A methodology for creating reusable ontologies

Thomas Frühwirth*, Wolfgang Kastner*, Lukas Krammer†

* Institute of Computer Engineering, TU Wien
Automation Systems Group, 1040 Vienna, Austria
{tfruehwirth, k}@auto.tuwien.ac.at

† Siemens AG Österreich
1210 Vienna, Austria
lukas.krammer@siemens.com

***Abstract*—Applications within the Internet of Things (IoT) increasingly build upon Semantic Web technologies to ensure interoperability of devices and applications. Hereby, a central aspect is knowledge representation, and in particular how devices can store and exchange semantic data. This paper defines and discusses a methodology for creating application-specific ontologies for the IoT. The two main objectives are to reuse existing knowledge on one hand and to derive reusable ontologies on the other hand. Thereby, a multi-agent system for switching optimization in the smart energy domain serves as a motivating example.**

I. Introduction

The ever-growing number of standards, protocols, data formats, types of devices, and variety of different applications within the Internet of Things (IoT) causes severe interoperability problems. Things within the IoT are often designed application-specific and are not able to exchange data and cooperate effectively, which is a significant barrier for the further development of Machine-to-Machine (M2M) communications. Semantic Web technologies may be applied not only in the World Wide Web (WWW) but also in the IoT to address these issues, which is then called the Web of Things (WoT) [1].

Knowledge representation plays a key role in Semantic Web applications. A number of knowledge representation systems is introduced and briefly discussed in Section II. The most powerful among them are *ontologies*, which are already heavily used for many different kinds of applications within and outside the IoT. Section III discusses a number of different existing ontology modeling techniques that focus on reusing existing sources. A methodology that concentrates on combining sources to reusable ontologies is introduced and discussed in Section IV. Section V exemplifies this methodology with a motivating application scenario. Finally, benefits and drawbacks of the approach, as well as ideas for future work, are addressed in Section VI.

II. Knowledge representation

Playing a major role in building context-aware, intelligent and more agile systems, knowledge representation is considered a key element within the IoT [2]. The expressiveness of various knowledge representation systems available differs.

Catalog: collection of terms without any additional information about the terms themselves or relations between them

Glossary: collection of terms amended by human-readable explanations; such as this list

Classification/Taxonomy: model for assigning terms to classes; classes are often hierarchically ordered

Semantic network: collection of terms and arbitrary relations between them, without formal semantics

Ontology: collection of terms and relations between them, including formal semantics

The predominant system for knowledge representation within the IoT and therefore also the main focus of this work are ontologies as defined by the World Wide Web Consortium (W3C). Thereby, an ontology is simply a graph consisting of nodes representing specific objects (individuals) or classes of objects, and edges representing relations between nodes. The graph in turn is nothing more than an elegant and illustrative way for representing the triples it contains, whereby each relation in conjunction with the nodes it connects constitutes such a triple. Triples are of the form *Subject-Predicate-Object*, such as in *"Paper" hasColor "White"*.

Even though ontologies are listed as a separate class of systems for knowledge representation, the expressiveness of ontologies themselves differ w.r.t. the Semantic Web language used for their representation (storage and exchange). Ordered from least expressive to most expressive these are Resource Description Framework (RDF), Resource Description Framework Schema (RDFS), RDFS-plus, and Web Ontology Language (OWL) [3]. And, finally, the most commonly used Semantic Web language OWL comes in different versions: OWL Lite, OWL Description Logic (DL) and OWL Full.

To make them easily accessible via the Web, ontologies are stored in ontology documents (typically in eXtensible Markup Language (XML) file format), just as HyperText Markup Language (HTML) documents. The process of generating such a textual representation of an existing ontology is called *serialization*. Naturally, textual representations of ontologies have to be *parsed* to load them into programs and perform operations on them. A powerful mechanism integrated in ontologies is the possibility to *import* other, already existing ontology files by referring to their online-accessible textual representation. Once another ontology is imported, its triples can be used directly, expanded with further information, or even linked to other imported ontologies. Of course, the powerful concept of ontologies and the corresponding tools can also be used offline.

