Development of Ontologies for Reasoning and Communication in Multi-Agent Systems

Sebastian Törsleff, Constantin Hildebrandt and Alexander Fay
Institute of Automation Technology, Helmut-Schmidt University, Hamburg, Germany

Keywords: Ontology Development, Ontology Engineering, Model-driven Engineering, Multi-Agent Systems, Cyber-physical Systems.

Abstract: In future cyber-physical systems, such as smart factories and energy grids, ontologies can serve as the enabler for semantically precise communication as well as for knowledge representation and reasoning. Multi-agent systems have shown to be a suitable software development paradigm for cyber-physical systems and may well profit from harnessing ontologies in terms of reduced engineering effort and better interoperability. This contribution presents a development methodology for ontologies that enable communication and reasoning in Multi-Agent Systems for cyber-physical systems. The methodology is unique in addressing a set of requirements specific to this application domain.

1 INTRODUCTION

Multi-agent systems (MASs) have the potential to play a key role in enabling future cyber-physical systems for smart factories and energy grids (Leitao et al., 2016; Vrba et al., 2014). Major benefits of MASs are modularity, flexibility and robustness. Ontologies can provide value to MAS, in terms of serving as the foundation for inter-agent communication on the one hand (Souza et al., 2016), and knowledge bases that agents can use for reasoning on the other hand (Laclavik et al., 2006; Subercaze and Maret, 2011).

A plethora of MAS-specific methodologies and tools has been developed over the past decades (Sturm and Shehory, 2014). The use of ontologies in MAS has been partially adressed in previous work on MAS methodologies (Freitas et al., 2015). However, the methodological development and utilization of ontologies for reasoning and communication in MAS remains an open issue as the ontological needs in industrial applications cannot be met with existing ontologies or even existing standardards, notwithstanding their lack of formalization (Hodges et al., 2017). Therefore, the authors developed the *Ontologies for Reasoning and Communication in Multi-Agent Systems* methodology (OReCo). It is not designed to act as a standalone MAS development approach. Instead it can be utilized within existing methodologies that follow a model-driven engineering approach, e.g. (Chella et al., 2004; Garcia-Ojeda et al., 2008; Linnenberg and Fay, 2018). Key features of OReCo are the integration model-driven engineering artefacts, the utilization of existing ontologies to reduce engineering effort, and the integration into the popular MAS development environment JADE[1]. OReCo comprises two parts: ontology development and ontology utilization. The former is the subject of this paper. The latter deals with actual implementation tasks related to the utilization of developed ontologies in MASs and will be the subject of a future publication by the authors.

Henceforth, for the sake of brevity, we refer to the term "ontology" as a machine-readable conceptualisation of domain knowledge implemented in the OWL 2 Web Ontology Language[2] (OWL 2), or more specifically the OWL 2 DL profile. Note, that this deviates from prominent definitions of the term, which do not imply a representation language or degree of formality, e.g. (Uschold and Gruninger, 1996). Only in some cases we use the term "ontology in the broadest sense" to refer to the full spectrum of conceptualisations as defined in (Lassila and McGuinness, 2001). Occasionally, we use the terms "lightweight ontoloy" and "heavyweight ontology" to differentiate between human-centred UML class

[1] http://jade.tilab.com

[2] http://www.w3.org/TR/owl2-overview/

Törsleff, S., Hildebrandt, C. and Fay, A.
Development of Ontologies for Reasoning and Communication in Multi-Agent Systems.
DOI: 10.5220/0008355804500457
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 450-457
ISBN: 978-989-758-382-7

diagram representation and machine-readable representation based on OWL 2 DL, respectively. The communication and reasoning in MASs for cyber-physical systems that are developed using model-driven engineering methodologies will be referred to as the "application domain" of the methodology from here on. Specific desired solutions, e.g. a distribution grid automation solution, we refer to as "use cases". Furthermore, a MAS developed for a specific use case will be referred to as a "system under development". In an analogous manner, we use the term "ontology under development" (also if multiple modular ontologies are being developed at once).

The remainder of this paper is structured as follows. In section 2 we present requirements regarding the development of ontologies in the application domain that have been gathered from industry. Afterwards, in section 3, we provide an analysis of related research and discuss in how far it withstands the requirements we identified. The methodology itself is presented in section 4 and preliminary evaluation results regarding its application in a smart energy grid-related research project in section 5. Conclusions and an outlook on future work of ours are provided in section 6.

