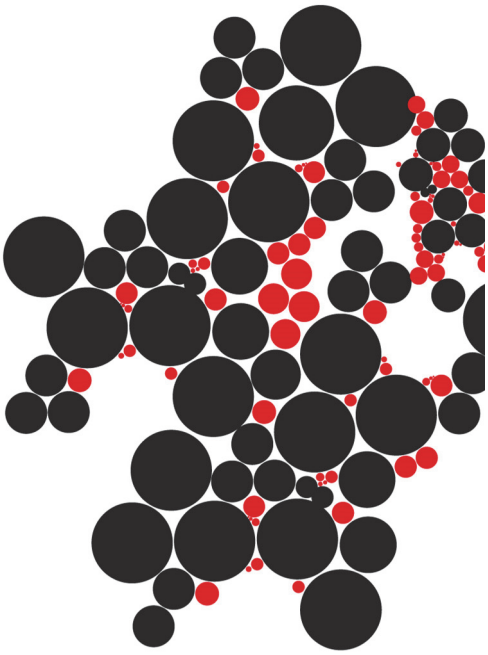
MASTER THESIS



THE VALUE OF ONTOLOGIES FOR DEVELOPING SEMANTIC STANDARDS

Jeffrey van den Brande

MSC BUSINESS INFORMATION TECHNOLOGY

EXAMINATION COMMITTEE

dr. ir. M.J. (Marten) van Sinderen

dr. ir. M.E. (Maria-Eugenia) Iacob

dr. ir. J.P.C. (Jack) Verhoosel (TNO)

UNIVERSITY OF TWENTE.

TNO innovation for life

17-06-2013

THE VALUE OF ONTOLOGIES FOR DEVELOPING SEMANTIC STANDARDS

MASTER THESIS

Enschede, 17 June 2013

AUTHOR

Jeffrey van den Brande
Master Business Information Technology
School of Management and Governance

GRADUATION COMMITTEE

dr. ir. M.J. (Marten) van Sinderen
University of Twente, Computer Science

dr. ir. M.E. (Maria-Eugenia) Iacob
University of Twente, School of Management and Governance

dr. ir. J.P.C. (Jack) Verhoosel
TNO, Connected Business

TNO innovation for life

UNIVERSITY OF TWENTE.

MANAGEMENT SUMMARY

A current trending topic in the information modeling discipline is ontologies. An ontology can be seen as something different from traditional information models. It is a formal, explicit specification of a shared conceptualization of a real-world domain. TNO developed a methodology to develop semantic standards using an information model, called MOSES. As in the literature currently no or few development methodologies exist that use an ontology instead of an information model for developing a semantic standard, one of the goals of this thesis is to develop such a development method, based on MOSES. By extending this methodology the benefits an ontology could bring can be examined as well. Therefore the main research question is formulated as: *How can the MOSES methodology be extended with the development and use of an ontology?*

For semantic standards interoperability is of great value. Interoperability refers here to the ease of exchange of information between domain stakeholders. The interoperability between stakeholders in a domain is expected to be able to be reinforced by making use of an ontology. In the literature specific aspects of ontologies were denoted to do this and also mindsets from other ontology development methods were found in the literature that reinforce interoperability. The most important aspects found are that an ontology provides a shared vocabulary with unambiguous concept descriptions for all stakeholders to improve interoperability. By means of validity rules an ontology can also rule out any configurations possible that do not square with the real-world domain and because an ontology can model processes and operation rules, dynamic domain behavior can be modeled as well.

The extension of the MOSES methodology with the development and use of an ontology means the information model that is developed is replaced with an ontology. MOSES is designed to explicitly capture all static and dynamic concepts of a domain. To be able to develop an ontology that at least covers these aspects, the Resource-Event-Agent (REA) upper ontology is used as ontological foundation for the ontologies to be developed in the method. This means all domain concepts have to be mapped on the concepts defined by REA. The main concepts of REA are resources, events, agents and commitments, where commitments correspond to agreements between two agents to exchange (the information of) one specific resource. The commitment is executed by at least one event sending the resource and at least one event receiving the resource.

Taking into account the interoperability benefits ontologies can bring, the MOSES methodology has been adapted to include the development and usage of an ontology. The exact methodology steps, mindsets and notations are documented well to facilitate practitioners to execute the development methodology. The first phase of MOSES is hardly changed, so in this phase the resources, events, agents and commitments of the domain still need to be identified in a MERODE model. To extend the MOSES methodology with an ontology, the business information modeling phase is replaced by a phase building an ontology base and another to further specify the ontology model. In the latter phase, the properties of the agents are determined and resources, but also their constraints. The latter is done by initially capturing the constraints in a semi-informal way to facilitate domain experts, followed by a formalization step. The final phase of MOSES is hardly altered; message structures can be generated from the ontology making use of an XML-translation.

To gain hands-on experience with the developed methodology, a case developing a microgrid with flexibility in energy demand and supply was performed. Using relevant literature sources and knowledge from domain experts, the extended methodology was applied. Performing the methodology resulted in a multipurpose ontology that not only can be used for deriving a semantic standard, but also can be used for other purposes. The case shows an example of an additional application where the congestion impositions on grid participants are shown in an overview. By having several domain experts and ontology development experts examining the case and its end products, the extended methodology was iteratively improved to suit the building of domain ontologies best.

PREFACE

This document contains my master thesis, the final document I produced for the master Business Information Technology at University of Twente. It contains the results of my research on a development methodology for semantic standards making use of an ontology, which I carried out at TNO. I sincerely hope that the results of this research contribute to the knowledge and practices within the company.

During the 6 months I worked on this project I encountered many challenges. Some were harder than others, but I learned how to cope with them as the project progressed. Especially in the start of the project the scope was not clear, which was also due to some lack of clarity and setbacks around arranging a real-life case to evaluate the methodology. After this uncertainty disappeared, a quick regain of focus and slight change of scope helped me to get up to speed.

Thanks to the excellent support of my TNO supervisor Jack Verhoosel this thesis has come to a good end. My university supervisors Marten van Sinderen and Maria-Eugenia Iacob also supported me well by posing critical questions at the right moments, which were highly valuable for my progress. I am very grateful for their help and support and therefore I want to express them my sincere gratitude.

My colleagues at TNO provided me, next to a nice atmosphere to work in, with a lot of inspiration and help with parts of my work. Therefore I would like to thank Dennis, Jasper, Michael, Linda, Matthijs and Istvan. In particular I would like to thank Ad Schrier for the interesting discussions, feedback and help on two core elements of my thesis; REA and MERODE. Without this help my work could not be as well-founded as it is now. Finally, I want to thank my parents for their support throughout my studies.

I hope you will enjoy reading this master thesis and can benefit from its content. If you have any questions, please feel free to contact me.

Jeffrey van den Brande

Enschede, June 2013

TABLE OF CONTENTS

1 INTRODUCTION

1.1 MOTIVATION AND BACKGROUND

The information modeling discipline is experiencing a shift from the use of information models towards the use of ontologies for describing and developing software solutions. The use of an ontology appears to provide several advantages over the use of a more traditional information model. At the same time there are still discussions going on in the information modeling domain about whether these advantages provide real added value. As a result of this development, TNO is interested in the added values the use of an ontology could bring when developing semantic standards.

Next to that, currently no development methodology for developing semantic standards using an ontology as means exists. As ontologies may provide benefits to the development process, developing a development methodology using an ontology as means, can show how these benefits can be utilized. Also, development methodologies for ontologies themselves are currently very diverse in their approaches, as each development methodology focuses on different properties of ontologies. A more general approach to ontology development should be determined to cover the development of ontologies for all business domains.

The following two subsections provide some more background information on the information modeling trend towards the use of ontologies and the diversity of development methods for ontologies and semantic standards, which both drive this research. Also, to test and validate the methodology that will be designed in this research, a case from the energy domain is attempted to be treated using this methodology. An introduction to this case is elaborated in the last subsection.

1.1.1 FROM INFORMATION MODEL TO ONTOLOGY

An information model can be seen as a traditional way of structuring definitions or meanings of things in the real world and specifying the relationships between these things in static semantics (Aßmann, Zschaler, & Wagner, 2006; Lee, 1999). An information modeling language is used to express this information model. On the other hand, ontologies represent a shared understanding of the important concepts in a domain by making explicit formal descriptions of concepts, instances and relations relevant to this domain (Kalfoglou, 2001; Nguyen, 2011). These are captured in ontology models, described by an ontology language and are shared between all domain stakeholders.

The discussion on the differences between information models and ontologies is still going on, but literature on ontologies points out that there are several differences and advantages of ontologies over (traditional) information models. For example, ontologies are expected to be able only to describe behavior, while information models can describe as well as prescribe behavior (Aßmann et al., 2006).

Also, the development of information models usually is focused on the creation of a (computer) system, whereas the aim of an ontology is to describe and create a shared understanding of the concepts of a domain (Aßmann et al., 2006). We suspect that in some cases the information models have some shortcomings to facilitate a perfect information exchange between actors in a domain. For example, Arango & Prieto-Diaz (1991) identify a need for a reusable infostructure that defines all aspects of a problem domain and its semantics to fill the gap between the kinds and forms of domain knowledge and the content and form of software assets for software construction. An advantage of an ontology over an information model postulated is that an ontology only consists of unambiguous definitions that are directly related to a set of relationships that hold among these definitions (Kalfoglou, 2001).

We suspect that next to these differences there are more differences and advantages of ontologies over information models. As TNO is interested in how the development of a domain ontology can add value to the interoperability in this domain, this research will focus on this type of benefits.

1.1.2 DEVELOPMENT METHODS FOR ONTOLOGIES AND SEMANTIC STANDARDS

While there are hardly any development methods for semantic standards available in the literature, the interest of TNO lies at the improvement of their current method for developing semantic standards: MOSES (Model gebaseerde ontwikkeling van semantische standaarden; model-based development of semantic standards). This method currently involves modeling techniques where traditional information models are highly involved. As a result of the belief ontologies might be the successors of information models that have more benefits, it is interesting to find out the possible benefits the use of an ontology could bring for the development of semantic standards.

Furthermore, there is currently no standard for the process of developing an ontology. Many development methodologies are available in the literature, such as Methontology and Enterprise Ontology (the most influential methodologies are reviewed in section 3.2). These methodologies all have the goal to create a description of a domain in an ontology, but each method focuses on different aspects important for ontology development. To develop a more general approach to ontology development for business domains, the most important aspects of each of these methodologies should be taken into account.

1.1.3 SMART GRIDS AND MICROGRIDS

In the energy domain a trend is going on to decentralize the energy supply and demand in an energy grid. To be able to involve all parties on the energy grid and to maintain a balance in energy demand and supply in the grid, more and new information needs to be exchanged between the involved parties. It is important that this information is specified unambiguously, so the different parties can easily integrate this information with their own systems. One solution for this information need could be the use of an ontology on this information exchange. Another is to use an information model. Therefore we want to get to know the differences between the two and why and how to use these models in this context.

At the moment there are many projects looking at how to achieve a guaranteed energy network balance by means of a smart grid, where all parties connected exchange information in order to minimize the imbalance between energy demand and supply on the energy network. TNO is collaborating with partners in the energy industry to enable the integration of a higher rate of distributed and renewable energy sources into the electricity grid by means of incorporating flexibility in electricity demand and supply to mitigate the energy supply uncertainty. By exchanging certain information between the involved parties, this flexibility can be achieved.

A specific variant of a smart grid is a microgrid, which is in fact a smart grid on local scale. A lot of information is exchanged between the involved parties. Think of, for example, microgrids that require the exchange of the information concerning energy streams in the grid. In most cases the information exchanged is based on (traditional) information models. For example, two of the commonly used frameworks in the energy sector, NIST (National Institute of Standards and Technology, 2012a) and OASIS Energy Interoperation (OASIS Open, 2012a), are both based on a traditional information model. How to facilitate interoperation best and to facilitate for the upcoming trend in renewable energy is an interesting question, for which this research will have an initial look at.

1.2 PROBLEM STATEMENT

Clearly there are differences between ontologies and information models. What these differences are and what the advantages of an ontology over information models are, is yet unclear. This thesis therefore includes a review of what benefits ontologies have, with a focus on interoperability benefits in a domain, as this is one of the important aspects of the methodology to be improved. Secondly, how and why ontologies could be used in a methodology for developing semantic standards is to be found out to ground the improvements to be made to the development method.

The established development methodology, once built, needs to be evaluated. By involving the microgrid problem of the energy domain, this method can be applied to test it. The added values of the development methodology can be determined by domain experts who can evaluate the end results of the method and compare with the results of their old implementation. Based on the evaluation, an improved methodology can be determined for MOSES involving the creation and use of an ontology.

1.3 RESEARCH QUESTIONS AND GOAL

The goal of this research is twofold. The first goal is to extend the MOSES methodology with the development and use of an ontology. This goal will be achieved by first studying the literature on ontologies, its differences and advantages for achieving interoperability. Then finding the best additions to MOSES for improving interoperability in a domain using an ontology and developing the methodology improvement.

The second goal is to treat interoperability issues in a microgrid (of the energy domain) to cope with the problems of the increasing uncertainty of energy supply and demand of renewable energy sources. This will be done by applying the improved MOSES methodology.

For achieving these goals, the following main research question has to be answered:

How can the MOSES methodology be extended with the development and use of an ontology?

To answer the main research question, a number of sub-questions have to be answered:

RQ1. What is the state-of-the-art on ontology development methodologies?
RQ2. What are the benefits of the development and use of an ontology for interoperability in a domain?
RQ3. What are good additions to the MOSES methodology for improving interoperability in a domain using ontologies?
RQ4. How can the extended methodology be applied and evaluated in the energy domain?

1.4 RESEARCH METHOD

To answer the research questions different methods can be applied to reach several outcomes. This research can be identified as a design-science research (Hevner, March, Park, & Ram, 2004), as one of the main goals is to produce a “viable artifact” in the form of a methodology (an ontology-development method). To design this model, the DSRM (design science research methodology) process model of Peffers, Tuunanen, Rothenberger, & Chatterjee (2007) (Figure 1) is followed. This process model allows to be started at different steps, depending on the initial approach to the design research (depicted as possible research entry points in Figure 1). This particular research will follow the nominal process sequence.

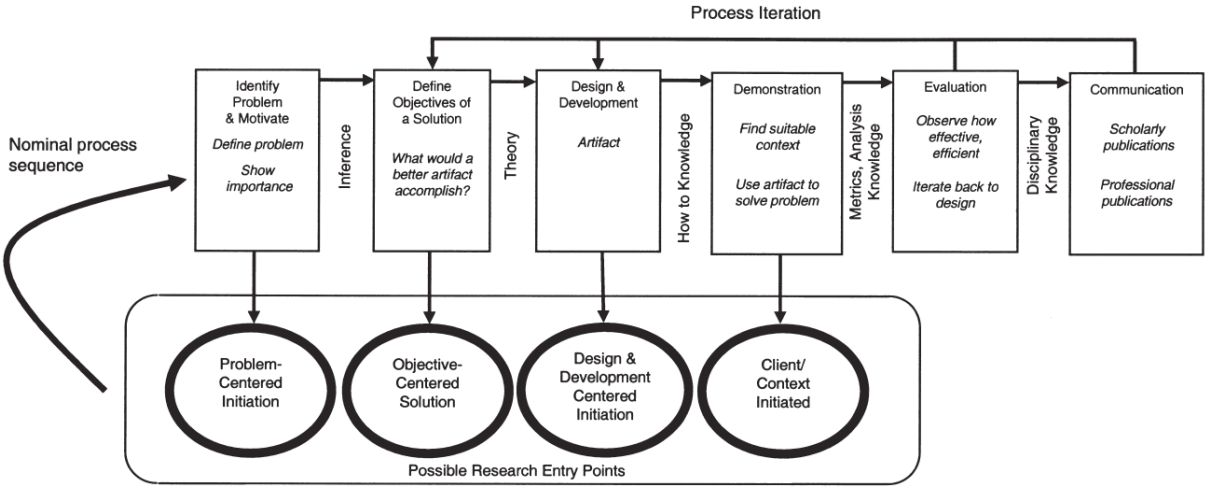


Figure 1: Design science research methodology process model (Peffers et al., 2007)

The design science research guidelines defined by Hevner et al. (2004) are aimed to support the design process towards an effective design artifact. Table 1 lists and describes each guideline and shows how these are satisfied by the methodology of this research.