III. State of the art in ontology design

Preliminary, ontology design has been driven by the idea to generate a single ontology covering the complete application scenario. It typically follows a step-wise approach, which often motivates the use of existing ontologies but, on the other hand, does not support the generation of reusable ontologies. The probably best-known example of this approach is "Ontology Development 101: A Guide to Creating Your First Ontology" [4]. It describes a methodology to ontology design structured into seven simple steps. "A lightweight methodology for rapid ontology engineering" [5] is another (six-step) approach falling into this category of ontology design methodologies.

Later on, ontology design methodologies were developed that start with a small or even empty ontology, and add other ontologies and concepts incrementally. An example of this category is "An Incremental and Iterative Process for Ontology Building" [6]. Yet again, reusing existing ontologies and other sources is a core aspect of the ontology design but deriving reusable ontologies along the way is not of particular interest.

The Open Semantic Framework (OSF) [7] strictly distinguishes between two types of ontologies: *core ontologies* provide general information (e.g. engineering units, weather data), and *Knowledge Packs (KPs)* add domain-specific concepts. This is major step away from monolithic ontologies as used in many of today's ontology-based frameworks.

IV. Methodology – creating reusable ontologies

A methodology that focuses on deriving reusable ontologies as well as reusing existing ontologies is illustrated in Fig. 1. It follows a "divide and conquer" or "divide and combine" approach using different levels. At the very beginning, the application-specific ontology is created but at this stage no concepts are added. Instead, it is divided in a top-down approach into smaller (Tier 1) ontologies, each intended to cover a well-defined subset. In general, Tier n ontologies are again divided into Tier $n+1$ ontologies. This process stops for a certain Tier n ontology for one of the following three reasons: (1) it is already covered by an existing ontology, (2) it is already covered by another source of structured and complete information (e.g. books, papers), or (3) it cannot reasonably be divided any further. The quality of each potentially useful source of information is thereby rated and documented. According to the reason why the process stopped at a certain point, (1) the ontology is either used directly, or (2) & (3) the ontology is created following standard ontology modeling techniques. In a bottom up approach, Tier $n+1$ ontologies are then combined (imported to) Tier n ontologies and extended with concepts that are not present in the Tier $n+1$ ontologies but arise from their combination. Certain basic Tier $n+1$ ontologies, such as modeling timing or engineering unit concepts, may thereby be imported in multiple Tier n ontologies. Others may not be used at all, e.g. because they were found to be redundant or too complex. Combining ontologies is continued until, finally, the application-specific ontology has been derived. Ontologies on each level should be self-contained to allow them to be reused in ontologies for other applications that are generated following the same methodology. The individual steps of the methodology are briefly discussed in the following sections.

A. Methodology step: divide

The step of dividing ontologies into smaller parts is heavily driven by the idea to reuse existing ontologies and other sources of information. Thus, developers should analyze available material and choose their sub-ontologies accordingly. Thereby, information already being available as an ontology and probably in some standardized ontology document format is preferred to other kinds of information. Nevertheless, developers should not completely ignore other sources of information and consider to include them in this step as well. In the following, a number of valuable sources for existing ontologies as well as additional valuable information is presented:

Conventional search engines: Conventional search engines focus on human-readable information such as HTML, Portable Document Format (PDF) or similar. Nevertheless, they often include all types of online-available documents in their search results and even provide options to search for specific file types (e.g. *.rdf or *.owl).

Ontology search engines: On the other hand, there are search engines specifically targeting ontologies. They provide meta-information about the ontology itself, e.g. the number of classes, properties and individuals contained.

Scientific publications: Ontologies still are a rather scientific way of structuring information, and the concepts many ontologies are built upon are published in papers and theses. Sometimes, the underlying publications refer to the corresponding online-available ontology documents.

Databases Databases in general and their structure in particular often may directly be translated to ontologies or at least provide a good starting point. Even if information is available in a machine-readable and structured form, it is important to ensure that there are no contradictions.

Human-readable documents: Plenty of information about various domains is available in standards, specifications, books, articles (scientific or other), company-internal documents, diagrams or similar. For using this information as basis for an ontology, it is important that the information satisfies certain conditions: (1) the information needs to be available in a structured form, (2) the information covers the required set of information, (3) the information does not lead to a contradiction. A typical example are existing Unified Modeling Language (UML) diagrams.

The divide step is performed repeatedly at the beginning of the workflow. Once reasonably small sub-ontologies are derived, it is not performed again but the model step and the select & combine step are applied alternately on each level.