2 REQUIREMENTS

The application domain outlined in the previous section imposes a set of specific requirements which a methodology, aiming to yield suitable ontologies, has to satisfy. These requirements have been gathered with involvement from industry partners in the research project CrESt[3] (Collaborative Embedded Systems), some of which originally were presented in (Hildebrandt et al., 2018b), others were added in the meantime.

R1 Systematic Requirements Elicitation: Identifying requirements is the first step in ontology development. As in software engineering in general, requirements are a key determinant for successful ontology development. Besides serving as the foundation for designing and implementing an ontology, structured requirements also facilitate future reuse of the developed ontology. Therefore, the methodology has to entail a systematic elicitation process.

R2 Model-driven Engineering Artefact Utilization: The methodology's application domain is the model-driven engineering of MASs. Thus, diagrams that specify the message exchange and internal program logic of agents are available. These are valuable artefacts with regard to the ontology requirements and implicitly contain all intended uses of the ontology under development. Accordingly, we define the utilization of such artefacts as a requirement.

R3 Ontology Modularity: Besides reusing existing ontologies for a specific use case, it is desirable to create ontologies that can be reused themselves. Apart from systematic requirements documentation, this goal can be achieved by modularizing the ontology under development. The modularity also facilitates the usage of distinct TBoxes for different agents, thus keeping individual TBoxes lean. Accordingly, ontology modularity represents another requirement.

R4 Domain Expert Centricity: This requirement is rooted in the fact that ontology experts usually do not have the necessary knowledge to model the required domain knowledge for a specific use case. Therefore, domain experts who possess the necessary domain knowledge have to take an active and central role in ontology development.

R5 Intermediate Ontology Conceptualization: While lightweight ontologies represented as UML class diagrams are not as expressive as heavyweight ontologies, they help to conceptualize domain knowledge transparently. This facilitates the integration of domain experts in ontology development (see R4). For this reason, we define the intermediate ontology conceptualization (with UML class diagrams) as a requirement.

R6 Multi-stage Utilization of Existing Ontologies: Domain conceptualization and ontology development are time consuming tasks. In order to reduce the engineering effort, it is hence advisable to make use of existing ontologies (in the broadest sense) whenever reasonable. This ranges from industry standards as the basis for concept definitions to integrating existing heavyweight ontologies. The utilization of ontologies with varying degrees of expressiveness requires their integration at different stages of the development cycle.

R7 Defined Ontology Bridging Procedure: Aiming for reusable, modular ontologies necessitates a defined ontology bridging procedure to connect ontologies as required by the system under development.

[3] https://crest.in.tum.de/

3 STATE-OF-THE-ART ANALYSIS

Existing ontology development methodologies can be divided into three generations with increasing sophistication and maturity (Simperl and Luczak-Rösch, 2014). We limit our subsequent analysis to the third generation, i.e. methodologies that emphasize the collaborative nature of ontology development and the involvement of non-ontology experts. Table 1 summarises the analysis of these methodologies with regard to the requirements outlined in the previous section.

DILIGENT is ontology expert-centric and focuses community adaption of an initially-developed ontology (Pinto et al., 2004). Another ontology expert-centric methodology is HCOME (Vouros and Kotis, 2006), which addresses the development of “living ontologies” in the domain of knowledge-intensive communities. In contrast, UPON Lite integrates domain experts into the development (de Nicola and Missikoff, 2016). Its drawback, however, is its predominant use of tabular representations to capture domain expert knowledge. R6 (Multi-stage Utilization of Existing Ontologies) is only partially satisfied by UPON Lite in that it recommends the utilization of existing standards but does not address their integration at multiple stages of the ontology development. The innovation of the NeON methodology lies in its emphasis of a “divide and conquer” approach, in which the definition of requirement subsets paves the way for developing and utilizing ontology modules (de Figueroa Baonza, 2010). However, substantial domain expert involvement is also missing in NeON. OntologyMaturing in turn stresses domain expert centricity and suggests capturing their knowledge using lightweight ontologies, however it does not satisfy, inter alia, the modularity requirement (Braun et al., 2007). An approach that is based on established methodologies from the software engineering domain is given by SCIM (John et al., 2017), though it falls short of addressing the integration of existing ontologies at multiple stages of the development cycle. The only approach that explicitly proposes the utilization of UML diagrams for eliciting ontology requirements has been presented in (Olszewska, 2015), though it is technically not a methodology and also satisfies none of the other requirements. Therefore, it is not included in Table 1.