Guideline	Description	Application in this research
1: Design as an artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.	An ontology development method will be designed.
2: Problem relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.	The problem will be first investigated and motivated in the following chapter to support the actual methodology design.
3: Design evaluation	The utility, quality and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.	The methodology will be applied on a case in the energy sector, which will be evaluated by using expert interviews.
4: Research contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.	This research will contribute a development method for semantic standards in a domain using ontologies.
5: Research rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.	Existing ontology development methods will be used as base for our design methodology. Methods for design evaluation by expert interviews will also be used.
6: Design as a search process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.	The design process will involve design iterations.
7: Communication of research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.	

Table 1: Design-science research guidelines (Hevner et al., 2004) and how they are satisfied in this research

1.5 DOCUMENT STRUCTURE

The main structure and concrete approach of this thesis is shown in Table 2. This table also maps the research questions, research methodologies and design-science guidelines to each chapter and describes briefly the outcomes of each chapter.

In chapter 2 a literature study is performed on ontologies, its relationship with information models, interoperability and ontology development methods. Chapter 3 performs another literature study, which has the goal to describe the aspects of ontologies in which it could improve the interoperability in a domain compared with information models. These first two chapters provide the required foundations for developing an improved methodology on developing semantic standards making use of ontologies.

The improved development methodology will be designed in chapter 4. This chapter is structured in first explaining the steps, then the involved mindsets, followed by the notations to be used. The established development methodology will then be applied on the microgrid case in chapter 5. The chapter contains a description of the energy domain and an elaboration on the microgrid problem. The chapter ends with an evaluation of the practices experienced with the methodology and the end result of applying the methodology. The final conclusions of this research are drawn in chapter 6.

Chapter	Design-science guidelines	Research questions	Methodology	Outcome
2: State-of-the-art	2, 5	RQ1	Literature study	A description of the current state-of-the-art on ontologies, its relationship with information models, interoperability and ontology development methods.
3: Interoperability benefits of the use of an ontology	2	RQ2	Literature study	A description of the aspects in which the use of an ontology improves interoperability in a domain compared with information models.
4: Development method for ontologies fostering interoperability	1, 4, 5, 6	RQ3	Model design	The concrete methodology for ontology development fostering interoperability. This includes an indication on the steps, mindset and notation to be used. Also, an elaborate description and reasoning on what are good additions to the original MOSES methodology is given.
5: An ontology for the energy domain	3	RQ4	Model evaluation	An overview of the energy sector, a description of the design process and the actual design of an ontology fostering interoperability in the energy domain. Also, the benefits of this ontology for the energy sector are discussed and the ontology (development method) will be evaluated using expert interviews.
6: conclusions	7			The final conclusions that can be extracted from this research.

Table 2: Research outline

2 STATE-OF-THE-ART

In this section the necessary theoretical background of the research will be provided. First, information models and its theoretical background is introduced. This is followed by a study on ontologies, the philosophy behind them, available design methods and languages and their uses. The chapter ends with an overview of literature about interoperability and a description of the original MOSES methodology.

2.1 INFORMATION MODELS

Information models have been introduced in order to provide a definition of meanings and interrelationships of data or information (Lee, 1999). They provide a means for sharing, integrating and managing these data. An information model can be seen as a representation of concepts, relationships, constraints, rules and operations to specify data semantics for a given domain. Stahl & Voelter (2006) define a model as "an abstract representation of a system's structure, function or behavior".

One of the main approaches to develop an information model is through the Model Driven Software Development (MDSD) discipline (Stahl & Voelter, 2006). Central in this discipline is that every information model should be an instance of elements of a model on a higher abstraction level. The way the information model is structured and which elements are allowed, is established by a metamodel. In the MDSD discipline a metamodel is defined as describing the possible structure of models, in other words, a modeling language. To maintain a certain uniformity in the modeling languages, a metametamodel can be defined describing the concepts available for metamodeling. These relationships are shown in the 4-metalevel hierarchy of OMG, as shown in Figure 2. Next to that, Stahl & Voelter (2006) note that the elements of an information model should be based on elements that reside in the domain where the information model will be applied.

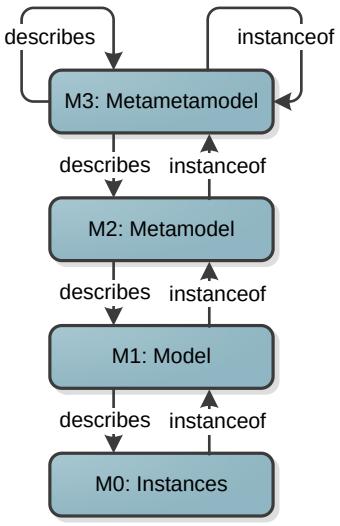


Figure 2: The four metalevels of OMG

Information models that respect a metamodel are also called MDA (Model Driven Architecture) models. Because the metamodel provides for the meaning of the model, it can be stated that MDA models include semantics (that are defined by the metamodel) (Stahl & Voelter, 2006). Additionally, Stahl & Voelter (2006) note that for information models to constitute a correct subset of the domain, metamodels need to ignore unnecessary and unwanted properties of elements in the domain. The best way to achieve this is by using a constraint language on the metamodel. Therefore it can be deducted that an information model on itself does not provide enough means to represent a domain of discourse. Additionally, Shanks, Tansley, & Weber (2003) claim that a good way of evaluating and validating a conceptual information model is to develop an ontology with domain experts that can be used as reference for the evaluation and validation process.

Many information modeling languages exist. The Unified Modeling Language (UML) is one of the most used languages for specifying information models by means of a class diagram (Object Management Group, 2012). A class diagram depicts the static structure of the elements of a system or structure. Another widely used modeling language is the entity-relationship (ER) modeling grammar (Burton-Jones & Weber, 1999). Central to this model type is the modeling of relationships between concepts and giving attributes to these relationships.

2.2 ONTOLOGIES

The idea of bringing explicit domain knowledge into the design of software models originated from artificial intelligence (AI) research (Kalfoglou, 2001). AI researchers developed knowledge engineering methods that proved to be powerful tools for transforming knowledge into machine-readable form to enable automated reasoning about the domain of interest. Ontologies can be used to represent such a form of domain knowledge.

An ontology can be seen as something different than information models, while there are still some aspects of ontologies that overlap with aspects of information models. Noy & McGuinness (2001) define that an ontology, like a (traditional) information model, also contains explicit formal descriptions of concepts in a domain. Nguyen (2011) also notes that an ontology, like information models, specifies concepts, relations and instances relevant to a domain.

This conceptualizing property of an ontology (in information systems) is depicted in Figure 3 of Hesse (2008). In this figure is shown that an ontology can be used as a referent to the (real world) domain. Here, an ontology is used to provide a conceptual model for identifying and understanding prerequisites, conditions and constraints for real world domains. An ontology is related to representation and conception, because information systems representations (e.g. models) are used by actors to refer to objects (referents) of a domain. Guizzardi (2007) clarifies this relation of ontologies even further using Figure 4. The philosophy behind this relationship model is that a model faithfully represents an abstraction by using the language primitives provided by a modeling language. Both the abstraction itself and the modeling language rely on conceptualizations. These are immaterial entities only existing in the mind of the user or a community of users of a language.

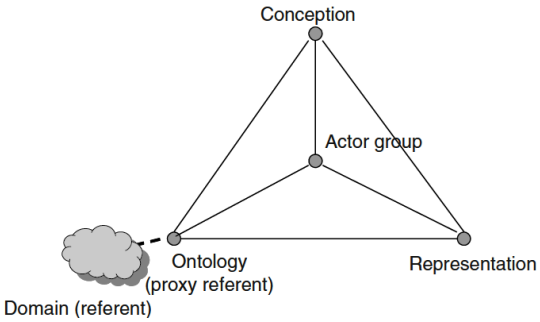


Figure 3: Semiotic tetrahedron of Hesse (2008)

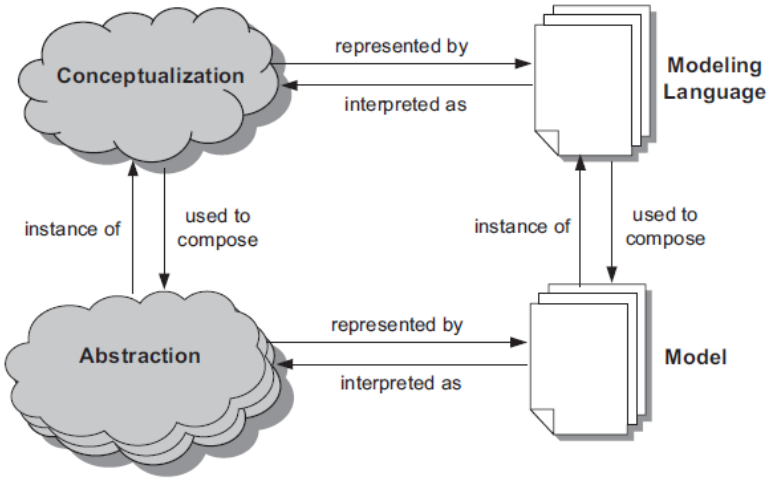


Figure 4: Relations between conceptualization, abstraction, modeling language and model (G Guizzardi, 2007)

The components an ontology should comprise of are classes, relations, formal axioms and instances (O Corcho, Fernandez-Lopez, & Gomez-Perez, 2007). Classes represent concepts or sets of instances, organized in taxonomies through which inheritance mechanisms can be applied. Relations are a type of association between concepts of the domain. Binary relationships can also be used to express attributes of a concept. Formal axioms

define the assertions that have to be made to ensure the consistency of an ontology. Instances represent individual things, which are an instance of a specific class.

Ontologies have four main application scenarios for ICT systems (Michael Uschold & Gruninger, 2004): (1) *neural authoring*, where a company develops its own neutral ontology for authoring, and then develops translators of this ontology to other terminologies of other systems that the company is collaborating with. This results in a high reuse of knowledge. (2) *Common access to information* introduces one ontology that is used as neutral interchange format facilitating the translations necessary between the different information formats of the used legacy software systems. By using one neutral interchange format, no translators are required between each system, but only between the systems and the ontology. In (3) *ontology-based specification* an ontology is used as foundation for the development of software systems, which will allow for high interoperability among these systems as the information exchanged is based on the same ontology. (4) *Ontology-based search* applies ontologies as a structuring device for information repositories; it can classify information at a high abstraction level. If mappings between ontologies of information repositories can be made, a search query could even retrieve answers from all linked repositories.

In section 2.1 the 4- metalevel hierarchy of OMG is explained (see also Figure 2). Originally, this hierarchy is only used for model driven engineering activities to develop information models based on metamodels on a higher abstraction level (Bezivin & Gerbe, 2001). It is argued by Aßmann et al., (2006) and Henderson-Sellers (2011) that ontologies can also be mapped to this hierarchy. They argue that ontologies can be classified in two broad areas: upper level ontologies and domain ontologies. A domain ontology comprises of a hierarchy of terms in a specific domain. This property has a high correspondence with a traditional model on level M1 of the OMG meta-levels hierarchy (Giancarlo Guizzardi, 2005). An upper level, or foundational, ontology defines the representation of an ontology. This can be seen analogous to metamodels in OMG's hierarchical level M2. An upper-level ontology can, for example, be an ontology representing language, like the Unified Foundational Ontology (UFO) of Guizzardi (2005), which in short, defines the foundational concepts for forming an ontology. Figure 5 visualizes the reasoning of Aßmann et al. (2006) and Henderson-Sellers (2011).

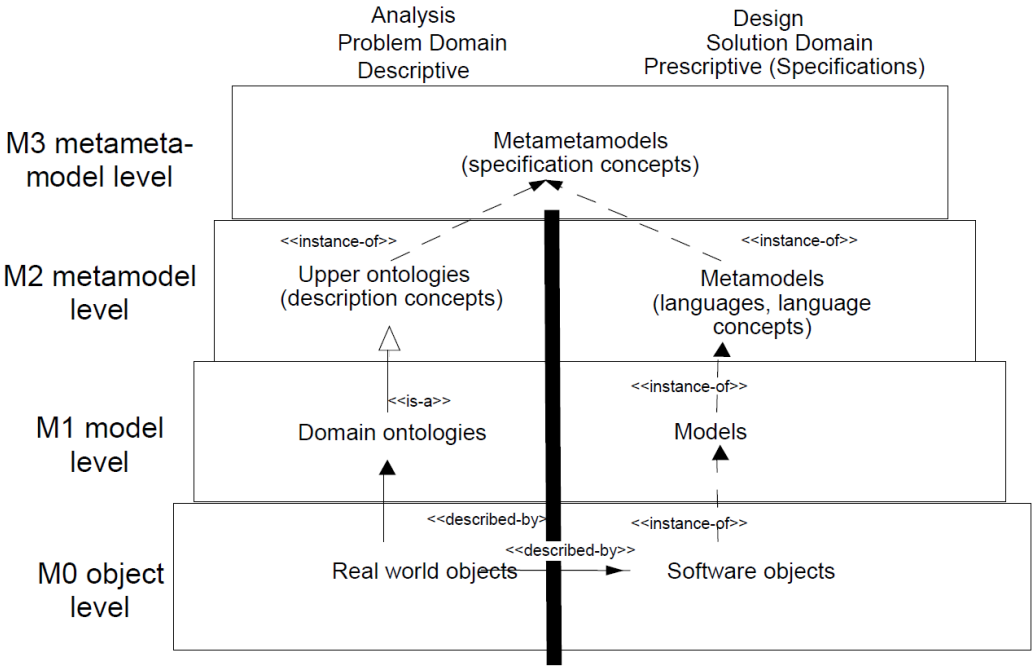


Figure 5: The ontology-aware meta-pyramid (Aßmann et al., 2006)

Henderson-Sellers (2011) identifies additionally that there is a relation between metamodels and upper ontologies and domain ontologies. These are related to each other in a so-called powertype construct. This

construct allows both instance-of and generalization relationships between levels. Upper ontologies are claimed to be classified by metamodels and domain ontologies are claimed to be instances of metamodels, while they are also a subtype of an upper ontology.

Ontologies consist of constructs that collectively impose a structure on the domain being represented, which constrain the interpretations possible of the terms involved (Kalfoglou, 2001). These constructs are often comprised by definitions of terms in a hierarchy lattice that are directly related to a set of relationships that hold among these definitions. This relates closely to the property of an ontology Aßmann et al. (2006) identified. They state that, intuitively, anything that is not explicitly expressed by an ontology is unknown. This is called the open-world assumption.

2.2.1 ONTOLOGY LANGUAGES

The actual encoding of a formal ontology is done in a specific ontology language. Over the years many different ontology languages have been developed with different aims. For developing an ontology ourselves, we have to know which ontology languages there are and what purpose they are optimized for.

Maniraj & Sivakumar (2010) reviewed the major ontology languages developed: CycL, KIF, Gellish, DOGMA, IDEF5 and OWL. They also identified 3 main categories for ontology languages: (1) *logical languages* entail first order predicate logic, rule based logic and description logic. (2) *Frame based languages* can be compared to relational databases and (3) *graph based languages* are aimed at building a semantic network.

The CycL ontology language (Cycorp Language) can represent knowledge from a knowledge base. The main concept of this language is to group constants in generalization/specialization hierarchies, stating general rules supporting inference about the concepts and naming constants used to refer to information for represented concepts. Also, the knowledge base is divided into microtheories, which are concepts and facts touching one particular realm of knowledge.

KIF, or Knowledge Interchange Format, has declarative semantics. This means expressions in the representation does not require an interpreter to understand or manipulate them. Also, it represents nonmonotonic reasoning rules and definitions of objects, functions and relations. The language can contain sets, sequences, numbers and arithmetics and relations.

Gellish is an open industry standard for defining data models, data language and a knowledge base with a taxonomy of concepts and a grammar for data exchange messages, supporting data storage and communication.

One of the most used ontology languages is the Web Ontology Language (OWL). It is a language for making ontological statements and is intended to be used over the World Wide Web. It includes both a syntax for describing and exchanging ontologies as well as formally defined semantics. The data described by an OWL ontology can be interpreted as sets of “individuals” and “property assertions”. The latter relates individuals to each other. Next to that, constraints can be defined using axioms. These constraints mainly involve sets of individuals and relations between them. The axioms are also used for providing semantics by allowing systems to infer additional information to the data explicitly provided.