B. Methodology step: model

Existing ontologies do not need to be re-modeled, however, in many cases the information will not be available in a standard file format. Existing ontology modeling techniques (cf. Section III) are used to generate ontologies from this information and also to model new concepts which are not found in any existing source. In addition, concepts that emerge from selecting and combining a number of lower level ontologies are added in this step. The model step is performed on each level of the methodology and alternates with the select & combine step until the application-specific ontology is derived.

C. Methodology step: select & combine

In this step, Tier $n+1$ ontologies are analyzed and rated to select suitable subsets that are then combined to Tier n ontologies. Thereby, the rating does not follow a strict, predetermined scheme but is subject to the developer's experience. A number of considerations has to be taken into account:

- Comprehensive ontologies reduce the effort of implementing missing concepts.
- On the other hand, concepts that are not used by the application cause unnecessary overhead in memory consumption and processing power.

- Preferably, existing sources should be well known and well accepted by the corresponding domain experts.
- In general, ontologies already used in other applications are preferred to others.
- Tier $n+1$ ontologies may contain information that is relevant for multiple Tier n ontologies.

Selecting the ”best” subset of ontologies is not a straightforward procedure and – taking into account the required effort – in most cases it is not practicable. However, results from the previous analysis provide a good basis. The preferable approach for selecting and combining Tier $n+1$ ontologies is to select each ontology only if it adds considerable value to the corresponding Tier n ontology. It is thereby necessary to re-evaluate the rating of each ontology after each step to avoid the selection of multiple, similar ontologies. After having selected and combined all relevant Tier $n+1$ to Tier n ontologies, the methodology continues with the model step on level Tier n.

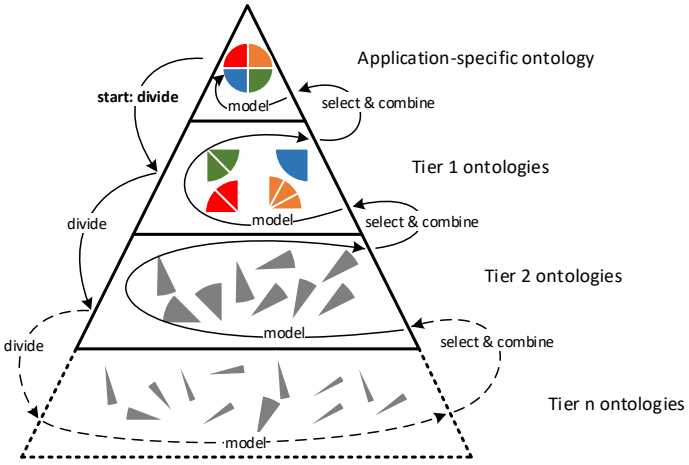

Fig. 1. Methodology for ontology reuse

V. Example following the methodology

The exact number of levels of the methodology presented in the previous section is not predetermined and may also be fixed during the process. However, a hierarchy of three types of ontologies, as illustrated in Fig. 2 has proven to be useful in practice and will also be used in this example. While the application-specific ontology keeps its name, Tier 1 ontologies will be referred to as *domain ontologies* or *domains*, and Tier 2 ontologies will be referred to as *fragment ontologies* or *fragments*. The individual steps of the methodology can be documented using a tabular, graphical or any other form of notation, adding information along the way. The methodology will be exemplified in the following, but first a motivating application scenario shall be presented.

Multi-Agent Systems (MASs) heavily build upon ontologies, which allow the individual agents to share a common understanding. Agents are situated in some *environment* and exchange *semantically meaningful messages* to fulfill their goals [8]. Thus, the ontology used within a MAS contains – to some extent – a model describing the agents' perceptions of the application domain and their capabilities to interact with the environment. Each agent offers services to other agents

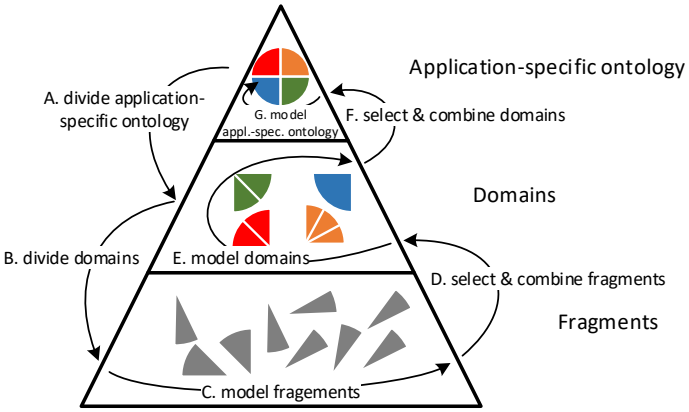

Fig. 2. Three-level methodology for the motivating example

and these services might be related to different parameters, e.g. to the costs of invoking a service, associated Quality of Service (QoS) parameters, security or even dependability considerations. Finally, agents may offer their services to other agents via different communication protocols. The motivating example for this paper shall be ”A distributed multi-agent system for switching optimization in low-voltage power grids” [9].