As can be seen, there is no methodology available that covers all or even the majority of the identified requirements. This shortcoming was the basis for the decision to develop a new ontology development methodology.

Table 1: Requirements satisfaction of existing ontology development methodologies.

	R1	R2	R3	R4	R5	R6	R7
DILIGENT			X				X
HCOME	X		X				X
UPON Lite		X		X		(X)	
NeON	X	X	X				X
Ontology Maturing				X	X		
SCIM	X			X	X		

4 METHODOLOGY

As outlined in the introduction, the overall OReCo methodology comprises two parts: ontology development and ontology utilization. In this contribution we focus on the ontology development part, which is divided into three stages, each of which comprising of multiple steps as depicted in Figure 1. Indicated by the coloured dots, multiple roles are involved in each step: the domain expert has deep knowledge of the specific use case, was involved in the design of the system under consideration and is aware of relevant (de-facto) industry standards; the software engineer is the end user of the ontology insofar as utilizing it in the implementation of the agents; the ontology expert is proficient in ontological engineering and provides technical support.

The methodology presented hereafter builds on previous publications of the authors (Hildebrandt et al., 2018a, 2018b). While these publications were rather generic in scope (covering, e.g., the development of UML profiles), OReCo is limited to the application domain as defined in section 1. This allows omitting irrelevant steps of the generic methodology and in turn emphasising aspects specific to OReCo’s application domain.

OReCo employs a waterfall-like life-cycle model. This is based on the rationale that the ontology’s scope is determined by the design of the system under development. That is, if the system design poses specific requirements, the ontology has to satisfy these requirements. An iterative-incremental approach for the ontology development would thus be inappropriate. This does of course not inhibit an iterative-incremental approach being applied to the overall system under development. In such cases, multiple iterations of the OReCo methodology would be driven by the enclosing development methodology while itself remaining waterfall-like.

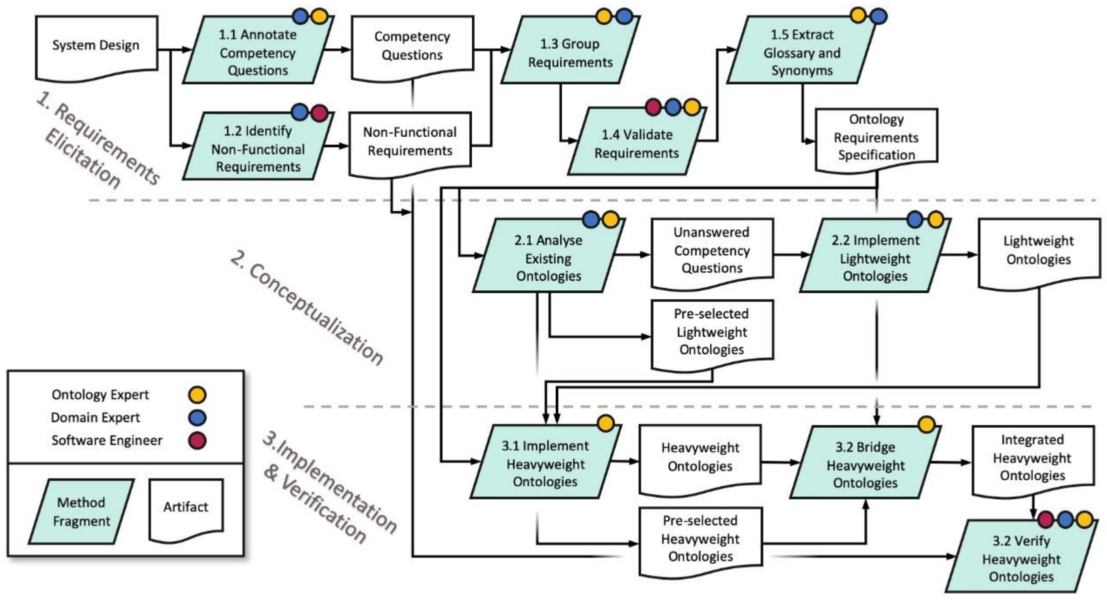
System Design
1.1 Annotate Competency Questions
1.2 Identify Non-Functional Requirements
Competency Questions
Non-Functional Requirements
1.3 Group Requirements
1.4 Validate Requirements
1.5 Extract Glossary and Synonyms
Ontology Requirements Specification
1. Requirements Elicitation
2. Conceptualization
2.1 Analyse Existing Ontologies
Unanswered Competency Questions
Pre-selected Lightweight Ontologies
2.2 Implement Lightweight Ontologies
Lightweight Ontologies
3.Implementation & Verification
3.1 Implement Heavyweight Ontologies
Heavyweight Ontologies
Pre-selected Heavyweight Ontologies
3.2 Bridge Heavyweight Ontologies
Integrated Heavyweight Ontologies
3.2 Verify Heavyweight Ontologies
Ontology Expert
Domain Expert
Software Engineer
Method Fragment
Artifact