Next to the languages reviewed by Maniraj & Sivakumar, the Semantic Application Design Language (SADL) is an interesting ontology language to look at (Crapo, Wang, Lizzi, & Larson, 2009). It has the purpose of making semantic modeling accessible to domain experts. It provides an authoring environment for building rich formal models to which domain-specific rules can be added. It is built upon the OWL and also uses rules expressed in SWRL (Semantic Web Rule Language) or Jena. In fact, this language could additionally be interesting as it was originally developed for improving the performance of smart grids.

Even though the main goal of the anticipated ontology that will be used in the methodology is not to be used over the World Wide Web, but in a dedicated network of the domain stakeholders, OWL will be used as the specification language of the ontologies supporting the method. One of the reasons of this choice is that OWL includes native support for assertions on individuals and properties of classes. Also, the ability of defining and reasoning with a set or subset of class types can be of benefit for domain modeling.

2.2.2 ONTOLOGY EDITORS

Oscar Corcho, Fernández-López, & Gómez-Pérez (2003) performed an extensive evaluation of the ontology editors available. Editors as DUET, OILEd, Onto Edit Professional, Ontolingua, Protégé, WebODE and WebOnto were considered as the most important. These were evaluated on their support for interoperability, general issues, usability and more.

For this project it was chosen to use Protégé. This ontology editor supports the OWL, SWRL and RDF languages well and is widely used by other ontology developers as well. It also supports the creation and execution of constraints and has the ability to merge ontologies via plugins (Oscar Corcho et al., 2003). Next to that, Noy & McGuinness (2001) provide an elaborated paper on how to use the Protégé editor in the correct manner.

2.3 INTEROPERABILITY

The quality and ease of information exchange activities between actors involves the interoperability of these actors. While this relation is clear, there are many different interpretations of interoperability. The study of Kosanke (2006) found up to 22 different definitions of interoperability. One of the most cited definition of interoperability is *"ability for two (or more) systems or components to exchange information and to use the information that has been exchanged"* (IEEE, 1990). Chen and Daclin (2006) extend this definition with the concept of exchange of functionality to *"the ability to (1) communicate and exchange information; (2) use the information exchanged; (3) access to functionality of a third system".*

Chen and Daclin (2006) identified three main concepts relating to (enterprise) interoperability: *interoperability barriers, concerns and approaches*. Interoperability barriers can be seen as fundamental concepts for interoperability, as most other interoperability issues are application domain specific. The interoperability barriers can be of *conceptual*, *technological* or *organizational* nature. Conceptual nature involves the syntactic and semantic differences in the information to be exchanged. Incompatibility of information technologies (e.g. IT infrastructures or platforms) is the technological barrier and the organizational nature relates to the definition of responsibility and authority that determine the conditions the conditions of the interoperations. To remove these barriers, an approach needs to be taken. This approach can use a common format for all models (integrated), use a common format on meta-level that allows mapping to a specific system (unified) or use no common format and interoperability is achieved "on the fly".

Interoperability concerns exist on four viewpoints: the interoperability of data, services, processes and business (D Chen & Daclin, 2006). The first refers to collaboration using different data models and query languages. Services interoperability concerns services or applications to be able to function together. This can be achieved by solving syntactic and semantic differences and finding connections to the varied databases. The interoperability of processes involves the study on how processes, internal or external, are connected. How the business functioning of interoperating partners (e.g. decision making or legislation) are understood and shared without ambiguity is of concern for business interoperability.

2.3.1 MEASURING INTEROPERABILITY

To return to the main research question of this thesis, on which factors affect interoperability in a domain, interoperability needs to be quantified. Ford, Colombi, Graham, & Jacques (2007) give an overview of the available system interoperability metric frameworks. Many interoperability metrics involve maturity models on a qualitative basis. Still few quantitative interoperability measures exist. Therefore our interoperability measures will have to rely on the few existing interoperability quantitative measures.

One of the few frameworks quantifying interoperability in concrete measures is of Chen, Vallespir and Daclin (2008). They expressed interoperability into measures of three categories: interoperability potentiality, compatibility and performance. Interoperability potentiality can be measured on each *interoperability concern* (i.e. data, services, processes and business interoperability) on five *levels* (David Chen et al., 2008): "(1) isolated: total incapacity to interoperate; (2) initial: interoperability requires strong efforts that affect the partnership; (3) executable: interoperability is possible but the risk of encountering problems is high; (4) connectable: interoperability is easy even if problems can appear for distant partnership; (5) interoperable: which considers the evolution of levels of interoperability in the enterprise, and where the risk of meeting problems is weak".

To measure interoperability potential, Daclin, Chen, & Vallespir (2006) state four properties of a system that are listed below. These properties represent the potentiality of a system to adapt in particularly in a federated environment. If a property is associated with a score of 1, it reinforces interoperability potential. The actual measures are straightforward, but are never explained by the authors. To avoid possible ambiguity of the measures, the interpretation used in this research for each property is briefly explained below.

- Open (1) vs. closed (0): an open system contains components that are allowed to be modified or upgraded, while this is not possible in a closed system.
- Decoupled (1) vs. coupled (0): components of a decoupled system can remain unaware of other components in the same system. In a coupled system components are aware of each other.
- Decentralized (1) vs. centralized (0): when a system is centralized, there is one central component with a specific goal that interacts with the rest of the system upon this, whereas a decentralized system there can be multiple of such components.
- Configurable (1) vs. not-configurable (0): attributes and other properties of a configurable system can be easily configured, while this is not the case with a not-configurable system.

2.4 MOSES

MOSES (Model gebaseerde ontwikkeling van semantische standaarden; model-based development of semantic standards) is a model development methodology designed by TNO specifically aimed at the development of semantic standards (Schrier, Van Bekkum, Krukkert, Verhoosel, & Roes, 2012). It consists of 2 parts: (1) iteratively design both the GDM (Gedeeld Bedrijfs-Domein Model; shared business domain model) and GIM (Gedeeld Bedrijfs-Informatie Model; shared business information model), then (2) design the GOM (Gedeeld Oplossingsmodel; shared solution model), followed by the actual implementation. The methodology takes an iterative approach, where each next step provides feedback for the step before (see also Figure 6).

The thought behind the development methodology is that parties that collaborate in one domain share a specific view on reality (i.e. ontology). This method therefore first identifies and explores the concepts present and the events that are happening in this domain. Only after this is clear, a true information model can be determined.

The way of notation for this method, supporting the mindset and method, therefore makes use of an Actor-Object-Event Table (AOE-Table) where all relevant actors and objects identified are connected with events. The

events in this table are not only interaction events, but also the events that create and remove the objects in the domain model. The relevant actors and objects are identified in a table that shows a short description and the demands and requests of each actor/object. Then they are presented in a UML class diagram, which makes clear their relationships. To make clear the sequence of events happening, Jackson Sequence Order Diagrams are drawn.

When the business information model is developed in the next step, the domain model is elaborated by creating elaborate descriptions of each event. Also, the Jackson Sequence Order Diagrams are extended by connecting actors and objects to each event. Objects themselves are also elaborated upon by adding attributes to them and extending the UML class diagram with object constraints for each object.

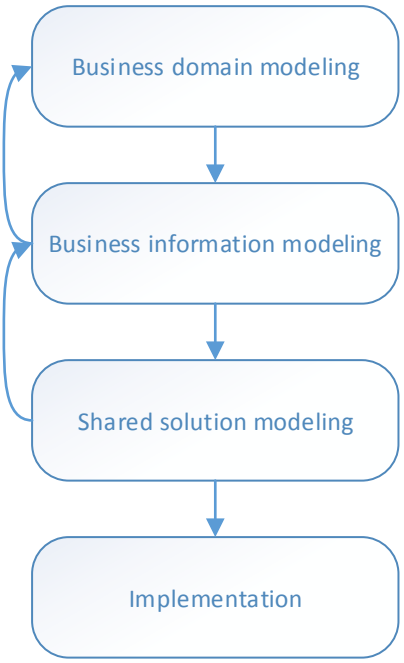


Figure 6: MOSES methodology steps

The third step, to develop a shared solution model, is achieved by transforming the business information model in structured messages facilitating interoperability between each actor. Depending on the technology chosen for the implementation (e.g. XML), (automatic) transformations can be performed from the business information model to standard message structures and dialog specifications. Eventually existing standards could be considered as reference for these exact implementations.

2.4.1 MERODE

The philosophy behind MOSES is largely based on the MERODE modeling approach (Snoeck, Michiels, & Dedene, 2003). This approach supports the development of one, consistent, model of formal semantics. The three main components of MERODE are object types, event types and participations, as shown in Figure 7. Objects represent all entities in the domain model to be described. Events represent all business events invoking objects. Participations relate objects to the events they are invoked with. Multiple objects can relate to each other by existence dependency relationships, meaning the existence of a given object depends on the existence of the other object it has an existence dependency relationship with (Snoeck et al., 2003). Table 3 gives an overview of and describes the most important concepts of MERODE.

Figure 7: Main components of MERODE expressed in UML

MERODE concept	Definition
Event	Corresponds to something happening in the real world. It occurs at one point in time and has no duration modeled. An event is considered to be atomic, which means it cannot be split into several sub-sevents.
Participation	Corresponds to the relationship an object has with an event. An object can be related to an event as “owner” or “acquirer”. If an object owns an event, it is the most dependent object on this event. If an object acquired an event, it participates in this event because of the propagation of this event to related objects.
Object	Corresponds to a real-world concept. It can be described by a number of properties.

Table 3: Definitions of the MERODE concepts

MERODE focuses on the semi-automatic verification of internal correctness of specifications by facilitating "consistency by construction". This means the software tool for building a model guarantees semantic consistency by applying rules during the development of the model. The MERODE model consists of three subviews: an existence dependency graph (EDG) organizing object types according to existence dependency and inheritance, an object-event table (OET) identifying event types and relates those to object types and a behavioral model where finite state machines (FSMs) show the states of each object and the transitions between the states.

MOSES adopts a large part of MERODE: the EDG and the OET. It even extends the OET with distinguishing objects from actors, leading to an AOET (actor-object-event table). The EDG is expressed in a UML class diagram in MOSES. Next to this, the consistency rules of MERODE are adopted (Snoeck, Dedene, Verhelst, & Depuydt, 1999; Snoeck et al., 2003). The following consistency rules are defined by MERODE relating the EDG and the (A)OET:

- Alphabet rule: each event can have only one effect on objects of a class: it either creates, modifies or deletes objects. Also, each object class requires at least one event to create and another one to destroy a class.
- Propagation rule: when a class is dependent on a master class, the dependent class is automatically involved in the event types the master class is involved in.
- Type of involvement rule: the creation, modification or ending event of a dependent class is automatically an event type for the master class.
- Inheritance rule: an object type inherits all event types from its parent object type, either unchanged or specialized.
- Default life cycle rule: objects must include at least 2 events: its first need to be an object creation event, its last needs to be an object ending event.
- Restriction rule: existence dependent object types must have a more deterministic life cycle definition than their master object type.
- Contract rule: when two or more object types participate in the same event, a common existence dependent object (contract) is required that participates in this event.

2.5 FOUNDATIONAL ONTOLOGIES FACILITATING BUSINESS DOMAINS

The method of MOSES (and MERODE) results in a UML class diagram with an OET to model both the static part of a domain (actors and objects and their relationships) and the dynamic part (events occurring by actors and involving objects). The MOSES methodology then continues by elaborating on the actors, objects and events by adding all required properties and restrictions to result in a satisfying domain model from which a technology-specific semantic standard can be derived for the domain. To come back to the main goal of this research, which is the extension of the MOSES methodology making use of an ontology instead of an information model, an alternative for the use of a UML class diagram in the current method should be found.

To develop an ontology that is well-founded comparable to MOSES, an upper ontology can be used that facilitates the expression of both the static and dynamic part of business domains. Only a few of these upper ontologies exist, from which e³value, Resource-Event-Agent (REA), Unified Foundational Ontology (UFO) and Business Model Ontology (BMO) are most common. The following subsections will elaborate on the e³value, UFO and REA alternatives. BMO will not be considered, because it focuses on the position and economic ties of one central actor, while we want to depict the whole business domain (Schuster & Motal, 2009).

2.5.1 THE ONTOLOGICAL FOUNDATION OF REA ENTERPRISE INFORMATION SYSTEMS

The focus of the *ontological foundation of REA enterprise information systems* lies at resources, events and agents (REA) (Geerts & Mccarthy, 2000). These correspond to objects and events of the MOSES/MERODE methodology (Figure 14 shows a mapping of the concepts). The REA ontological foundation has the goal to specify the economic rationale behind business collaborations (Schuster & Motal, 2009). Its origins can be traced back to business accounting where business transactions were recorded with a technique called double-entry bookkeeping. Currently REA is used as ontological framework for the ISO Open-edi specification and is part of the work of the United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT), which is an international e-business standardization body. In Figure 8 a UML representation is constructed based on Geerts & Mccarthy (2000) and Gailly & Poels (2007) in an attempt to gain a clear overview on the constructs of REA.

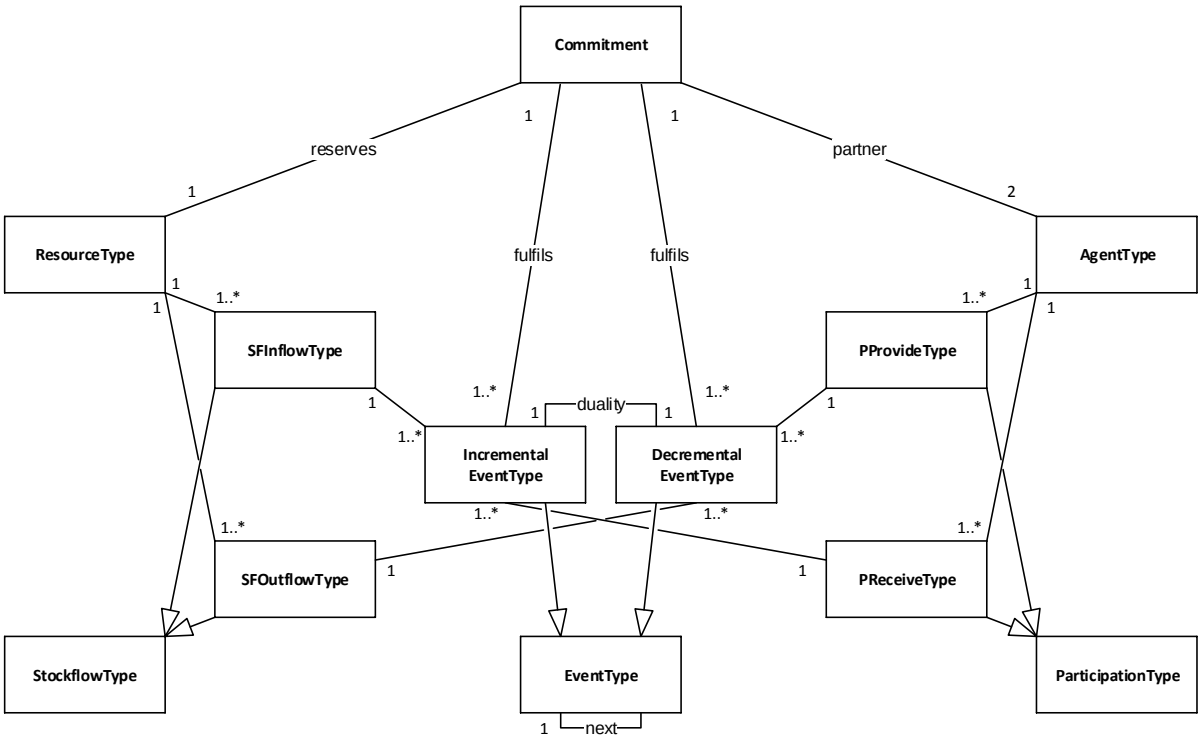


Figure 8: UML representation of REA (based on Geerts & Mccarthy (2000) and Gailly & Poels (2007))

In REA every business transaction is recorded as a double entry (a credit and a debit entry) (Andersson et al., 2006). An (economic) event represents an exchange of (economic) resources between two (economic) actors. To get a resource, an agent has to give up another resource. This combination of two events is called a *duality*. Events often occur as consequences of existing obligations of an actor, i.e. they "fulfil" the commitments agents bound themselves to. These obligations are therefore called *commitments*. To clarify the concept of commitment, Geerts & Mccarthy (2000) defined it as an "agreement to *execute* an economic event in a well-defined future that will result in either an increase of resources or a decrease of resources". By its definition, in REA, a commitment always exists between exactly two agents and one resource type. The *partner* relationship between a commitment and two agents and the *reserves* relationship between a commitment and a resource represent these ties.