A. Divide application-specific ontology

The goal of this first step is to reduce the complexity of the given problem set by separating it into multiple domains. The exact number and types of domains depend on the complexity of the application and on the different views developers might have on their problem sets. The description of the motivating example in the previous section already is a specific view on MASs. However, it provides a good starting point to divide the application-specific ontology into smaller domain ontologies:

Agents and services: Agents in the sense of MAS-theory typically have a certain structure and a number of attributes: a name, an address, etc. They are often closely related to the services they offer and are therefore part of the same domain. Services themselves also may have a name, information required to invoke them, etc. [10]

Power system resources: As the intended MAS operates in the Smart Grid (SG)-domain, power system resources are a major part of the agents' environment. Nevertheless, power system resources are not directly related to agents but merely one field of application where MASs can be applied. Therefore, they form a separate domain.

Communication protocols: Communication protocols form another domain because they are used not only by agents but also by other devices in the IoT.

Rating: The rating domain includes QoS and dependability attributes, but also concepts to assign them to individuals of the three other domains, e.g. the availability of a service, the reliability of an electric switch, or the integrity features of some communication protocol. Thus, the rating domain allows agents to consider this additional information in their decision-making process.

As a starting point, the four identified domains are arranged in a simple graph as shown in Fig. 3. This graph will be complemented with additional information along the process.

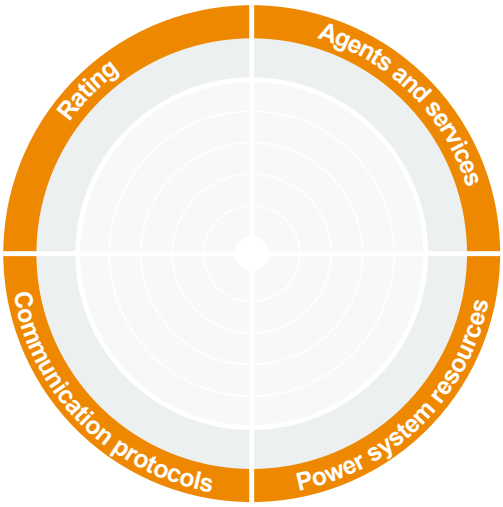

Fig. 3. Step 1: Domains are identified and added to the graph.

B. Divide domains

Considering the motivating example, there is a great amount of fragments which could be selected (and therefore added to Fig. 3). The most promising ones have been added in Fig. 4. Their discussion is grouped by domains:

Agents and services domain: Despite of being a main field of application for ontologies, there are not many ontologies specifically focusing on MASs and agents on a meta-level, and hardly any are available in standard ontology file formats. In their core ontology for the Onto2MAS framework [11], Donzelli et al. define an agent primarily based on its behavior and its role within the MAS, which is a reasonable approach for their intended use case of generating MASs directly from information given within the ontology.

As most publications about MASs in the SG-field do, Donzelli et al. themselves refer to the Foundation for Intelligent Physical Agents (FIPA) specifications for specifying the message types, which agents use for communication. Besides this definition of the FIPA-Agent Communication Language (ACL) [12], FIPA also gives a quite general description of agents in their FIPA agent management specification [13]. Although not being available in standard file formats, in this specification FIPA introduces the fipa-agent-management ontology, which defines the basic components of a FIPA-compliant MAS. Unfortunately, the ontology itself is not very sophisticated and would impose severe limitations if one would directly translate it to an W3C ontology: most parameters are specified to be of type ”string” or ”set of string”. Considering, for example, the languages supported by a service, creating dedicated nodes which represent the corresponding languages and referencing them in the ontology is beneficial over listing the supported languages as a ”set of strings”.