Figure 1: Overview of the OReCo methodology's ontology development part.

Note, that OReCo does not prescribe a specific OWL 2 profile. However, due to its wide tool support and decidability (Hitzler et al., 2010), the OWL 2 DL profile can be considered the default implementation language for ontologies developed using OReCo.

4.1 Requirements Elicitation

The specification of ontology requirements is partly based on the NeON methodology (Suárez-Figueroa et al., 2009). Some of the tasks in the ontology requirements specification according to NeON can be omitted, since their results are the same for any use case in OReCo's application domain.

Table 2: Generic results of task 1-2 of the NeON Ontology Requirements Specification for OReCo's application domain.

Purpose	Provide the foundation for reasoning and communication in the multi-agent system under development
Scope	All classes and properties required for communication and reasoning in the system under development
Implementation Language	Intermediate: UML class diagrams Final: OWL
Intended End Users	Software engineers (and, strictly speaking, agents)

This holds for task 1 and 2, whose results are shown in Table 2. Further tasks of the ontology requirements specification according to NeON have been modified to take into account specific needs of the application domain and to improve overall validity with regard to the requirements discussed in section 2.

Step 1.1. *Annotate Competency Questions* processes artefacts from model-driven engineering, i.e. the system design comprising UML diagrams such as sequence diagrams (covering the message exchange between agents) and activity diagrams (covering the internal application logic of agents). These diagrams are annotated with competency questions (CQs) and answers using a custom UML profile, which facilitates their automated export into a tabular representation including a reference to the respective agent. Strictly speaking, the annotation is limited to informal CQs (Gruninger and Fox, 1995). Later on, these informal CQs will be the basis for formal SPARQL queries that are used for verification once the ontology has been implemented. Note that task 3 "Identify intended end uses" of the NeON methodology has been omitted. This is due to the fact that the intended uses are already given by the system design.

Step 1.2 *Identify Non-Functional Requirements* deals with aspects that are not covered by CQs, e.g. a specific ontology naming convention or the mandatory use of industrial standards. The latter could result from the system design, e.g. if a agent interacts with a technical system that requires the use of a specific protocol. Non-functional requirements are recorded in tabular format.

To lay the foundation for the reuse of existing ontologies and the development of modular ontologies, the requirements are clustered in step 1.3

Group Requirements. Useful groups are subdomains such as technical properties of a battery storage or generic concepts such as units of measure. Further grouping may be performed in case of agents requiring very specialized knowledge. Such groups, by facilitating ontology segmentation, can help reduce the TBox size (and thus inprove the reasoning performance) of agents with distinct knowledge bases. This step can also comprise the definition of a group hierarchy, which determines the development sequence in case the ontology under development is composed of multiple interdependent ontologies.

Step 1.4 *Validate Requirements* serves to ensure that the requirements are complete, consistent, verifiable, understandable, unambiguous, concise and modifiable. Guidance on how to evaluate these criteria are provided in (Suárez-Figueroa et al., 2009). An additional criterion is wether a requirement is implementable, e.g.. if a CQ can be answered with an ontology. This characteristic has been added since non-ontology experts might add CQs that cannot be covered by an ontology with reasonable effort. In case of invalid requirements, step 1.1 and/or 1.2 of the methodology have to be revisited (not depicted in Figure 1). Note that the prioritization of requirements as per the NeON methodology has been omitted. This is based on the rationale that the ontology under development serves as an enabler for reasoning and communication in the system under development as prescribed by the system design. This means that any requirement that is not met by the ontology will have negative repercussions with regard to the system under development. Accordingly, it is assumed that all requirements have equal priority. Potential dependencies between ontology segments that require a specific development sequence are determined in step 1.3.