The inflow or outflow of a resource, related to an event is depicted by a *stock-flow* relationship of an event with a resource. This relationship also has a certain kind of duality; it can describe the using, consuming, giving, taking or producing of a resource, which is either an inflow or outflow of a resource. An event with an inflow stock-flow should therefore have a duality with an event with an outflow stock-flow.

REA also comprises of agents, which have certain commitments in the business domain. This relationship is called *partnership*. The events that these commitments *fulfil*, the agent is related to via *participations*. Here the duality principle is also recurs in the form of participations in the form of *provision* or *receiving* participations. Because of the dual nature of the REA ontology, every event has exactly one receiving agent and exactly one providing agent.

To ensure the internal correctness of a REA model, REA comes with several axioms. These axioms are in agreement with the consistency rules defined in the MERODE methodology used by MOSES. The enumeration below shows the REA axioms (adapted from Geerts & Mccarthy (2000)).

Axiom 1. At least one inflow event and one outflow event exist for each economic resource; conversely inflow and outflow events must affect identifiable resources.

Axiom 2. All events effecting an outflow must be eventually paired in duality relationships with events effecting an inflow and vice-versa.

Axiom 3. Each exchange needs an instance of both the inside and outside subsets.

2.5.2 E³VALUE

A second upper ontology alternative is e³value of Gordijn & Akkermans (2003). Its original aim is to provide a means to analyze the profitability of business cases by identifying all value exchanges in a business case (Andersson et al., 2006). This upper ontology therefore provides a good means to depict all exchanges of value objects in a domain as well. Its notation is relatively simple. Figure 10 shows the e³value notation and Figure 9 shows a simple example from Schuster & Motal (2009). The example shows two actors (Buyer (A) and Seller) having two value exchanges (money is transferred from the Buyer to the Seller and goods are transferred from the Seller to the Buyer; B and C). These two value exchanges are an inflow and outflow of value for the actors, indicated by value ports (D). They are part of one value interface (E), meaning the two exchanges belong to one combination of exchanges. The sequence of value exchange events is depicted by a start stimulus (G), stop stimulus (H) and the path in between (F). In the example, the Buyer initiates the value exchange, after which it stops at the Seller.

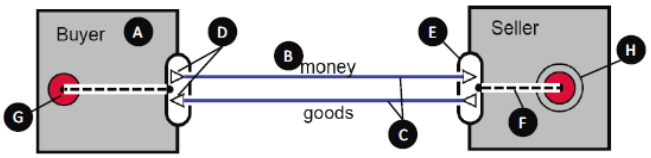


Figure 9: Simple example of an e³value exchange (Schuster & Motal, 2009)

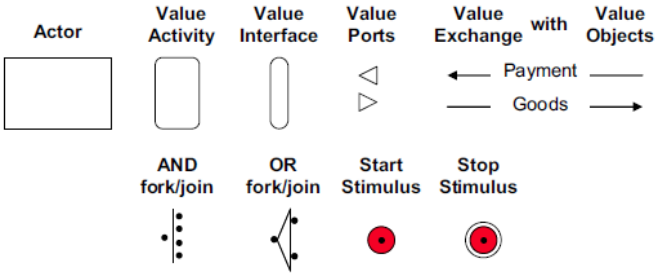


Figure 10: e³value notation (Schuster & Motal, 2009)

For gaining more insights into the e³value ontology and for comparison purposes, the ontology expressed in UML is restated in Figure 11 below from the work of Gordijn & Akkermans (2003). This diagram clearly shows how the model components relate to each other. An important aspect of e³value is that every value exchange connects to exactly 2 value ports (one inflow and one outflow). This construct therefore implies economic reciprocity (Andersson et al., 2006). Next to this, e³value allows the specification of the value height of a value object. When a full business case is modeled using e³value and all value heights of value objects are specified, the business profitability of the business case can be calculated (Gordijn, Osterwalder, & Pigneur, 2005).

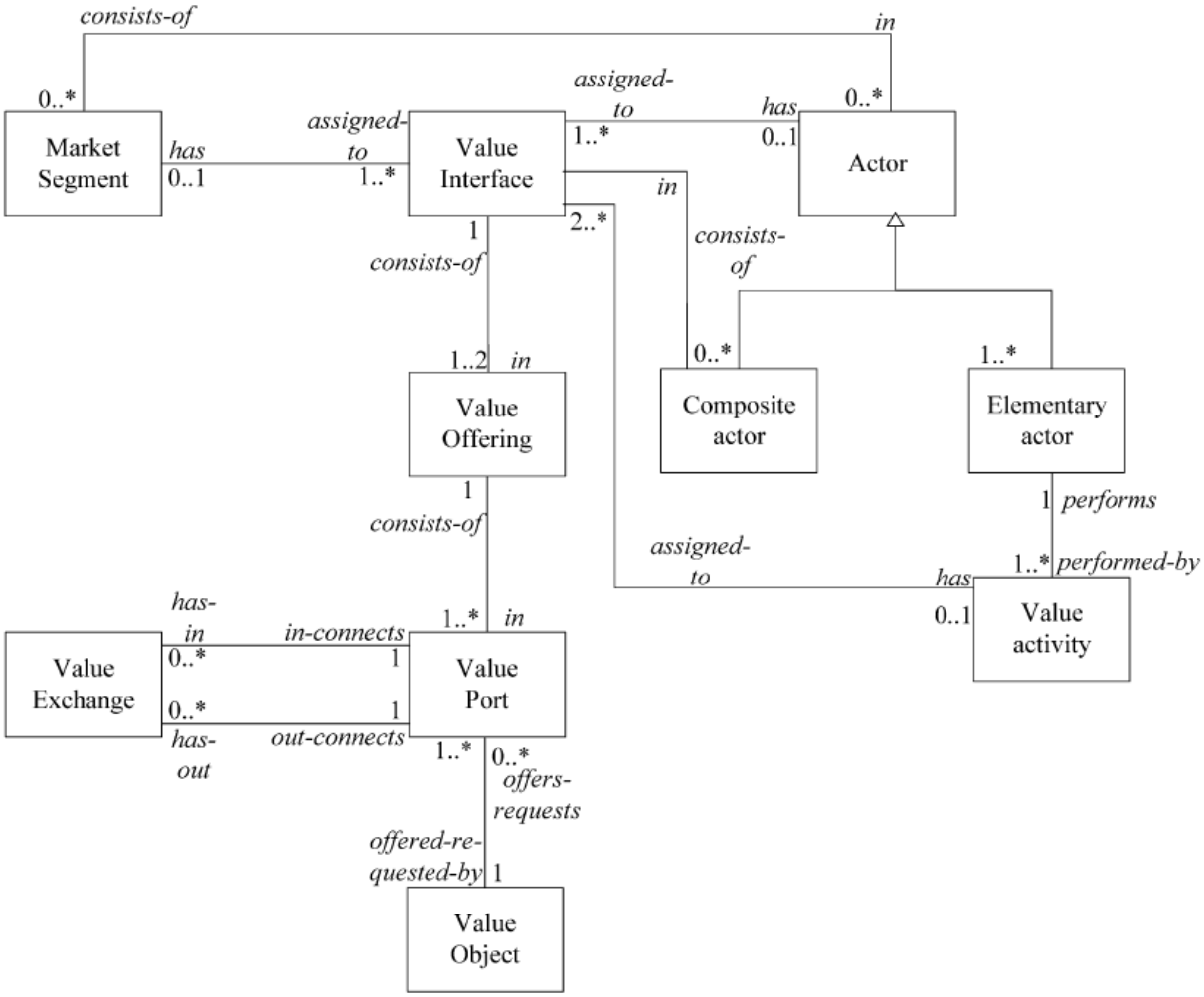


Figure 11: UML representation of e³value (Gordijn & Akkermans, 2003)

2.5.3 UNIFIED FOUNDATIONAL ONTOLOGY

Part of the Unified Foundational Ontology (UFO) of Guizzardi (2005) can also be qualified as alternative. UFO-C is the part of the UFO that focuses on business process modeling (Giancarlo Guizzardi & Wagner, 2005). It includes concepts like (physical) agent, (non-agentive object and (action) event, which correspond with the elements actor, object and event respectively of MERODE/MOSES. Figure 12 shows a UML representation of UFO-C from the work of Giancarlo Guizzardi & Wagner (2005).

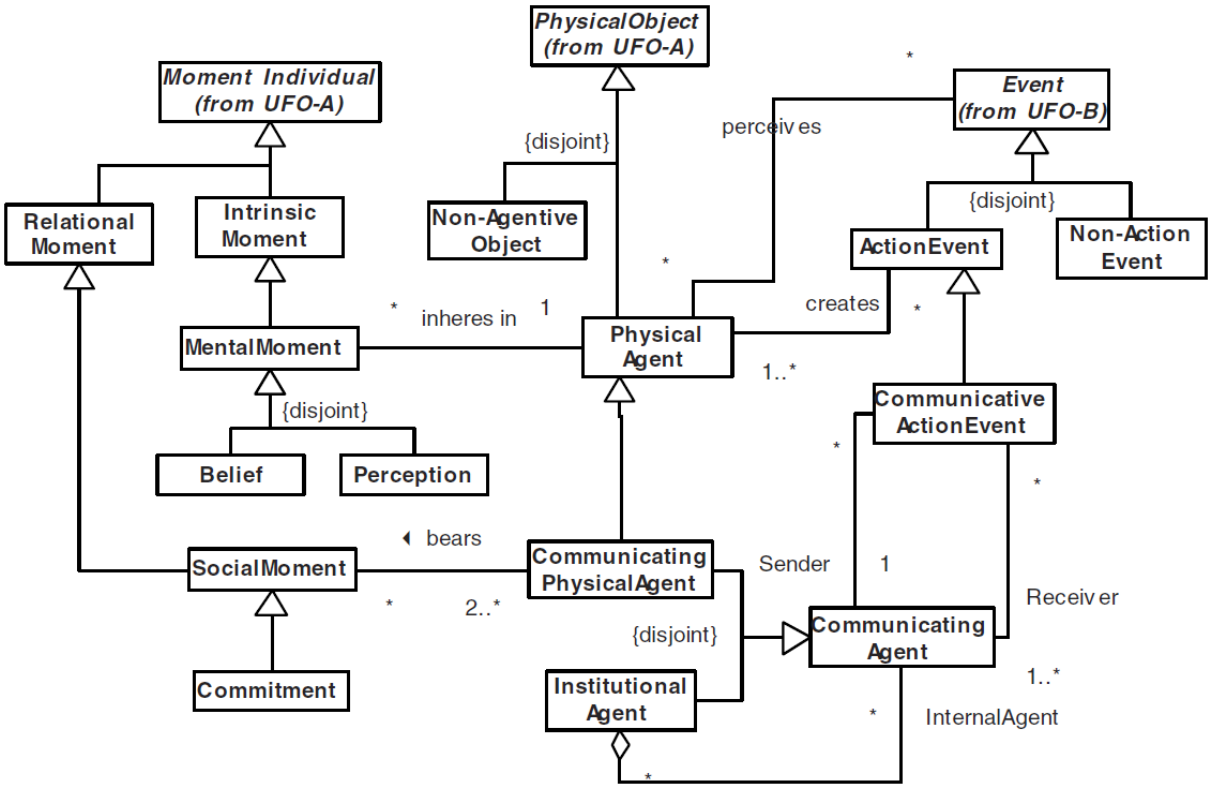


Figure 12: UML representation of UFO-C (Giancarlo Guizzardi & Wagner, 2005)

In UFO-C agents create events. For the purposes of supporting the MERODE/MOSES methodology, the *CommunicativeActionEvent* (subtype of *ActionEvent*) can be used, which is related to one agent that takes the role of *sender* in the event and to one or more agents that have the role as *receiver*. Also, relationships between two or more *CommunicatingPhysicalAgents* can be specified as a *SocialMoment* or *Commitment*. Therefore, the ties between interacting agents can be modeled explicitly. *Non-Agentive Object* can fulfil the role of the objects as specified by MERODE/MOSES. In UFO-C, this component has no predefined relationships between any of the other defined concepts. Therefore, to get the same relationships an object has in MERODE/MOSES, all these relationships should be manually added.

2.5.4 EVALUATION OF ALTERNATIVES

Choosing one of the abovementioned upper ontologies as alternative for the UML class diagram used in MOSES, the most important requirements are that it can be matched with MERODE as best as possible and that the interactions between actors can be easily derived from its models. The most important pros and cons of each upper ontology is summarized below, after which the most suitable alternative is chosen.

e³value supports the MERODE components object (e³value: value object), event (e³value: value offering) and actor. Although these components are related to them in a similar fashion as in MERODE, in e³value some relationships in MERODE require a chain of relationships, e.g. an event (e³value: value offering) relates to an actor via a "*consists-of*" relationship with value interface, which relates to actor. The notation of e³value models is, on the other hand, highly comprehensive, as value chains can be traced easily (Gordijn et al., 2005). The constraints and rules of e³value support profitability analysis of the business cases modeled with it. Next to profitability analysis e³value does not provide for many rules, like the constraints and rules MERODE defines. Therefore, if e³value would be used, the upper ontology needs to be extended with almost all constraints and rules defined by MERODE.

UFO-C also encompasses the MERODE components object (UFO-C: non-agentive object), event and actor (UFO-C: physical agent), but does not provide all relationships between them as specified by MERODE. If UFO-C would be used, the upper ontology needs to be extended with these basic relationships.

REA clearly supports the MERODE components object (REA: resource), event and actor (REA: agent). Also, their relationships are clearly specified. REA comes with a large predefined set of constraints and axioms the instances of the upper ontology need to respect. A large part of these constraints correspond with the rules and constraints entailed by MERODE. For example, in REA each inflow event needs to be related to an outflow event, which corresponds with the alphabet rule of MERODE (in section 4.2.2 the constraints of REA are mapped with the constraints of MERODE). If REA would be used, it is expected that only a few constraints should be needed to be added to the upper ontology. Therefore, REA is considered as most suitable upper ontology alternative and will be used as upper ontology in the redesigned MOSES methodology.

3 INTEROPERABILITY BENEFITS OF THE USE OF AN ONTOLOGY

To find out what aspects of ontologies affect the interoperability in a domain positively, this chapter describes a literature study enumerating these aspects. In the next chapter these aspects are used as a foundation for the mindset, method and notation of the development methodology. The literature in this chapter comprises of authors that directly compare properties of ontologies with those of information models, or only describing properties of information models or ontologies. Also, the most influential ontology development methodologies are reviewed in the literature research. This chapter starts with an overview on interoperability and how an ontology affects it in a domain. After that, an overview of the relevant ontology aspects is given, followed by a detailed description of each aspect. At the end of the chapter, the ontology development methodologies are reviewed.

3.1 ASPECTS IMPORTANT FOR THE STAKEHOLDER

The end users of the model are the people and organizations that have to participate in the interactions happening in the domain the model concerns. What is important for the stakeholders is to have the best support for interoperability in their domain. Therefore this section describes the literature about aspects of ontologies that are expected to improve the interoperability. It also describes some literature on other improvements ontologies bring compared with a (traditional) information model. The aspects are an important foundation for the philosophy, or mindset, behind the development of the development methodology in the next chapter. The literature is organized per aspect to generate a clear overview of each aspect. An overview of the important aspects is shown below in Table 4, followed by a detailed description of each aspect.