The concept of agents can also be found in other ontologies, such as the COmmon Semantic MOdel (COSMO)[1], the FIESTA-IoT[2] ontology and the DOLCE+DnS UltraLite (DUL)[3] ontology. However, they all do not specify the inner structure of an agent as detailed as FIPA does and are therefore not very suitable for the present use case.

Power system resources domain: The Common Information Model (CIM) is probably the best-known model for power system resources. The standard document IEC 61970-501 [14] specifies the CIM in RDFS format[4].

The Prosumer-Oriented Smart Grid (ProSG) ontology [15] has a stronger focus on the consumer level and is, therefore, less comprehensive regarding power systems resources. However, it might be a valuable source once the example use case is extended towards Distributed Energy Resources (DERs) in particular because of its availability in OWL file format[5].

Communication protocols domain: A core aspect of any MAS is cooperation between agents which heavily builds upon communication. While application layer protocols are often specifically designed for the MAS (s.a. FIPA), they typically rest upon existing lower-layer protocols, e.g. Transmission Control Protocol (TCP), Internet Protocol (IP), and corresponding Link layer protocols, probably even including security mechanisms s.a. Transport Layer Security (TLS). While typically not being integral part of the MAS specification, in an open MAS, knowledge about which lower level communication protocols and security mechanisms are supported by each agent is still very important to select a suitable communication partner. A typical source for such information are ontologies about the IoT or the Internet in general, s.a. the IoT Network ontology[6] which covers a number of wired and wireless link technologies.

Another valuable starting point for an ontology about communication protocols is provided by Jablonski’s ”Guide to Web Application and Platform Architectures” [16]. Therein, Jablonski provides a quite comprehensive classification of Internet standards and technologies. The root classification entities are Interaction, Logic, Security, Data Management, Description, Presentation, Export/Import Interface, and Platform Software. These root classification entities are then further subdivided into Category Bags (CBs). The author himself states that the classification is incomplete and can be extended by additional CBs. Finally, a number of existing Internet standards and technologies is assigned to each CB, and additional items (e.g. upcoming protocols) can easily be added.

Rating domain: Priority in MASs is on cooperation between

[1] http://micra.com/COSMO/COSMO.owl
[2] http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot.owl
[3] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[4] http://www.langdale.com.au/CIMXML/
[5] http://data-satin.telecom-st-etienne.fr/ontologies/smartgrids/proSGV3/ProSGV3.owl
[6] http://srg.seecs.nust.edu.pk/iotchecker/Finalized\%20Ontologies/IoTNetwork.owl

agents, which requires finding and invoking services. However, the same service may be offered by various agents probably via different interfaces. A rating scheme is required to select the most appropriate one. For this reason, FIPA specifies the fipa-qos ontology [17].

When dealing with critical infrastructure, traditional QoS attributes often are extended by aspects targeting the notion of dependability as defined by Avizienis et al. [18]. Their structured view on dependability in the form of a dependability tree, and the corresponding statements about the relationships between dependability attributes, means, and threats provide an excellent starting point for an ontology about dependability.

In regards of W3C-compliant ontologies, the Quality of Service Modeling Ontology (QoS-MO)[7] is a very sophisticated approach [19]. It includes not only a variety of QoS and dependability attributes but also holds concepts for rating these attributes, which enables agents to select a suitable service.

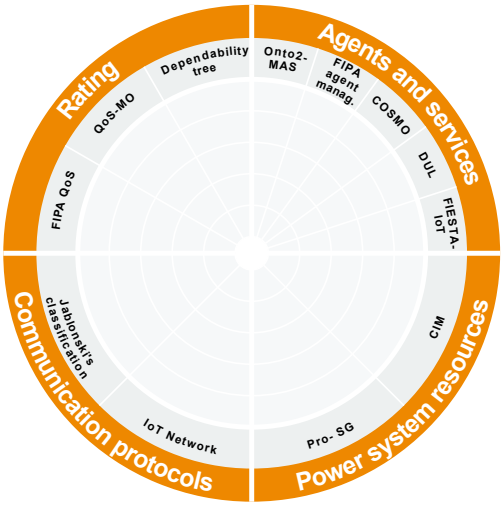

Fig. 4. Step 2: Fragments are added to the graph and assigned to the corresponding domains.