In step 1.5 *Extract Glossary and Synonyms*, the foundation is laid for the subsequent analysis of existing ontologies. The glossary comprises three parts. The first part shows terms and frequency based on the CQs, the second one based on the related answers, and the third part identifies objects that can be seen as instances of other classes. Additionally, synonyms are recorded for each glossary term to facilitate the survey of existing ontologies. The final result of the requirements elicitation stage is an *Ontology Requirements Specification* covering all requirements for the ontology under development.

4.2 Conceptualization

The conceptualization stage begins with step 2.1 *Analyse Existing Ontologies*. In this step, the ontology and domain experts first perform a survey to identify suitable ontologies (in the broadest sense) based on the glossary developed in step 1.5. This will result in a pool of ontologies (in the broadest sense) potentially ranging from plain dictionaries to heavyweight ontologies. Ideally, the survey yields a set of heavyweight ontologies that satisfy all requirement (CQs and non-functional). In that case one could jump straight to step 3.2 of the methodology. The more realistic case, which we assume henceforth, is the identification of both lightweight and heavyweight ontologies, whereas some CQs remain unanswered. These remaining CQs will be processed in step 2.2 *Implement Lightweight Ontologies*. Here, domain experts, supported by ontology experts, conceptualize an ontology addressing the remaining CQs as a UML class diagram. We recommend performing this step with the concept of so-called content ontology design patterns in mind (Hitzler et al., 2016). These are small modular and extendible ontologies which can be used in multiple use cases. This approach is specifically useful for industry standards that have a high likelihood of being reused, but also in cases where generic domain conceptualizations are useful. For instance, consider a network topology ontology for energy grids that can be extended for the domain gas, electricity etc. Further orientation marks with regard to the segmentation of the lightweight ontology implementation task are given by the requirement groups defined in step 1.3. Naturally, the pre-selected lightweight ontologies need to be considered in this step to avoid overlaps.

4.3 Implementation and Verification

The last stage of the development begins with step 3.1 *Implement Heavyweight Ontologies*. Inputs for this step are the pre-selected and newly implemented lightweight ontologies as well as the ontology requirements specification. The latter provides information not included in the lightweight ontologies, e.g. naming conventions. The groundwork for the implementation of content ontology design patterns has been laid in step 2.2. Thus, this step can be completed by sequential transformation of each lightweight ontology into a heavyweight ontology. Guidelines for the actual modelling in OWL can be found in (Pinto et al., 2004).
Figure 2 shows an exemplary scenario in which the overall ontology under development comprises three TBoxes that are being used by two distinct agents. In this scenario, *Agent1*'s knowledge base imports *TBox1* and *TBox2*. Accordingly, “TBox bridges”, i.e.

TBox statements that connect existing TBoxes, are required between *TBox1* and *TBox2* as well as between *TBox2* and *TBox3*. These statements are symbolized by the white lines within the agent's knowledge bases. Technical alternatives and guidance on how to bridge ontologies can be found in (Hodges et al., 2017).

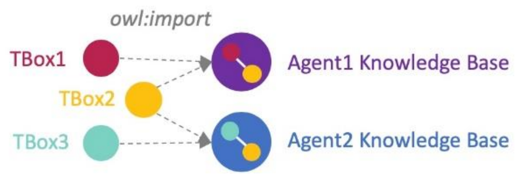

Figure 2: Exemplary utilization and bridging of TBoxes in different agents.

Step *3.2 Bridge Heavyweight Ontologies* deals with integrating the resulting heavyweight ontologies. A key determinant of this step is the utilization of these ontologies by the agents, which is captured in the ontology requirements specification.

Lastly, in step 3.3 *Verify Heavyweight Ontologies* the developed heavyweight ontologies are verified. First, a consistency check of the resulting ontology is performed using a suitable reasoner, e.g. HermiT[4]. If this is successful, SPARQL queries are defined for the CQs identified earlier. The results of the query need to be consistent with the respective answers of each CQ in order for this step to succeed. Otherwise, step 3.1 or 3.2 have to be revisited.