No.	Aspect	Mindset for ontology development
1	Vocabulary	Unambiguous concept descriptions and supports set theory on domain concepts
2	Validity rules	Limit instances and concepts to only valid configurations of the domain in reality
3	Context	Describe relevant concepts of the whole domain; not only system specific concepts.
4	Sharedness	Take into account different stakeholders need to work with and understand the ontology
5	Open world assumption	Not everything that is modeled with an ontology will restrict its interpretation of the domain
6	Descriptive	Describe behavior in the domain; do not prescribe it
7	Representation	Take the additional ways of representing a domain, like the definition and reasoning about sets and subsets and the definition of axioms on the domain, into account
8	Understanding	Ensure ontologies can be understood by humans
9	Formal semantics	Describe properties and behavior of domain concepts in a concise way
10	Automated reasoning	Rules and semantics of all concepts relevant to the stakeholders in the domain can and should be derived from domain experts and literature
11	System interoperability potential	Structure the ontology as an open system with decoupled, decentralized and configurable components
12	Dynamic modeling	Include dynamic behavior of the stakeholders by modeling their processes and operation rules

Table 4: Important ontology aspects

3.1.1 VOCABULARY

An ontology, as well as information models, provide a vocabulary for a language (Aßmann et al., 2006). A vocabulary provides a terminology for the concepts and roles that are being modeled (Gasevic & Djuric, 2006). For vocabularies it is important to have well-defined, unambiguous, definitions of concepts to prevent misinterpretations. Concepts are the sets of individuals and roles being binary relationships between these individuals. These can be either atomic or complex. Atomic concepts are simple and only have names. Complex concepts have descriptions, which can be expressed in a description logic language. This allows and facilitates the use of extensive set theories on concepts and of semantic relationships with other concepts.

As a vocabulary entails unambiguous descriptions of concepts and facilitates extensive set theories for concepts for ontologies, it is an important aspect to consider for improving domain interoperability and thus should be taken into account in the methodology. Specifically, the steps where the main concepts of the domain are identified, have to take this into account. These are mainly the step where the domain's actors and objects are identified and the step where the domain's events are identified in the phase where the basic shared domain model is determined.

3.1.2 VALIDITY RULES

Validity rules limit the modeling possibilities of concepts. Both ontology models and information models support this, although information models require an extension to do this. The validity rules for ontologies specifically aim at limiting the instances and concepts of an ontology to only valid configurations of the domain in reality (Aßmann et al., 2006). The validity rules assert instances to hold this consistency in any system (Gasevic & Djuric, 2006). By implementing this business logic, situation-specific data can generate document implications for the current situation, which therefore can improve interoperability in the domain it is applied on. Therefore, in the methodology interoperability needs to be taken into account as best as possible, so the instances and concepts modeled in the ontology are limited to configurations of the real world domain. Therefore in the improved methodology the ontology development phase has to comprise of steps that determine constraints on the concepts in the ontology to guarantee only domain configurations can occur that represent valid situations in the real world domain.

3.1.3 CONTEXT

When looking at the context of information models and ontologies, a clear distinction can be noticed. Information models are targeted towards a specific application and describe the concepts of reality, their interrelation and their static semantics. The development of these models are usually aimed at the use of them in a system to realize a system that is as efficient as possible (Henderson-Sellers, 2011). To ensure valid configurations of the real world are described, information models make use of well-formedness rules, written in, e.g. a language like Object Constraint Language (OCL), which is commonly used with UML class diagrams (Aßmann et al., 2006). Ontologies, on the other hand, are not aimed towards one or even more applications; they are intended to describe a whole, predefined, domain. Jarrar & Meersman (2009) even define ontologies explicitly as not capturing application requirements. In fact, the more independent the domain model of an ontology is, the higher its reusability in the domain.

When focusing on interoperability, a model that aims at describing (at least the essential parts of) a domain of a system would be of better use than a model that is designed for a specific application. Aßmann et al. (2006) call these two types domain models and system models. Whereas a system model is a description or specification of a system and its environment for a specific purpose, which is the type of ontologies. A domain model describes the environment, or domain, of a system, but is not necessarily focused on a system and its purpose only. This type is of information models. The methodology focuses on interoperability and uses an

ontology. Therefore the methodology should focus on describing concepts relevant for the domain and not only system-specific concepts. When determining the domain's basic shared model, the steps identifying the actors, resources, commitments, events and their relationships should take this into account.

3.1.4 SHAREDNESS

In general, information models are used for specifying or describing a specific system. They could be regarded as "plain artifact models" which can be seen as models that do not involve many other actors that need to know this model (Aßmann et al., 2006). Therefore information models usually do not have to take into account that others may want to use or understand it next to the system that it is built for; hence its sharedness is usually low.

Ontologies actively support a shared understanding between their users about the domain they describe (Aßmann et al., 2006). By sharing a model that is well-understood by all users in a domain, their interoperability performance can increase. The ontology itself should therefore take into account that it needs to be understood and be able to be used by all stakeholders. To embed this mindset in the methodology, the development of a basic shared domain model, like in the original MOSES methodology, should still be the first phase of the development methodology, creating a wide, shared model of the business domain. Also, the steps of this phase, identifying the relevant actors, resources, commitments and events in specific, should ensure these concepts are relevant and can be well-understood by all stakeholders. Next to that, also when more specific properties of actors and resources are determined (a step in the phase where the ontology is further elaborated), the same should be ensured.

3.1.5 OPEN WORLD ASSUMPTION

A property of an ontology defined by Aßmann et al. (2006) is the "open world assumption". This property means that anything not explicitly expressed by an ontology is, intuitively, unknown. Therefore ontologies can be seen as an under-specification of everything in the real world or as a partial model. For modeling purposes, all relevant concepts should be defined, but because of this property, concepts that exist in the domain but are not relevant for the reason the ontology is created can be left out. Information models underlie the assumption that what has not been specified is either implicitly disallowed or implicitly allowed (closed-world assumption) (Aßmann et al., 2006; Henderson-Sellers, 2011). The open world assumption of ontologies creates some space for the addition of extra interpretations or constraints to the domain modeled when deemed necessary for preserving the reality assertion in the ontology. Therefore the things modeled with an ontology should not cover as much domain information as possible, but should only cover the information and concepts relevant for the ontology's purpose. In the ontology development phase of the methodology, therefore it should be taken into account that only the necessary elements need to be determined when actors and objects and their properties are determined in the ontology.

3.1.6 DESCRIPTIVE

All ontologies are in fact a descriptive model of a domain. This means ontologies describe the real conceptualizations of a domain (see also Figure 4). The opposite of descriptive is prescriptive. Many information models are in the form of templates from which a computer system can be implemented (Aßmann et al., 2006). In these cases information models are prescriptive; a complete and final description is provided, indicating that anything not defined is wrong or not true. Aßmann et al. (2006) do note that sometimes ontologies are used in a prescriptive manner, however they should better not be called ontologies, but specification models. The methodology to be developed aims at creating an ontology before reaching platform-specific solutions. Therefore the behavior in a domain should be described; not prescribed.

Also, it is possible to distinguish two basic notions of the “is-represented-by” relation between a model and the corresponding conceptualizations (Aßmann et al., 2006). In a descriptive model (e.g. an ontology) the model describes the world, i.e. the world’s objects are related with concepts of the model in an “is-described-by” relation. In a specification model this is an “instance-of” relation.

As developers of a domain model and ontology might be inclined to prescribe the domain behavior instead of describing, in the improved methodology it should be explicitly noted and taken into account that the dynamic behavior in the domain should be derived only from the information about the domain. Especially in the methodology steps where the events happening in the domain are identified, described and linked with actors and resources, this aspect should be brought to the attention. This includes the steps where events are identified, the AOET is created and the UML activity diagrams are developed for determining the basic shared domain model. Also, when determining class instances and defining constraints to the behavior in the ontology model, this is an important aspect to be considered.

3.1.7 REPRESENTATION

Ontologies cannot be represented in UML or another (traditional) information modelling language, because an important criterion to evaluate ontology design quality is minimum ontological commitments, i.e. use as few notation types as possible (Falbo, Guizzardi, & Duarte, 2002). Therefore a graphical language must embody only notations that are necessary to express ontologies. This is not the case of UML and many other graphical languages used for expressing traditional information models. Also, because an ontology intends to be a formal model of a domain, it is important that the language used to describe it supports more functionality than a traditional modeling language that only describes classes and their properties and relationships. Section 2.2.1 looks at languages specifically aimed for modelling ontologies. A few of the beneficial modeling facilities of ontologies are the support for (reasoning about) sets and subsets of classes and the definition of axioms on the domain concepts. As the methodology involves an ontology, these additional ways of representing a domain should be taken into account. In the methodology, the steps building the base for the ontology, elaborating on their properties and determining their constraints in the domain need to take along these additional modeling techniques.

3.1.8 UNDERSTANDING

Henderson-Sellers (2011) states that ontologies can be understood by both humans as well as computers. This property enables both a good integration with computer systems as well as with the people using it. When models are well-understandable, they can also be shared and used for communication between different actors more easily since the model is understood well by all its actors. Therefore understanding is an important aspect for increasing the interoperability potential.

When comparing with information models, not all of these models are aimed at the understandability of humans. The focus of information models usually lies at developing an efficient functioning system, where models essentially only need to be understood by (the model designer and) the IT system. To reach a high understandability of the methodology, the ontology should be able to be understood by humans. This especially plays a role when defining the actors and resources of the domain into the basic shared domain model.

3.1.9 FORMAL SEMANTICS

An ontology is intended to be a formal model of a domain (Falbo et al., 2002). Therefore an ontology needs a formal method for capturing the semantics of the domain. By using a formal language, the model elements can be represented in a precise and unambiguous way. Also, the specification of formal axioms on the

interpretation of the model structure is supported by the formal semantics. This way misinterpretations can be prevented and concepts and their properties and behavior in a domain can be precisely described. Therefore the methodology should also include concise and unambiguous properties and behavior of domain concepts. The methodology steps that determine the ontology concepts' properties and formal constraints should therefore take this into account. Note that because the open world assumption applies for ontology models (see also section 3.1.5), this does not mean the defined concepts are exhaustive, i.e. concepts could comprise more properties than are modeled and other concepts could still be added to the model, still having a model conform the real world.

The majority of traditional modeling methods do not support formal semantics, like UML. At the moment there are model extensions being developed to also support the capturing of semantics, like pUML (Evans & Kent, 1999). Even though information models can be extended with formal semantics, they are only an addition that is not part of the core thoughts behind information modeling.

3.1.10 AUTOMATED REASONING

Other aspects of ontologies, like validity rules and formal semantics, described above allow for automated reasoning about the described conceptualization of a domain (see also Figure 4) (Henderson-Sellers, 2011). In particular the support for disjoint classes, set intersections and set complements are seen as important modeling concepts for facilitating automated reasoning. The rules and formal semantics can be derived from relevant domain knowledge sources, like domain experts and literature about the domain. Because of the rules and restrictions an ontology also includes, ontologies can also describe the behavior of domain concepts. Automated reasoning is therefore possible with ontologies. Because of that, ontologies that derive their knowledge from domain experts and literature can prove powerful automated reasoning tools. When an ontology is shared between stakeholders, each can reason the same way about the domain and therefore interoperability between these stakeholders is facilitated (Smith & Welty, 2001). To incorporate this aspect in the methodology, the step where the properties of the actors and resources are determined in the ontology need to take into account that the properties are as suitable as possible for automated reasoning about them. Thereupon the methodology step determining the formal constraints for these actors and resources can perform automated reasoning to construct ontology constraints.

3.1.11 SYSTEM INTEROPERABILITY POTENTIAL

For achieving a system that has a high potential for interoperability, Daclin et al. (2006) defined four properties the architecture of a system should comply with. The first property is *open*, which means components should be open for modification or upgrading. The architecture should comprise of *decoupled* components, so they should be able to continue their functioning independent of other components in their environment. Another property that improves interoperability potential of a system is the *decentralization* of the components in the architecture. This means that there is no central component that organizes the functioning of the whole system. The fourth property of a system should be to have *configurable* components, where attributes and other properties of the system should be able to be configured easily. As the methodology aims for the best interoperability potential, the ontology involved should comprise of decoupled, decentralized and configurable components. This plays an important role in the methodology steps where in the basic shared domain model the components are related to each other, where the base for the domain ontology is built based on the basic shared domain model and where components of other ontologies are reused and placed in the ontology model.

3.1.12 DYNAMIC MODELING

The collaboration and interoperation of different stakeholders on enterprise level can be supported by a shared ontology between the stakeholders (Panetto & Molina, 2008). To fully support interoperation, the stakeholders should be able to have insights in the dynamic behavior (i.e. business processes and operation rules) of the other stakeholders of the collaboration. The literature study of Panetto & Molina (2008) shows that on knowledge level of enterprise integration and interoperability, knowledge about business processes and operation needs to be shared among the involved parties. Therefore the ontology of the method, which will be shared among the involved parties in the domain, should include the processes and operation rules derived from the dynamic behavior of the stakeholders in the real world. In the improved methodology, this needs to be considered in the steps where the events happening in the domain are identified and elaborated upon in UML activity diagrams, but also at the point where the events are imported from the basic shared domain model into the ontology model and where class instances are determined, good insights are required in the domain's dynamic behavior.

Panetto & Molina (2008) make a distinction between compatibility and full system integration. Compatibility is something less than interoperability, where systems interfere with each other's functioning. Full system integration goes a step further than interoperability, where a certain degree of functional dependence is involved. Interoperability, as argued, lies in the middle of an integration continuum between compatibility and full integration.

3.2 MINDSETS FROM ONTOLOGY DEVELOPMENT METHODOLOGIES

As there is not one standard for the development of an ontology, different methodologies exist. Different authors have different perspectives, mindsets and notation styles for developing an ontology. To gain insight in how ontologies can be developed and in which ways they can contribute to interoperability in a domain, the most influential (i.e. most cited and reviewed) methodologies are reviewed. Table 5 gives an overview of the mindsets important for building the ontology facilitating interoperability deducted from the other ontology development methodologies.

No.	Method	Mindset for ontology development
13	Enterprise ontology	Capture domain knowledge initially in a semi-informal way using carefully crafted natural language definitions to prevent loss of knowledge from e.g. domain experts
14	Methontology	Reuse and integrate existing ontologies parallel to the other ontology development activities
15		To guarantee ontology completeness, all terms should be concise, partial complete and consistent
16		Domain knowledge can be acquired by interviewing experts and reviewing literature about the domain
17	TOVE	Formulate competency questions based on the main problem and scope. When the ontology can be used to answer all questions, the ontology can be considered complete, for its intended purpose
18	Ontology Development 101	When there are viable modeling alternatives, keep the intended application and anticipated model extensions in mind
19		Ontology development is an iterative process involving interviews with domain experts or application testing
20	DILIGENT	Involve different types of stakeholders: ontology engineers, domain experts, but also end-users

Table 5: Important mindsets from ontology development methodologies

Gasevic and Djuric (2006) looked at several ontology development methodologies and concluded that most focus on building ontologies, some include methods for merging, reengineering, maintaining and evolving ontologies, and others build on general software development processes and practices. They note there is no best methodology, as there is no "correct" way to model a domain and that ontology development is necessarily an iterative process. Two conclusions are drawn from their survey: (1) many common points between methodologies exist, usually parts are only named differently or have different granularity, (2) many principles and practices are analogous to those of software engineering.

3.2.1 ENTERPRISE ONTOLOGY

A simple, yet effective approach to ontology development is the *Enterprise Ontology development methodology* coined by Uschold & King in 1995 (Fernández-Lopéz & Gómez-Pérez, 2002). The method involves 4 main steps: (1) identify the purpose of the ontology to clarify why it needs to be built and what its intended uses are. The second step is (2) building the ontology. This step is broken down into 3 steps: (2.1) capturing, which means identifying the key concepts and relationships in the domain of discourse, (2.2) coding, which is the capturing of the concepts and relationships in a formal specification language and (2.3) integrating existing ontologies, which means reusing eventual existing ontologies in the ontology to be developed.

The idea behind the capturing step (2.1) is that unambiguous text definitions are defined for all concepts and relationship. These definitions should be defined in a semi-informal way; defined through carefully crafted natural language definitions that are as precise as possible (Helena Sofia Pinto & Martins, 2004). This facilitates the formalization of the domain knowledge in the following step for the domain experts (M Uschold & Gruninger, 1996). Based on these definitions, a formal model can be developed in a more restricted (ontology) language. Uschold & Gruninger (1996) also see the best way to identify which concepts to capture in the ontology is by means of a middle-out approach. This approach prevents a level of detail that is either too high or too low by starting out from the most important concepts, then defining higher level concepts in terms of these and specializing the concepts when necessary (Fernández-Lopéz & Gómez-Pérez, 2002).