C. Model fragments

Identified fragments that are not already available as ontologies in any standardized ontology file format are modeled using existing ontology modeling techniques. The ontologies derived until this point are not at all specific to the use case but rather reflect small portions of the identified domains in very abstract and general ways (e.g. attributes, threats, and means of dependability; main elements of an agent; known IoT communication protocols; power system resources). This course of action increases the likelihood of deriving fragment ontologies that are also useful for researchers beyond the exact same field of expertise, e.g. someone who specialized on MASs for production facilities, or developers who want to extend their ontologies with dependability features.

[7]The owl-representation of the QoS-MO ontology is currently offline but kindly was provided by Mr. Tondello upon request.

D. Select & combine fragments

For selecting possibly relevant sources of information, it has proven useful to collect and rate them according to their suitability for the example application scenario. The exact ratings are subject to the authors' experience. The rating for each fragment has been added to Fig. 4, resulting in Fig. 5. The height of each bar, starting from the center of the diagram, directly corresponds to the associated rating, i.e. high bars $\hat{=}$ high ratings while low bars $\hat{=}$ low ratings. For instance, the Onto2MAS framework does not cover the inner structure of agents to the level of detail required for the example application scenario, and is therefore rated low. The FIPA agent management specification is very comprehensive in this regards, and for this reason rated higher.

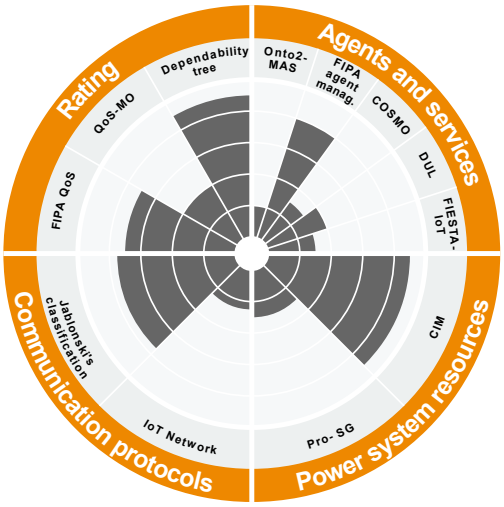

Fig. 5. Step 3: Ratings for each fragment are added to the graph.

All ontologies that have not been selected following this approach have been removed from Fig. 5, resulting in Fig. 6. The remaining fragments are combined by importing them to the corresponding domain ontologies. It is thereby without problems to use the same fragments for multiple domain ontologies. Ontologies created on this level are already biased by the developer, who considers certain fragments relevant or not. However, they can easily be reused by the developer himself for related applications or by other developers of the same field of expertise, who often share a similar mindset.

E. Model domains

Most of the concepts that are required in each domain are already provided by one of the imported fragments. Additional concepts, and in particular concepts that link nodes from the individual fragments, are added on the domain level following standard modeling techniques.

F. Select & combine domains

All of the identified domain ontologies are highly relevant for the example application scenario and therefore imported into a new, application-specific ontology. As described in Section *V-D Select & combine fragments*, no additional information is added to the ontologies when performing this step.

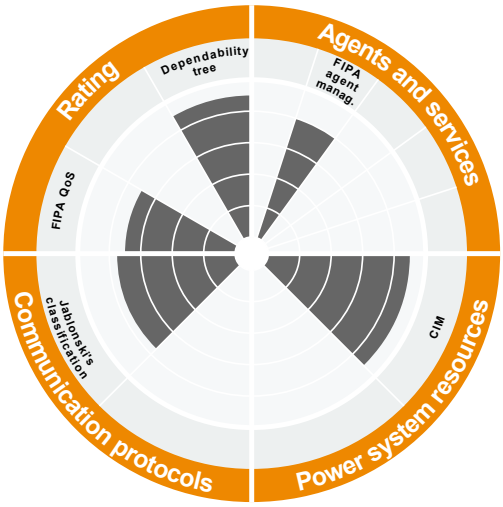

Fig. 6. Step 4: Unused fragments that are removed from the graph.

G. Model application-specific ontology

Information about interdependencies between the individual domains is added to the application-specific ontology. For example, it is now possible to link services to the power systems resources they control, specify the communication protocols and security features required to invoke the services, and rate each service w.r.t. its dependability features, s.a. availability and reliability. This completes the process and the application-specific ontology can, finally, be deployed.