5 EVALUATION

This section deals with a preliminary evaluation of the proposed methodology and its satisfaction of the requirements presented in section 2. The use case for the evaluation was a MAS-based low-voltage distribution grid automation solution that had been developed in Agent.HyGrid[5], a research project dealing with smart energy grids. UML diagrams specifying the message exchange between agents and their inter program logic, i.e. the system design, formed the basis for the ontology development, illustrating the satisfaction of requirement **R2**. It was possible to group the identified CQs based on their attribution to agents. For instance, one group comprised CQs focussing the grid topology and was only relevant to one agent, while another group adressed the flexibility of distributed energy resources, which multiple agents exchange information about. This provided the foundation for the concurrent execution of subsequent conceptualisations and underlines the methodology's satisfaction of **R1**. As no suitable ontology for flexibilities could be identified by the domain experts in step 2.1, they modelled a lightweight ontology from scratch, based on the unanswered CQs (**R4** and **R5**). In contrast, the IEC 61970-501 standard was identified as suitable for conceptualizing the grid topology, which conveniently had been implemented as a heavyweight ontology already (Schumilin et al., 2017). After the lightweight ontology for flexibilities had been implemented as a heavyweight ontology in step 3.1, it was bridged with the grid topology ontology in step 3.2 (**R6** and **R7**). While the flexibility ontology is now used by all agents, the grid topology is only used by one agent, which demonstrates the advantageuousness of the ontology modularity support by our methodology (**R3**).

6 CONCLUSIONS AND OUTLOOK

Ontologies have the potential to significantly reduce the engineering effort required in MAS development. OReCo is a methodology comprising two parts: developing ontologies and utilizing ontologies for reasoning and communication in MASs for cyber-physical systems. In this contribution, we presented the first part, which is unique in addressing requirements specific to this application domain. While providing significant value in its current state already, additional research is required to improve usability and reduce the required engineering effort even further.

In this spirit, the authors are currently conducting a systematic survey of existing ontologies in the smart grid space. The results of this survey will serve as the foundation for a software tool that supports ontology and domain experts in the development process by providing recommendations for reuse of existing ontologies based on the glossary in step 2.1 *Analyse Existing Ontologies*.

Further improvements to the methodology will also result from ongoing and planned evaluation activity in the context research projects dealing with smart factories and energy grids.

[4] http://www.hermit-reasoner.com

[5] http://www.agent-hygrid.net

ACKNOWLEDGEMENTS

The contribution presented in this paper was funded by the German Federal Ministry of Education and Research under grant number 01IS16043Q – Collaborative Embedded Systems (CrESt).

REFERENCES

Braun, S., Schmidt, A., Walter, A., 2007. Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering, in: Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at the 16th International World Wide Web Conference (WWW2007).

Chella, A., Cossentino, M., Sabatucci, L., 2004. Tools and Patterns in Designing Multi-Agent Systems with PASSI. WSEAS Trans. Commun. 3, 352–358.

de Figueroa Baonza, M. del C.S., 2010. NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse.

de Nicola, A., Missikoff, M., 2016. A lightweight methodology for rapid ontology engineering. Commun. ACM 59, 79–86. https://doi.org/10.1145/2818359

Freitas, A., Bordini, R.H., Meneguzzi, F., Vieira, R., 2015. Towards integrating ontologies in multi-agent programming platforms, in: Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2015). IEEE, pp. 225–226. https://doi.org/10.1109/WI-IAT.2015.207

Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., Valenzuela, J., 2008. O-MaSE: A Customizable Approach to Developing Multiagent Development Processes, in: Luck, M., Padgham, L. (Eds.), Agent-Oriented Software Engineering VIII: 8th International Workshop, AOSE 2007. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-79488-2_1

Gruninger, M., Fox, M.S., 1995. Methodology for the Design and Evaluation of Ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI'95).

Hildebrandt, C., Törsleff, S., Bandyszak, T., Caesar, B., Ludewig, A., Fay, A., 2018a. Ontology engineering for collaborative embedded systems - Requirements and initial approach. CEUR Workshop Proc. 2060, 57–66.

Hildebrandt, C., Törsleff, S., Caesar, B., Fay, A., 2018b. Ontology Building for Cyber-Physical Systems : A domain expert-centric approach, in: 14th IEEE International Conference on Automation Science and Engineering (IEEE CASE 2018).

Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A.A., Presutti, V., 2016. Ontology Engineering with Ontology Design Patterns: Foundations and Applications. IOS Press.