Step (3) is the evaluation of the ontology with requirement specifications, competency questions and/or the real world. The final step (4) is documentation of the ontology. The approach recommends to take the middle-out approach when capturing the concepts of the domain, which means that the most important concepts should be identified first, then generalized and specialized in other concepts.

An important aspect of the Enterprise ontology development methodology is to prevent loss of knowledge from e.g. domain experts. This is, in any case, an aspect of domain ontology development that should be taken into account. Therefore the mindset should also be adopted for the improved methodology in the next chapter to capture knowledge in carefully crafted natural language definitions. This approach should be adopted at least at the steps where the actors, events and objects are identified for the basic shared domain model, but also where they are specified further in the phase of ontology development. Also in this phase, domain constraints should be determined first in a more informal way, to formalize them in a later step.

3.2.2 METHONTOLOGY

Fernández-Lopéz and Gómez-Pérez (2002) compared ontology development methods with the IEEE standard for developing software life cycle processes. They conclude that no methodology is fully mature compared to the IEEE standard and that each methodology seems to have different approaches, which makes it difficult to integrate all into one unifying methodology. *METHONTOLOGY* seems the most mature of the methods reviewed. This method can be used for both designing ontologies at the knowledge level from scratch as well as combining existing ontologies (Oscar Corcho et al., 2003). It is based on the activities defined by the

standard software development process of IEEE. Therefore it extensively defines and describes management, development and support activities.

Project management activities include scheduling, which plans which tasks are to be performed, how they have to be performed and what resources are required (Fernández-Lopéz & Gómez-Pérez, 2002). Control guarantees task completion and quality assurance assures the output of each task is sufficient. The development-oriented activities include specification, where the intended uses and end-users of the ontology are stated. Conceptualization is the following activity, where the knowledge of the ontology is put in a model. Fernández-López, Gómez-Pérez, & Juristo (1997) see that total completeness of an ontology specification never can be proven, so it has to be guaranteed that the ontology specification is sufficient. The authors state that the model needs to be concise (each term is relevant and there are no duplicate terms), partial complete (the terms cover the required terms for solving the initial problem) and consistent (the meaning of all terms make sense in the domain). For acquiring this knowledge, the authors suggest to perform structured and non-structured interviews with experts and text analysis of books and other literature on the domain.

The next development activity is formalization, which intends to transform this model into a formal model. The next activity is implementation, where the formal models are transformed into a computable language and the final activity is maintenance, where the authors see the ontology continuously needs updates and corrections throughout its life.

Methontology also denotes support activities supporting the development activities. These include knowledge acquisition, integration, evaluation, documentation and configuration management. Knowledge acquisition and evaluation are mostly performed at the initial activities, where domain knowledge has to be acquired and the ontology specification need to be evaluated. Integration of other ontologies is seen as a parallel activity to the development activities. For controlling the changes to the ontology, documentation on each phase and generated product is generated, and by configuration management, these changes are controlled.

A mindset relevant for our methodology that is underscored by Methontology is to model domain concepts as concise and consistent as possible. This strokes with the mindset identified in section 3.1.9 about formal semantics. The conciseness and consistency of terms is in the improved methodology of highest concern at the steps where the properties of the actor and resource concepts are specified, as well as where the ontology constraints are formalized.

Another relevant mindset involves the reuse and integration of other existing ontologies. Like Methontology, this activity can be seen as a parallel activity to the other development activities. Our methodology therefore should also include this step in the phase where the actual ontology is developed. Also, additional knowledge about the domain can be retrieved by interviewing experts and reviewing domain literature to build and extend the domain ontology. The latter is especially important for the methodology steps where information about the domain is entered in the domain model and ontology, being the steps that identify the domain's actors, resources, commitments, events and their relationships, but also the steps refining them with additional properties and constraints.

3.2.3 CYC

Specifically for system communication and interoperability the *Cyc* ontology development method is developed (Fernández-Lopéz & Gómez-Pérez, 2002). It is mainly oriented to support the acquisition of knowledge, in three different degrees of automation of knowledge acquisition (O Corcho et al., 2007). The methodology has therefore three phases: (1) manual extraction of common sense knowledge, (2) computer aided extraction of common sense knowledge and (3) computer managed extraction of common sense knowledge. Where the first phase proposes manually coding because of the lack of the ability of learning machines to search for new common-sense knowledge, the second phase involves a person codifying knowledge that is already present in a

(Cyc) knowledge base. In the third phase knowledge sources are automatically fed to a knowledge tool for automatic knowledge extraction.

3.2.4 TOVE

The *TOVE* ontology development methodology is more elaborated and focuses on competencies to both formulate and evaluate the ontology (Fernández-Lopéz & Gómez-Pérez, 2002; Helena Sofia Pinto & Martins, 2004). The first step involves the development of motivating scenarios that set intuitive possible solutions to the scenario problem. In essence, this step identifies the problem and scope of the ontology to be developed. Step two is to formulate informal competency questions that are inspired by the first step. The ontology should be able to give answer to all these questions for it to be considered complete for the intended purpose of the ontology. Then the terminology of the ontology needs to be specified in a formal language (step 3). After this is done, the competency questions can be formalized in step 4. In step 5 axioms and definitions for the terms can be defined. In the 6th step the conditions under which the solutions to the competency questions are checked on completeness.

The TOVE methodology sets clear goals for the development of an ontology by formulating competency questions based on the main problem and scope. This mindset is deemed to result in highly effective ontology models as it touches the real reasons why the ontology is developed and what it should comprise of after its development. Therefore at the start of the initial phase where a basic shared domain model is determined, the scope of the model should be identified by posing competency questions.

3.2.5 ONTOLOGY DEVELOPMENT 101

Noy & McGuinness (2001) developed an ontology development method with the notion that there is no one correct way to model a domain. They notice there are always viable modeling alternatives and the best alternative depends on its intended application and possible extensions anticipated. Also, to develop an ontology, for its concepts to reflect the real world in the best possible way, evaluation of the ontology is required. This can be done by using the ontology in applications, problem-solving methods or by discussing it with experts in the field. In many cases these evaluations lead to revisions of the initial ontology, which implies the ontology development should be an iterative process.

The methodology itself consists of 7 steps: (1) determine the ontology domain and scope; Noy & McGuinness (2001) make use of posing competency questions, as proposed by the TOVE ontology development methodology. (2) Investigate existing ontologies for reuse. (3) Enumerate important terms in the ontology; the method does not prescribe a way how to do this, but assumes in some way a list of terms with certain properties is created. (4) Define the classes and class hierarchy in a modeling tool (e.g. Protégé). (5) Define the properties of these classes; also performed in the modeling tool. (6) Define the facets of the slots (properties); in the same modeling tool the value types (e.g. string or Boolean), domains and ranges (i.e. another class that is able to generalize the property) of each property should be defined. (7) Create instances; based on the classes, define individual instances that have certain properties defined.

Following these steps leads to a concrete ontology model. Because the methodology is specialized on the use of a modeling tool (Protégé), the method is very practical. The thoughts behind the steps in the method are based on the belief that the developer already has an idea of how the concepts should be defined and how they relate to each other. This might not always be the case, nonetheless steps 4 to 7 clearly describe what steps to undertake using an ontology modeling tool. After the methodology has been described, (Noy & McGuinness, 2001) describe how to model special cases in an ontology modeling tool, which is very practical for ontology developers.

In our development methodology modeling alternatives might also come up. Therefore keeping in mind the intended application and anticipated model extensions is an important mindset to take into account as well. It is expected that modeling ontology alternatives will come up in the methodology step where the properties of actors and objects in the ontology are specified, but also where class instances and their restrictions are determined.

Also, the mindset that many iterations are required in the ontology development is important, because for domain ontology development many domain experts and a lot of literature need to be consulted and model evaluations need to be performed. Especially in the development phase where the basic shared domain model is determined, a lot of information about the domain is retrieved and therefore this phase should be iterated.

3.2.6 DILIGENT

Pinto, Tempich, & Staab (2009) take a slightly different approach than the before-mentioned ontology development methodologies. The authors developed a methodology for distributed loosely-controlled evolving ontologies (abbreviated to DILIGENT). Interoperability between all stakeholders plays an important role. The method reasons that the stakeholders of a domain participate in some form of cooperation-relationship. Even in the case of competitiveness for the same resources, participants could still collaborate to compete against external threats.

The core idea behind the method is to start out from a small and useful shared ontology. The stakeholders then adapt their own version of the shared ontology for their own purposes, in a local variant. A control board of stakeholders analyze all local ontology variants and develop a new version of the shared ontology. This way an ontology evolves that is shared by all stakeholders and, in the end, comprises of all relevant domain concepts entailing the most optimal ontology supporting interoperability among the stakeholders. The board should have a well-balanced participation of different kinds of actors. It is important to, at least, involve ontology engineers, domain experts and users of the system.

The ontology development method itself is divided in 5 phases. The first phase is called *build*; in this phase the initial, small, ontology is developed from scratch. Although no development method is prescribed, the ontology needs to focus on usability and usefulness. Phase 2 is called *local adaptation*. In this phase the ontology is used and tailored by each stakeholder individually. In phase 3, *analysis*, the board of stakeholders analyzes incoming requests and observations of changes for the shared ontology. The board also decides on which changes would most benefit the users. The fourth phase is *revision*, where the board makes a revised version of the shared ontology that includes the changes decided on in the previous phase. This revised shared ontology is released in phase 5, *local update*. The users put the new ontology to use in their system. After this phase, the ontology development iterates and the process is continued from phase 2 on.

An important aspect of this ontology development methodology for our own methodology is to involve stakeholders of all different types to create a widespread cover that is relevant for all stakeholders. Ontology engineers, domain experts, but also end users should therefore be considered. In our method this should be taken into account at the steps where domain information is added to the basic shared domain model, being the steps where the actors, resources, commitments, events and their relationships are identified, but also during the whole phase where the ontology model is developed using expert information.

4 DEVELOPMENT METHOD FOR ONTOLOGIES FOSTERING INTEROPERABILITY

This chapter describes the proposed improved MOSES methodology. First, the methodology steps are explained, followed by the reasoning about each modeling choice (i.e. the mindset behind the methodology). At the end of the chapter, the way of notation is elaborated upon.

4.1 THE METHODOLOGY STEPS

The methodology is a result of many iterations as a result of comment of modeling experts, domain experts, but also the case study described in the next chapter. Figure 13 shows an overview of the steps to take in the methodology. Table 6 shows per methodology step a short description, the input for the step and what deliverable (e.g. a diagram or overview) the step results in. Also, the important methodology mindsets from Table 4 and Table 5 are mapped to each step. The reasoning behind each step is elaborated in the next subsection.

The methodology focuses on developing a model on the level of the business and information level of a business domain. It comprises of 4 main phases; *determine basic shared domain model*, *build ontology base*, *develop ontology* and *determine technology-specific solution*. Compared with the original MOSES methodology the first phase can be compared with the *business domain modeling* phase, where in both phases a foundation is laid supporting the actual information modeling activities later in the methodology. Also, like in the MOSES methodology, this phase requires design iterations to come to a complete initial model. An addition to this first phase is the initial step where the scope of the domain to be described is determined. Also, throughout the whole development methodology the REA upper ontology now plays a large role. This can be recognized in the identification steps, where the domain concepts are mapped to resources, agents, commitments and events of the REA upper ontology.

The second and third phase can be compared with the *business information modeling* phase of MOSES. Instead of building an information model from which a solution can be derived, an ontology is developed from which a solution can be derived. The reason that two separate phases are chosen to “replace” the old phase is that a conversion needs to take place from the basic shared domain model to the REA-ontology. After this conversion, the ontology can be specified further in an iterative fashion.

The final phase of both the old and new version of the MOSES methodology are about the same; derive a technology-specific solution from the model developed in the previous phase. The difference is that in the new methodology the solution is derived from an ontology instead of an information model.

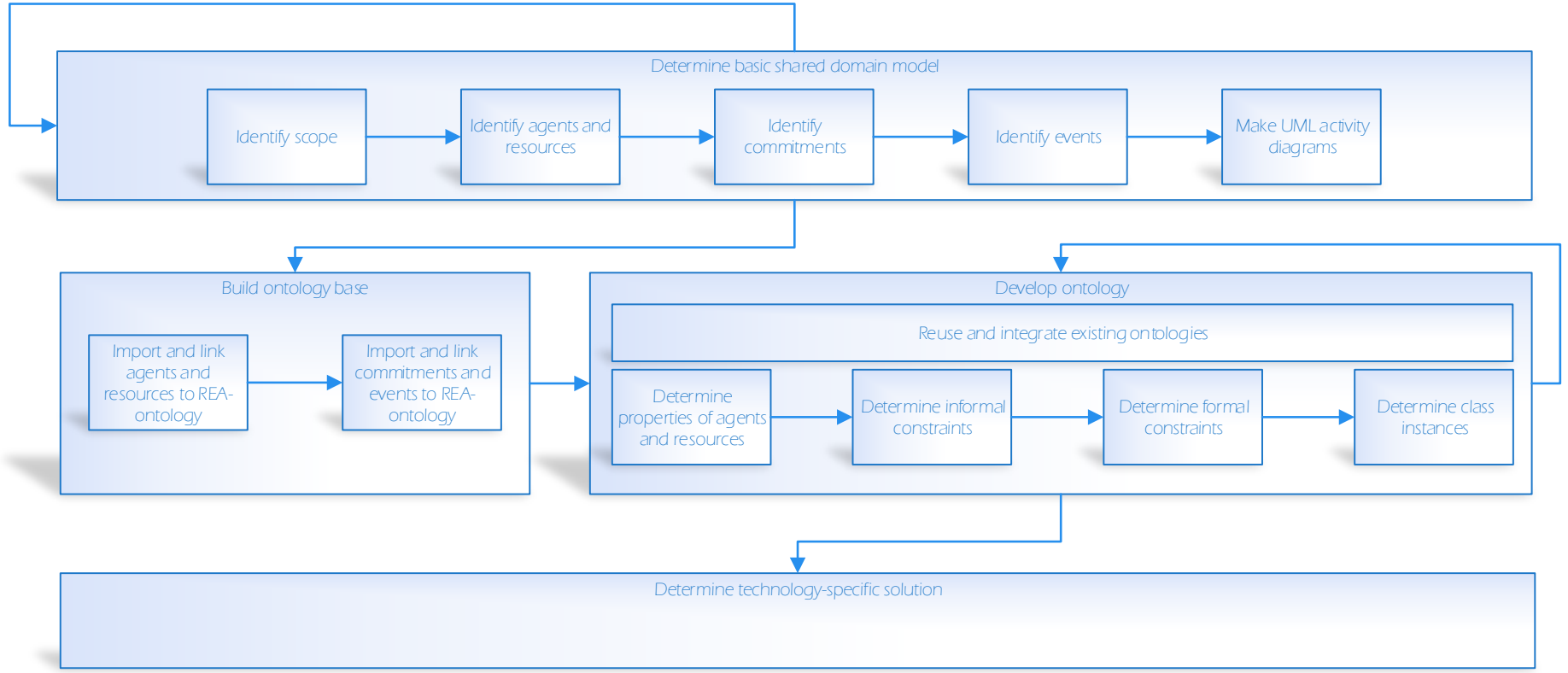


Figure 13: Methodology steps

Step	Description	Mindset	Deliverable	Input
Determine basic shared domain model	Identify the main relevant concepts of the domain, their relationships and the events they are involved in. This is an iterative process.	4, 19		
Identify scope	Identify the main scope and determine with the stakeholders the competency questions the system to be developed needs to be able to answer	17	Competency questions	Interviews domain experts
Identify agents and resources	Identify the actors and objects involved in the interoperability interactions in the domain	1, 3, 4, 8, 13, 16, 20	Actor and object descriptions	Interviews domain experts; domain literature
Identify commitments	Identify the commitments between actors with resources of the domain. The relationships and classes can then be visually represented in a UML class diagram	3, 11, 16, 20	Commitment descriptions and UML class diagram	Interview domain experts; domain literature; actor, event descriptions
Identify events	Describe the events happening in the domain related to the identified commitments. Based on these events, create an AOET	1, 3, 4, 6, 12, 13, 16, 20	Event descriptions and AOET	Interviews domain experts; domain literature; actor, resource, commitment descriptions
Make UML activity diagrams	Based on the identified events, create UML activity diagrams	6, 12	UML activity diagrams	Event descriptions
Build ontology base	Create an ontology model based on the basic shared domain model. From this ontology an interoperability solution can be derived. It should be clear which ontology language and editor are used. The created ontology makes use of the REA upper ontology.	11	Domain ontology base using the REA upper ontology	REA-ontology; domain model
Import and link agents and resources to REA ontology	Create classes for each object and actor, having either Resource or Agent as superclass	5, 7		REA-ontology; domain model;
Import and link commitments and events to REA-ontology	Create classes for each event, having Event as superclass. Also create properties for each relationship	7, 12		REA-ontology; domain model
Develop ontology	Extend the ontology base with domain knowledge	11, 20	Domain ontology using the REA upper ontology	Domain ontology base
Reuse and integrate existing ontologies	Parallel to the other activities in this phase, where possible existing ontologies should be integrated and reused in this ontology	14		Existing related ontologies

Determine properties of agents and resources	Give actors, objects and events properties (i.e. attributes) that are relevant for the domain	4, 5, 6, 7, 9, 10, 13, 15, 16, 18, 20		Interview domain experts; domain literature
Determine informal constraints	Create semi-informal constraints for the objects, events and actors in the model to ensure only realistic domain configurations of the ontology are possible	2, 13, 16, 20	List of semi-informal constraints	Interview with domain experts
Determine formal constraints	Formalize the semi-informal constraints in the ontology	2, 7, 9, 10, 15	Ontology constraints	List of semi-informal constraints
Determine class instances	Create instances of the class types as they exist in the real world domain	6, 12, 16, 19	Ontology class instances	Interview domain experts; domain literature
Determine technology-specific solution	An interoperability solution can be directly derived from the designed ontology.		Domain ontology	

Table 6: Methodology steps and their description, mindset, input and deliverable

4.2 THE METHODOLOGY MINDSET

The steps defined in Figure 13 and Table 6 have been defined and ordered in a thought-out way. The philosophy, or mindset, behind them is explained in this section. Each step is based on a specific mindset grounded with the literature reviewed in the previous chapters. Also the main ideas what should be achieved or done by a step will be elucidated.

