terms of use cases in MAS. As well as, Tran and Low (2008) focus on ontological capabilities to produce MAS whose components are interoperable and reusable.

Some other works attempt to use ontologies to improve aspects, specific of a kind of architecture. For instance, Girardi and Leite (2008) address explicitly the model of the MAS domain. Nyulas et al. (2008) present an ontology-driven framework for deploying MAS upon JADE (Bellifemine et al. 2007). Sensoy et al. (2010) and Fornara and Colombetti (2009) use OWL ontologies to define the policies that rule agent behaviour.

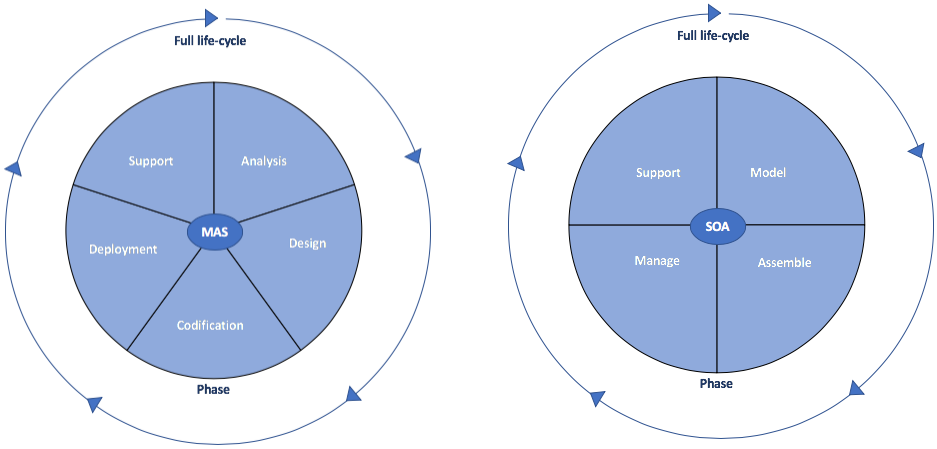

Figure 2: Theoretical MAS (left) and SOA (right) lifecycle.

4 Conclusion and Future Work

This paper briefly discusses the role of ontologies in SE according to a perspective that closely matches the theoretical life-cycle. These roles vary considerably across the development lifecycle. The use of ontologies to improve SE development activities is still relatively new (2000 onward) but it is definitely no more a novelty. Indeed, the role of such structures is well consolidated in certain SE aspects, such as requirement engineering. On the other hand, despite their well-known potential as knowledge representation mechanisms, ontologies are not completely exploited in the area of SE.

We believe that ontology-driven software development will be consolidated in the next future, becoming more popular and, eventually, extensively adopted in fact. To contribute to this goal, we seek to explore uses of ontologies for the development of both MAS and SOA. We will attempt to harness their potential, including automatic reasoning, as a tool to enable the verification and validation of these systems. This is, ensuring consistency and traceability of artefacts along the development life-cycle, and consistency against the client needs, respectively.

One limitation of our research, is that we are focusing on traditional developments, where phases occur mostly sequentially. However, industry has fully embraced agile developments. It is unclear that agile practitioners are willing to adopt ontologies as a tool, unless we ensure that they can provide a clear benefit and they be used in a lean way, without introducing significant overhead to the agile development process.

5 References

Akerkar, R., and Sajja, P. (2010). *Knowledge-based systems*. Jones & Bartlett Publishers.

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing multi-agent systems with JADE* (Vol. 7). John Wiley & Sons.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). "The Semantic Web". *Scientific American*, 284(5), 34-43.

Bhatia, M.P.S., Kumar, A. and Beniwal, R. (2016). "Ontologies for software engineering: Past, present and future". *Indian Journal of Science and Technology*, 9(9).

Bizer, C., Heath, T., & Berners-Lee, T. (2011). "Linked data: The story so far". In *Semantic services, interoperability and web applications: emerging concepts* (pp. 205-227). IGI Global.

Bürger, T., & Simperl, E. (2008). "Measuring the benefits of ontologies". *In OTM Confederated International Conferences On the Move to Meaningful Internet Systems* (pp. 584-594). Springer, Berlin, Heidelberg.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). "What are ontologies, and why do we need them?". *IEEE Intelligent Systems and their applications*, 14(1), 20-26.

Cossentino, M., Gaud, N., Hilaire, V., Galland, S., & Koukam, A. (2010). "ASPECS: an agent-oriented software process for engineering complex systems". *Autonomous Agents and Multi-Agent Systems*, 20(2), 260-304.