Hitzler, P., Krötzsch, M., Rudolph, S., 2010. Foundations of Semantic Web Technologies. Chapman & Hall.

Hodges, J., García, K., Ray, S., 2017. Semantic Development and Integration of Standards for Adoption and Interoperability. Computer (Long. Beach. Calif). 50, 26–36.

John, S., Shah, N., Stewart, C., Samlov, L., 2017. Software Centric Innovative Methodology for Ontology Development, in: Proceedings of the 9th International Conference on Knowledge Engineering and Ontology Development (KEOD 2017). pp. 139–146. https://doi.org/10.5220/0006482901390146

Laclavik, M., Balogh, Z., Babik, M., Hluchy, L., 2006. AgentOWL: Semantic Knowledge Model and Agent Architecture. Comput. Informatics 25, 419–437.

Lassila, O., McGuinness, D.L., 2001. The Role of Frame-Based Representation on the Semantic Web, Technical Report KSL-01-02. https://doi.org/10.1177/0363 54659702500204

Leitao, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T., Colombo, A.W., 2016. Smart Agents in Industrial Cyber – Physical Systems. Proc. IEEE 104, 1068–1101. https://doi.org/10.1109/JPROC.2016.2521931

Linnenberg, T., Fay, A., 2018. Software Engineering for Agent Based Energy Systems, in: 14th IEEE International Conference on Automation Science.

Olszewska, I.J., 2015. UML Activity Diagrams for OWL Ontology Building, in: Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K2015) - Volume 2: KEOD. pp. 370–374. https://doi.org/10.5220/0005633103700374

Pinto, H.S., Staab, S., Tempich, C., 2004. DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies, in: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04).

Schumilin, A., Stucky, K.U., Sinn, F., Hagenmeyer, V., 2017. Towards ontology-based network model management and data integration for smart grids, in: 2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES). https://doi.org/10.1109/MSCPES.2017.8064530

Simperl, E., Luczak-Rösch, M., 2014. Collaborative ontology engineering: A survey. Knowl. Eng. Rev. 29, 101–131. https://doi.org/10.1017/S0269888913000192

Souza, M., Moreira, A., Vieira, R., Meyer, J.-J.C., 2016. Integrating Ontology Negotiation and Agent Communication, in: Tamma, V., Dragoni, M., Gonçalves, R., Ławrynowicz, A. (Eds.), Ontology Engineering. Springer International Publishing, Cham, pp. 56–68.

Sturm, A., Shehory, O., 2014. The Landscape of Agent-Oriented Methodologies, in: Shehory, O., Sturm, A. (Eds.), Agent-Oriented Software Engineering - Reflections on Architectures, Methodologies, Languages, and Frameworks. Springer-Verlag Berlin Heidelberg, pp. 137–154. https://doi.org/10.1007/978-3-642-54432-3

Suárez-Figueroa, M.C., Gómez-pérez, A., Villazón-Terrazas, B., 2009. How to Write and Use the Ontology Requirements Specification Document How to Write and Use the Ontology Requirements, in: Meersman, R., Dillon, T., Herrero, P. (Eds.), On the Move to Meaningful Internet Systems. Springer-Verlag, Berlin Heidelberg, pp. 966–982. https://doi.org/10.1007/978-3-642-05151-7

Subercaze, J., Maret, P., 2011. Programming Semantic Agent for Distributed Knowledge Management, in: Elci, A., Koné, M.T., Orgun, M.A. (Eds.), Semantic Agent Systems - Foundations and Applications. Springer-Verlag Berlin Heidelberg, pp. 47–65.

Uschold, M., Gruninger, M., 1996. Ontologies : Principles , Methods and Applications. Knowl. Eng. Rev. 11, 93–136. https://doi.org/10.1.1.111.5903

Vouros, G.A., Kotis, K., 2006. Human-centered ontology engineering: The HCOME methodology. Knowl. Inf. Syst. 10, 109–131. https://doi.org/10.1007/s10115-005-0227-4

Vrba, P., Marik, V., Siano, P., Leitao, P., Zhabelova, G., Vyatkin, V., Strasser, T., 2014. A review of agent and service-oriented concepts applied to intelligent energy systems. IEEE Trans. Ind. Informatics 10, 1890–1903. https://doi.org/10.1109/TII.2014.2326411