4.2.1 DETERMINE BASIC SHARED DOMAIN MODEL

This methodology step, or phase, origins from the original MOSES methodology. As elaborated on in section 2.4, this phase is closely linked to the MERODE methodology where actors, objects and events are central for the development of a shared business domain model. It is important to develop this domain model in such a way that all stakeholders involved should be able to do their work with the resulting model (Aßmann et al., 2006). Interviews with domain experts is therefore a crucial input for this process and also the analysis of domain literature (e.g. process documentation and workflows) is important.

The phase is adapted towards the use of the REA-ontology in a later stage of the methodology. Because the concepts in REA are more elaborated, the adaptations are extending the original methodology at this aspect. MERODE only distinguishes between objects and events, whereas REA distinguishes events and 3 different types of objects, being resources, agents and commitments. As is explained in section 4.2.2, these object types can be mapped to these objects, but for the development process a structured approach to identify these 3 objects in the initial phase can support the ontology development in a later phase. Also, because it is expected for the expressiveness of concepts is higher in an ontology model than in a UML class diagram, the addition of concept details is moved out of this phase, to the phase where the ontology is developed. Therefore, where the original approach specifies the identification of concept properties and other details, the new approach does this in the phase where the ontology will be developed.

When developing the shared business domain model, it can become complicated to maintain consistency between all the events, resources, commitments and agents related to each other. Therefore, the tool called MERMAID (or JMERMAID)[1] can be used for creating events, resources, commitments and objects linked to each other. The tool ensures consistency during the development and can output straightforward Object-Event-Tables and class diagrams. In particular for the generation of a class diagram and OE-Table, this tool proves useful.

Domain experts and literature not always provide a clear, complete, overview of all actors, objects, events and processes happening in the domain. Therefore, this process should be iterated until a complete, full, ontology of the domain is defined (Noy & McGuinness, 2001).

4.2.1.1 IDENTIFY SCOPE

As was identified by Fernández-Lopéz & Gómez-Pérez (2002), the reasons why an interoperability solution needs to be made should drive the whole process and its methodology. Therefore, the methodology should start by identifying the purpose and scope of the project. Also, competency questions should be developed together with the stakeholders. These questions should comprise all the things the solution should be able to fulfil. The set scope and competency questions can be referred to throughout the whole development process to ensure a tight link with the scope and the actual problem to be solved.

[1] MERMAID can be found at http://merode.econ.kuleuven.ac.be/mermaid.aspx

4.2.1.2 IDENTIFY AGENTS AND RESOURCES

An agent is a party involved in the interactions in a domain. A resource supports the business activities of the involved agents. When identifying them, it is important to define the resources and agents as unambiguous as possible to prevent misinterpretations of data and to facilitate reasoning about specific versions of a resource type (Gasevic & Djuric, 2006). Also, the resources and agents defined need to be described in the way they are in the domain; not how you want to specify them in a system to be built using this model (Jarrar & Meersman, 2009). The descriptions and names should be understandable by computers as well as human readers (Henderson-Sellers, 2011).

The thought behind this step is to specify semi-informal descriptions of agents and resources here, which are expressed by and highly comprehensible for humans to ensure no essential information is lost when domain experts are interviewed (Helena Sofia Pinto & Martins, 2004). The domain experts should comprise of a variety of stakeholder types, so not only domain specialists, but also system end-users, system engineers and others (H.Sofia Pinto et al., 2009).

4.2.1.3 IDENTIFY COMMITMENTS

Agents and resources relate to each other. In the perspective of REA, one resource type (often physical resources, but also other types such as information) is exchanged between two agents. In cases where more than 2 agents are involved and/or more than 1 resource type is involved in one commitment, it should be possible to split this commitment in multiple commitments. When developing an ontology in a later stage, the information commitments contain can be used for reasoning purposes.

Using the identified information on resources, agents and commitments, an existence dependency class diagram can be drawn that summarizes this information. For each commitment a short description can be made to give insights in the context and meaning of the commitment. The information on the relationships can be based on interviews with domain experts and domain literature. As with all identification steps in this methodology, it is important to create descriptions as unambiguous as possible (Gasevic & Djuric, 2006). This gives an overview of the domain concepts, which will also be used as input for the ontology development in the next phase. In this step it is important to keep in mind that the interoperability potential of the domain model can be capitalized by ensuring the model structure is open with decoupled, decentralized and configurable components (David Chen et al., 2008).

4.2.1.4 IDENTIFY EVENTS

When actors are performing business activities to fulfil their commitments, events are triggered that induce cooperation between different agents. These events occur at a specific moment in time and may involve interaction with other objects. As noted by Panetto & Molina (2008), the dynamic behavior of the stakeholders involved should also be taken into account. This means business processes and operation rules of the parties involved should be included in the model. The original MOSES methodology also takes this into account, where this methodology widens this to ontologies in the steps following.

Like with the previous step, the identification of events should also take into account the ambiguity of terms and the understandability of humans. Events should also be described as they happen in the domain (and not in a future information system) and in a semi-informal way ensuring no essential domain knowledge is lost.

Now the actors, resources, commitments and events of the domain are identified, the events can be linked to the actors, commitments and resources involved. As the MERODE methodology prescribes, the creation of an Object-Event Table (OET) facilitates the understanding of each event and creates overview on how the events,

actors and objects are connected. As described in section 2.4 the MERODE methodology sets several requirements to the lifecycle of objects in an instantiated model, the definition of the type of participation an object has and the definition of the type of responsibility an actor has in an event. As with the previous steps, also in this step it is important to keep in mind to describe the behavior and relations of an event as it is in the domain, and not in a contemplated information system.

4.2.1.5 MAKE UML ACTIVITY DIAGRAMS

Also part of the original MOSES methodology is the creation of insights in event sequences. As an alternative to clarify the event sequences, UML activity diagrams can be used. This type of diagram is more frequently used and supports the modeling of parallel events and events influencing other events. Therefore in this methodology event sequences will be expressed in UML activity diagrams. The mindset behind expressing event sequences is that this generates a clear overview on the order of execution of events, which facilitates the sequencing of events in the ontology later on. Also, modeling the dynamic behavior of a domain is argued to increase interoperability (Panetto & Molina, 2008).

4.2.2 BUILD ONTOLOGY BASE

This next step clearly distinguishes itself from the original MOSES methodology. By developing an ontology instead of an information model, this new methodology distinguishes itself. As described in section 2.5, the REA-ontology will be used as ontological base where all actors, objects and events can be related to. The philosophy behind this is that when the ontology is completed, it is very easy to reason about it, extend or adapt the ontology and to generate the technology-specific solution from it. When performing this step, it should be clear which ontology language and editor will be used.

To building an ontology, this step comprises of a transformation of the basic shared domain model to an ontology using the REA ontology foundation. Two steps are involved importing the actors, objects and events to the ontology base. It should be taken into account that the concepts imported should be structured as an open system to enhance the ontology's interoperability potential. This means the components need to be decoupled, decentralized and configurable (Daclin et al., 2006).

To build this ontology base, the links should be defined between the REA ontological foundation with MERODE and the exact definitions of the used concepts should be stated. Therefore, Table 7 gives an overview of the definitions of the REA concepts. An overview of the definitions of the MERODE concepts can be found in Table 3. Table 8 maps the elements of the REA ontological foundation with the elements of MERODE. Even though the REA ontological foundation does not precisely match all rules of MERODE, the overlap is large (Geerts & Mccarthy, 2000; Snoeck et al., 1999, 2003). The REA upper ontology, in combination with OWL, does not fully cover all aspects of MERODE. Therefore, to close this gap, a few OWL restrictions need to be defined. Table 9 shows which restrictions are added to which REA class. Figure 14 shows the REA ontology in UML to be used in the method, mapped with the elements of MERODE.

REA concept	Definition
Resource	Corresponds to a real-world economic resource. It has certain properties, an identity and value. A resource is the subject of the economic exchanges in the domain.
Event	Corresponds to something that happens in the real world. As REA focuses on economic exchanges, an event in REA corresponds with a business transaction where exactly two *agents* exchange *resources*. *Events* are atomic, meaning it cannot be split in sub-events. Economic exchanges are by nature a duality relationship, where both an economic inflow and outflow are involved. Therefore, an *event* always has a duality relationship with another *event* that is either an inflow or outflow event. This also means that an event has

	exactly one *stockflow* that describes a *resource* inflow and exactly one *stockflow* that describes an outflow of this same *resource*.
Agent	Corresponds to a real-world stakeholder participating in *events* in the domain of discourse, for which it made certain *commitments*.
Stockflow	Describes the connection of *event* with *resource*. The event can lead to a stockflow that is *using, consuming, giving, taking* or *producing* exactly one resource, which this relationship is representing. Stockflows also have a nature of duality. A transformation can have a *use* or *consume* stockflow leading to a *produce* stockflow. An exchange should include both a *give* and *take* stockflow. When resources are *used*, sometimes they disappear from the domain of discourse. When resources are *consumed*, they always disappear. At a *production* stockflow, a new instance of resource is created.
Commitment	The agreement between (exactly) two *agents* to execute at least one *incremental event* and at least one *decremental event* of the same resource. A commitment has a *reserves* relationship with *resource* to describe the inflow and outflow *resource type* scheduled by the commitment. It also has a *partner* relationship with *agent* to describe the *agent*s involved.
Participation	Describes an *agent* involved in an *event*. An *agent* can either be participating as receiver or provider in an *event*.

Table 7: Definitions of the REA concepts

MERODE	**REA & OWL**
Alphabet rule: each event has only one effect on objects of a class (either creates, modifies or deletes objects)	Definition *stockflow*: an event relates to exactly one resource via a stockflow-relation.
Alphabet rule: each object class needs at least one event for its creation and deletion	Axiom 1: At least one inflow event and one outflow event exists for each economic resource; conversely inflow and outflow events must affect identifiable resources
Alphabet rule: the behavior of an object type P must contain all and only the event types for which there is a C, M or E in the column of P in the OET	Axiom 2: All events effecting an outflow must be eventually paired in duality relationships with events effecting an inflow and vice-versa
Propagation rule: when object type D is existence dependent of an object type M, the latter is by default also involved in all event types D is involved in	OWL inheritance of classes: subclasses inherit properties (including associated event types) from their superclasses
Type of involvement rule: an existence dependent object type cannot start to exist before its master	Not natively supported. By using an OWL restriction (1) this rule can be asserted.
Inheritance rule: an object type inherits all event types and properties from its parent object type, either unchanged or specialized	OWL inheritance of classes: subclasses inherit properties (including associated event types) from their superclasses.
Default life cycle rule: objects must include at least 2 events: its first needs to be an object creation event, its last needs to be an object ending event	Not natively supported. By using two OWL restrictions (2 and 3) this rule can be asserted.
Default life cycle rule: events need to have a determined sequence order	Not natively supported. This rule can be asserted by adding a “next” association between *commitments* to indicate the sequence of *commitments* and therefore *events*.
Restriction rule: existence dependent object types must have a more deterministic life cycle definition than their master object type	OWL inheritance of classes: subclasses inherit properties (including associated event types) from their superclasses.
Contract rule: when two or more object types participate in the same event, a common existence dependent object (contract) is required that participates in this event.	A *commitment* (incremental or decremental) relates exactly one agent to exactly one resource. (note that a reciprocal pair of an incremental and decremental commitment is equal to a full commitment or

	contract between two agents to exchange one resource)
In the OET, propagated participations are marked as “Acquired”, else as “Owned”	Inherited properties in OWL are explicitly indicated as inherited.
A creating event type for a dependent class is a creating or a modifying event type for the master class	When in OWL an instance of a class with associations to other instances (dependencies) is created, by the *inverse property* of such dependencies, the other instances automatically also are modified to relate to this new instance.
A modifying event type for a dependent class is also a modifying event type for its master class	In exceptional cases a class can influence another class making use of a SWRL rule
An ending event type for a dependent is an ending or modifying event type for its master	In exceptional cases a class can influence another class making use of a SWRL rule
All object types are only related through associations that express existence dependency	A *commitment* (incremental or decremental) relates an agent with a resource (both called objects in MERODE). This is the only type of existence dependency to be modeled in MERODE.
The cardinality of existence dependency relationships is defined (how many occurrences of the dependent object type can be dependent of one master object at one point in time)	OWL natively includes existential and universal restrictions in associations. Also, an exact number, or range of numbers, can be specified as association restriction.
An event is defined as an atomic unit of action that represents something that happens in the real world.	An event is either an incremental or decremental event, which corresponds with exactly one in- or outflow of a resource to or from exactly one agent. This is also defined as an atomic unit of action. In practice, every event defined with the MOSES technique maps to two REA events (one incremental and one decremental event).
An event reflects how domain objects come into existence (creating events), are modified (modifying events) and disappear from the universe of discourse (ending events).	An event is either an incremental event that can create a resource (an object in MERODE) (comes into existence) or receives an existing resource from another agent (modifies the owner of the resource), or it is a decremental event that can delete a resource (e.g. by consuming the resource) or sends an existing resource to another agent (modifies the owner of the resource).
Object-event participations	An (inflow or outflow) *stockflow* defines the relationship between a resource (an object in MERODE) and an event. A (provide or receive) *participation* defines the relationship between an agent (an object in MERODE) and an event. *Stockflow* and *participation* cover all object-event participations possible in MERODE.
Participating objects are assumed to concurrently execute all their methods when an event is triggered	When agents have a *partner* relationship with commitment, the events related to this commitment (*fulfil* relationship) need to be executed concurrently

Table 8: Mapping of MERODE elements to REA and OWL elements

No.	Class	Restriction
1	REA-element	If a REA-element is associated with one or more REA-elements, all these elements need to exist before this first element can be created.
2	ResourceType	If an instance of this class exists, then also at least one incremental event should exist that has a stock-inflow with this class and also at least one decremental event should exist that has a stock-outflow with this class.
3	AgentType	If an instance of this class exists, then also at least one incremental event should exist that has a receive participation with this class and also at least one decremental event should exist that has a provide participation with this class.