VI. Conclusion and Future Work

The past has shown that creating ontologies that reflect all aspects of a certain sphere of interest to full extent is hard for simple scenarios and most likely impossible for complex ones. The focus should therefore be on creating simple and reusable ontologies which are then combined and extended to fully-fledged ontologies in multiple steps. The number of steps is not predetermined but a hierarchy of three types of ontologies has proven to be useful in practice.

The procedure presented in this work allows to create application-specific ontologies faster and easier by strongly encouraging the reuse of existing work, not only in the form of existing ontologies but also in other form including articles, reports, specifications, and books. Furthermore, each of the designed ontologies generated while following the procedure, except for the application-specific ontology, focuses on being reusable by other developers or in other projects. An obvious drawback of this approach is that the resulting application-specific ontology might contain more nodes and relations than actually required for the specific use case. This causes additional resource consumption for storing and processing.

There is a number of aspects which has not been fully covered in this work but are left open for future work. An important topic to be discussed is the reusability of the generated ontologies during the complete project life-cycle, from project initiation over planning and execution to project closure. In order to make this methodology applicable, a kind of design tool is considerable. Such a tool guides a developer through the process and provides him/her a "meta structure". It would further support the easy reuse of fragments (e.g., community driven library of fragments). And, finally, the reuse of available information and existing ontologies may be restricted by copyright and licensing issues [20].

References

[1] A. Gyrard, P. Patel, S. K. Datta, and M. I. Ali, "Semantic Web meets Internet of Things and Web of Things," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 917–920.

[2] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the Internet of Things," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 1793–1798.

[3] D. Allemang and J. Hendler, *Semantic Web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[4] N. F. Noy and D. L. Mcguinness, "Ontology development 101: A guide to creating your first ontology," Tech. Rep., 2001.

[5] A. De Nicola and M. Missikoff, "A lightweight methodology for rapid ontology engineering," *Communications of the ACM*, vol. 59, no. 3, pp. 79–86, 2016.

[6] A. L. A. Menolli, H. S. Pinto, S. S. Reinehr, and A. Malucelli, "An incremental and iterative process for ontology building," in *ONTOBRAS*, 2013, pp. 215–220.

[7] S. Mayer, J. Hodges, D. Yu, M. Kritzler, and F. Michahelles, "An open semantic framework for the Industrial Internet of Things," *IEEE Intelligent Systems*, vol. 32, no. 1, pp. 96–101, 2017.

[8] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications – Part I: Concepts, approaches, and technical challenges," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1743–1752, 2007.

[9] T. Frühwirth, A. Einfalt, K. Diwold, and W. Kastner, "A distributed multi-agent system for switching optimization in low-voltage power grids," in *Proceedings of the 22nd IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2017, pp. 1–8.

[10] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications—part ii: Technologies, standards, and tools for building multi-agent systems," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1753–1759, 2007.

[11] C. Donzelli, S. A. Kidanu, R. Chbeir, and Y. Cardinale, "Onto2MAS: An ontology-based framework for automatic multi-agent system generation," in *Signal-Image Technology & Internet-Based Systems (SITIS), 2016 12th International Conference on*. IEEE, 2016, pp. 381–388.

[12] FIPA TC Communication, "FIPA ACL Message Structure Specification," 2002.

[13] FIPA TC Agent Management, "FIPA Agent Management Specification," 2004.

[14] International Electrotechnical Commission (IEC), "61970-501 Energy management system application program interface (EMS-API) – Part 501: Common Information Model Resource Description Framework (CIM RDF) schema ," March 2006.

[15] S. Gillani, F. Laforest, and G. Picard, "A generic ontology for prosumer-oriented smart grid." in *EDBT/ICDT Workshops*, 2014, pp. 134–139.

[16] S. Jablonski, I. Petrov, C. Meiler, and U. Mayer, *Guide to web application and platform architectures*. Springer, 2004.

[17] FIPA TC Nomadic Application Support, "FIPA Quality of Service Ontology Specification," 2002.

[18] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, Jan 2004.

[19] G. F. Tondello and F. Siqueira, "The QoS-MO ontology for semantic QoS modeling," in *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 2336–2340.

[20] V. Rodríguez-Doncel, C. Santos, P. Casanovas, and A. Gómez-Pérez, "Legal aspects of linked data–the european framework," *Computer Law & Security Review*, vol. 32, no. 6, pp. 799–813, 2016.