Davis, A. M. (1993). *Software requirements: objects, functions, and states*. Prentice-Hall, Inc..

Ding, W., Liang, P., Tang, A. and Van Vliet, H. (2014). "Knowledge-based approaches in software documentation: A systematic literature review". *Information and Software Technology*, 56(6), pp.545-567.

Fornara, N., & Colombetti, M. (2009, May). "Ontology and time evolution of obligations and prohibitions using semantic web technology". In *International Workshop on Declarative Agent Languages and Technologies* (pp. 101-118). Springer, Berlin, Heidelberg.

Gigante, G., Gargiulo, F. and Ficco, M. (2015). "A semantic driven approach for requirements verification". In *Intelligent Distributed Computing VIII* (pp. 427-436).

Girardi, R., & Leite, A. (2008). "A knowledge-based tool for multi-agent domain engineering". *Knowledge-Based Systems*, 21(7), 604-611.

Graja, M., Jaoua, M., and Belguith, L. H. (2011). "Building ontologies to understand spoken Tunisian dialect". *arXiv preprint* arXiv:1109.0624.

Gruber, T. R. (1993). "A translation approach to portable ontology specifications". *Knowledge acquisition*, 5(2), 199-220.

Gruber, T. (2009). "Ontology". *Encyclopedia of database systems*, 1963-1965.

Gruninger, M., and Lee, J. 2002. "Ontology - Applications and Design". *Communications of the ACM* (45:2), pp 39 - 41.

Henderson-Sellers, B. (2011). "Bridging metamodels and ontologies in software engineering". *Journal of Systems and Software*, 84(2), 301-313.

Hofweber, T. (2018). "Logic and Ontology", *The Stanford Encyclopedia of Philosophy* (Summer 2018 Edition), Edward N. Zalta (ed.).

Kaiya, H., and Saeki, M. (2005). "Ontology based requirements analysis: lightweight semantic processing approach". In *5th International Conference on Quality Software*, 2005 (QSIC 2005).

McGuinness, D. L., and Van Harmelen, F. (2004). "OWL web ontology language overview". *W3C recommendation*, 10(10), 2004.

Nuseibeh, B., and Easterbrook, S. (2000). "Requirements engineering: a roadmap". In *Proceedings of the Conference on the Future of Software Engineering* (pp. 35-46). ACM.

Nyulas, C. I., O'Connor, M. J., Tu, S. W., Buckeridge, D. L., Okhmatovskaia, A., & Musen, M. A. (2008). "An ontology-driven framework for deploying jade agent systems". In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*-Volume 02 (pp. 573-577). IEEE Computer Society.

Obrst, L. (2003). "Ontologies for semantically interoperable systems". In *Proceedings of the twelfth international conference on Information and knowledge management* (pp. 366-369). ACM.

Paulheim, H. (2009). "Ontologies for user interface integration". In *International Semantic Web Conference* (pp. 973-981). Springer, Berlin, Heidelberg.

Ray, S. K., Singh, S., and Joshi, B. P. (2009). "Exploring multiple ontologies and WordNet framework to expand query for question answering system". In *Proceedings of the First International Conference on Intelligent Human Computer Interaction* (pp. 296-305). Springer, New Delhi.

Sadraei, E., Aurum, A., Beydoun, G., and Paech, B. (2007). "A field study of the requirements engineering practice in Australian software industry". *Requirements Engineering*, 12(3), 145-162.

Şensoy, M., Norman, T. J., Vasconcelos, W. W., & Sycara, K. (2010). "OWL-POLAR: Semantic policies for agent reasoning". In *International Semantic Web Conference* (pp. 679-695). Springer, Berlin, Heidelberg.

Stantchev, V., and Malek, M. (2010). "Addressing dependability throughout the SOA life cycle". *IEEE Transactions on Services Computing*, (2), 85-95.

Tran, Q. N. N., & Low, G. (2008). "MOBMAS: A methodology for ontology-based multi-agent systems development". *Information and Software Technology*, 50(7-8), 697-722.

Tsui, F., Karam, O., & Bernal, B. (2016). *Essentials of software engineering*. Jones & Bartlett Learning.

Uschold, M., Healy, M., Williamson, K., Clark, P., and Woods, S. (1998). "Ontology reuse and application". In *Formal ontology in information systems* (Vol. 179, p. 192). IOS Press Amsterdam.

Van der Hoek, W., and Wooldridge, M. (2008). "Multi-agent systems". *Foundations of Artificial Intelligence,* 3, 887-928.