Table 9: Overview of additional OWL restrictions supporting MERODE

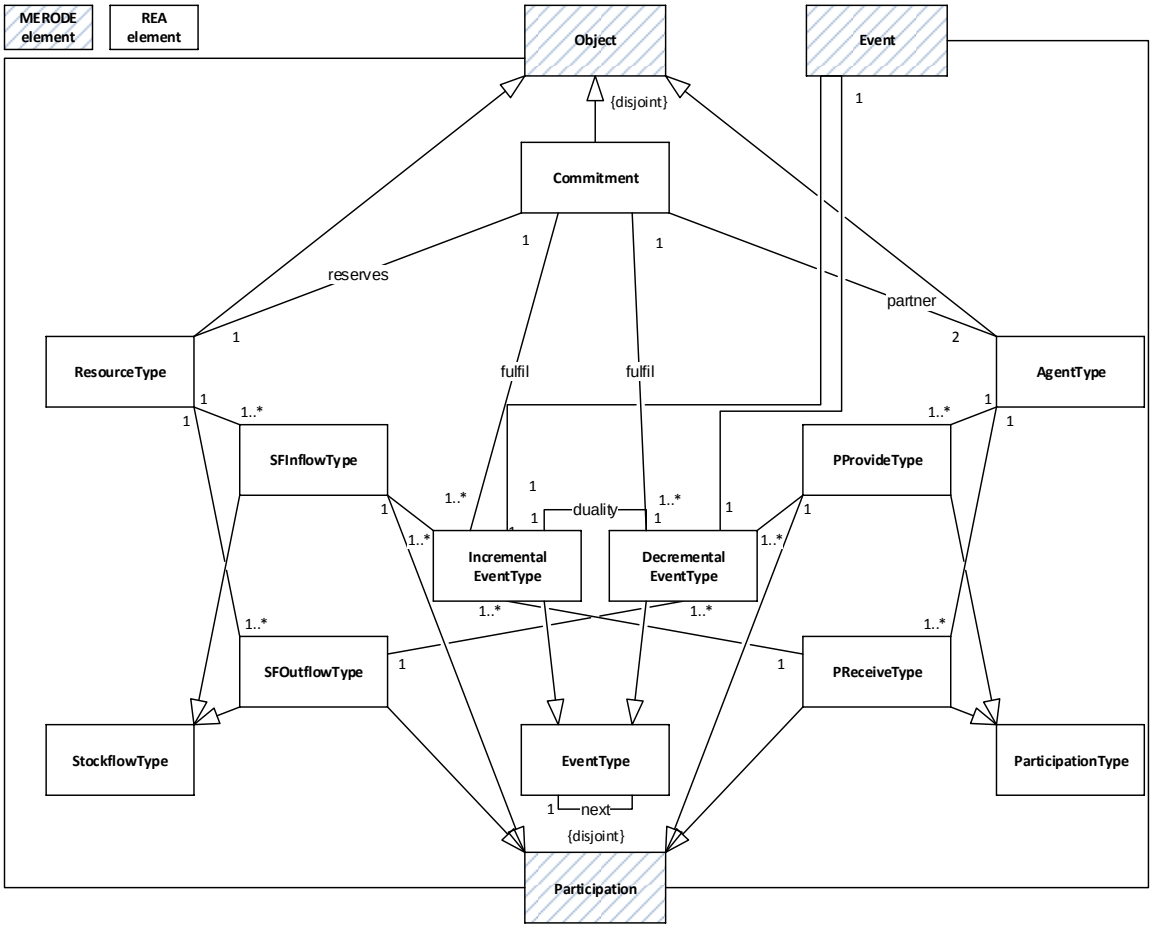


Figure 14: Mapping REA elements to MERODE elements

4.2.2.1 IMPORT AND LINK AGENTS AND RESOURCES TO REA-ONTOLOGY

By transforming the objects from the business domain model to the (REA) ontology model, the base for further specifying the properties and behavior of things in the domain is created. This allows more functionality compared with the specification of a traditional information model (Falbo et al., 2002). It should be noted that the ontology adopts the open world assumption, which means the modeled concepts do not restrict its interpretation of the domain (Aßmann et al., 2006).

As shown in Figure 14, the resources, agents and commitments can be extracted from the MERODE objects. The participations between these MERODE objects can be derived to REA stockflows (either an inflow or outflow of a resource) and participations (either a providing or receiving participation of an actor). An event in

MERODE itself has to be mapped on two separate REA events, as REA distinguishes between the incremental and decremental part of a (transaction) event in MERODE. Table 10 shows this mapping in an overview table.

MERODE element	REA element
Object	Resource or Agent or Commitment
Participation	Stockflow inflow or Stockflow outflow or Provide participation or Receive participation
Event	Incremental event and Decremental event

Table 10: Table overview of MERODE elements mapping to REA elements

4.2.2.2 IMPORT AND LINK COMMITMENTS AND EVENTS TO REA-ONTOLOGY

The same goes for the transformation of commitments and events to the (REA) ontology model. Figure 14 shows commitments can be extracted from the MERODE objects and event types need to be extracted from MERODE events. For the latter it is important to pay attention to the ontology class relationship properties and the relationship cardinality. Events in MERODE need to be expressed in incremental and decremental events, linking to the commitments they fulfil. We are only interested in the modification events of the commitment, as the starting and ending of a commitment is not covered by the REA ontology.

In the OE-Table an event has modification methods relating the event to the involved actors and resources. In the REA ontology stockflow types need to be defined for the related resource (an inflow stockflow type relating the resource with the incremental event type and an outflow stockflow type relating the resource with the decremental event type). Participation types need to be defined for the related agents; a provide participation type for the providing agent in the related event and a receive participation type for the receiving agent of the event. Figure 15 shows a sample OE-Table, where agents “Agent1” and “Agent2” have a commitment “Commitment1” to exchange resource “Resource1”. On the figure is indicated which method leads to which concept in REA.

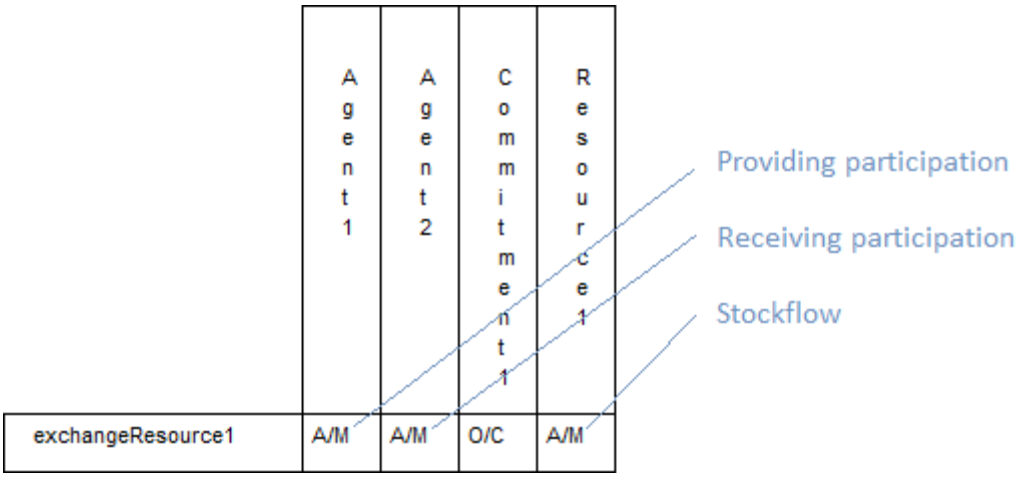


Figure 15: Sample OE-Table indicating relationships with event

4.2.3 DEVELOP ONTOLOGY

Elaborating on the ontology base, this phase adds more detail and other domain knowledge to the ontology. Also in this phase it is important to strive for high interoperability potential by modeling the ontology components in a decoupled, decentralized and configurable way (Daclin et al., 2006). To ensure high interoperability between all stakeholders, it is important to involve different types of participants from all stakeholders in the steps of this phase (H.Sofia Pinto et al., 2009).

4.2.3.1 REUSE AND INTEGRATE EXISTING ONTOLOGIES

In this phase the domain model is more or less formalized and elaborated upon. When this happens, it should be taken into careful consideration to check whether parts of the ontology to be formalized already are specified in another related existing ontology (Oscar Corcho et al., 2003; Fernández-López et al., 1997). This way unnecessary work is avoided and the ontology will be consistent with other systems also using this existing ontology. This way the ambiguity of concepts is further reduced in favor of interoperability. The reusing and integrating of existing ontologies should happen in parallel with the other development steps in this phase.

4.2.3.2 DETERMINE PROPERTIES OF AGENTS AND RESOURCES

The agents and resources of the ontology only have names and relations to events specified after the transformation in the previous two steps. Therefore, to add specifications to the agents and resources, properties need to be created in the ontology model. When performing this step, only the currently relevant relationships add properties should be included, as the ontology model falls under the open world assumption (Aßmann et al., 2006). To distinguish between properties used for reasoning purposes only and properties that also need to be included in the information to be exchanged, the latter category of properties needs to be added as sub-properties of the general “message” property of the REA upper ontology.

Additional information on the domain can be acquired from domain experts and literature, but also from the domain model specified in the first phase. According to Noy & McGuinness (2001) it may occur that multiple modeling options can be applied. In this case, Noy & McGuinness (2001) advise to choose the modeling option that facilitates anticipated model extensions.

For the development of the properties, it is recommended to formulate these in the best way facilitating reasoning about classes and sets and subsets of classes and their instances (Henderson-Sellers, 2011), which is something that can improve interoperability communication. Next to that it is important to have the concepts describing the domain instead of prescribing behavior of a computer system to ensure the ontology remains solution-independent (Aßmann et al., 2006). The descriptions need to be as concise and unambiguous as possible to prevent misinterpretations (Falbo et al., 2002). Also eminent is to ensure all terms specified to be partial complete (i.e. coverage of the terms and the right level of granularity of each term), concise and consistent (Fernández-López et al., 1997). Furthermore it needs to be taken into account that all the stakeholders need to be able to work with and understand the concepts of the ontology (Aßmann et al., 2006).

4.2.3.3 DETERMINE INFORMAL CONSTRAINTS

Constraints, either formal or informal, have the task to limit the instantiations of the developed ontology model only to configurations that are realistic for the represented concepts of the domain concerned (Gasevic & Djuric, 2006). This step lists all these constraints. By first defining all constraints in a semi-informal language, the chance of loss of domain knowledge during the constraint elicitation from domain experts becomes smaller (Helena Sofia Pinto & Martins, 2004).

4.2.3.4 DETERMINE FORMAL CONSTRAINTS

After all informal constraints have been recorded for domain knowledge preservation reasons (Gasevic & Djuric, 2006), these are formalized in the ontology. This way the ontology can automatically reason what configurations are valid and invalid. This step cannot be automatized, as semi-informal constraints can be formalized in many different ways; it needs to be done manually.

The ontology development tool we use supports the restriction language SWRL (Semantic Web Rule Language) well and is very intuitive in its use (Nguyen, 2011). Also, SWRL is designed as the standard restriction language for ontologies in the semantic web. Therefore it is used as language to formalize the constraints. Although its syntax is easy to understand, the rest of this thesis it is assumed the reader knows about the SWRL syntax. More information about SWRL and its syntax can be found in (Horrocks et al., 2004).

4.2.3.5 DETERMINE CLASS INSTANCES

At this point in the process the ontology should comprise of all types of classes possible in the domain. This should be enough information to develop a general semantic standard for all general types of the concepts defined. To add some more specific information of special cases, this step creates instances of the class types in the ontology that are present in the real world domain. For each instance individual properties need to be set, as predefined in the class types. Also, subsets of individuals can be created as another instance. For example, a select number of persons belong to one group of friends, where the group of friends is a subset of persons. Specific constraints for certain instances can also be determined. These can be included in the iteration cycles of this development phase.

In this step it is important to take into account that the instances modeled are only describing the real world domain and do not prescribe it (Aßmann et al., 2006). Like with the class types, the instances describe dynamic behavior in the domain. Instances need to specialize on the exact events of specific agents, so each stakeholder is aware of the specific dynamic behavior and the corresponding operation rules of each domain instance (Panetto & Molina, 2008).

4.2.4 DETERMINE TECHNOLOGY-SPECIFIC SOLUTION

As the ontology includes both the dynamic and static elements of the domain concepts, it facilitates the transformation from the model to a technology-specific solution. The final result of this last methodology step is a semantic standard, which in most cases is no more than a set of message structures in a specified sequential order that is used as a standard for the exchange of domain information (Schrier et al., 2012).

As the used ontology language, OWL, is fully XML-based, the information queried can be transformed into a technology-specific solution and/or message diagrams using an XML translation. Languages like XSLT or XPath proved very useful for this type of translations (W3C, 1999a, 1999b). The events determine the sequence of messages to be sent and received, while all sub-properties of “message” are the information that needs to be exchanged. Because of time limitations this thesis will not go into detail in this step.

Using a query language, such as SPARQL (W3C, 2013), the REA-based ontology can be queried for more specific information for the message structures. This information is retrieved from the instances defined in the ontology. Ontology building tool Protégé has built-in support for SPARQL queries.

To get a clear overview of the messages to be sent and received by which agents, sequence diagrams are a good way of visualizing this (Lethbridge & Laganière, 2005). Section 4.3.5 and Figure 18 provide detail in their notation. The foundations for determining message structures from the ontology using the REA upper ontology lay at the events. Sending and receiving information is represented in the REA upper ontology by the

decremental and incremental event types. Here, the decremental event type is defined as sending information (i.e. the information of the resource related to this event) and the incremental event type as receiving information. Because by definition of REA (see also section 2.5.1) incremental and decremental event types that have a duality relationship, exactly the same resource (i.e. information to be exchanged) is related to both events (via stockflows).

The sender of a message is identified by the agent that has a provide participation with the decremental event type involved. The receiver of this message is the agent that has a receive participation with the dual incremental event type of the decremental event type. To indicate the sequence of event execution according to the domain restrictions, if modeled, events refer to the next event happening after the execution of the current event happened. It is possible to have multiple following events. Depending on the situation one of the events appointed as next event will be executed.

The designed ontology can be queried for this information with a SPARQL query as shown in Query 1. Table 11 shows the structure of the output table of this query. The query retrieves the message field that needs to be sent based on all data properties of the resources that are exchanged. In case a real-time instantiation of the domain is maintained in the form of individuals, the actual messages, senders and receivers can also be retrieved by adding "?message", "?sender" and "?receiver" to the output table columns in the SELECT line of the query. While there are much more applications for retrieving information from the developed ontology, this is the main query that needs to be executed to get the required information for transforming to message structures for creating a semantic standard. An example of another application of the ontology information is shown in section 5.2.6.1, where an overview of the congestion impositions on grid participants can be created by querying the information from an ontology developed with this development methodology.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX my: <http://www.rea-ontology.com/rea-smartgrid#>
SELECT ?eventtype ?messagefield ?sendertype ?receivertype ?nexteventtype
WHERE {
        ?decrementalevent rdf:type ?eventtype.
        ?eventtype rdfs:subClassOf* my:DecrementalEventType.
        ?allsfs rdfs:subPropertyOf my:eventHasSf.
        ?decrementalevent ?allsfs ?relatedsfs.
        ?allrelatedresources rdfs:subPropertyOf my:sfHasRes.
        ?relatedsfs ?allrelatedresources ?resource.
        ?messagefield rdfs:subPropertyOf* my:message.
        ?resource ?messagefield ?message.
        ?allparticipations rdfs:subPropertyOf my:evHasP.
        ?decrementalevent ?allparticipations ?participations.
        ?allagents rdfs:subPropertyOf my:pHasA.
        ?participations ?allagents ?sender.
        ?sendertype rdfs:subClassOf my:AgentType.
        ?sender a ?sendertype.
        ?allDualities rdfs:subPropertyOf my:dualEvent.
        ?decrementalevent ?allDualities ?dualevent.
        ?dualparticipations rdfs:subPropertyOf my:evHasP.
        ?dualevent ?dualparticipations ?dualparticipation.
        ?alldualagents rdfs:subPropertyOf my:pHasA.
        ?dualparticipation ?alldualagents ?receiver.
        ?receivertype rdfs:subClassOf my:AgentType.
        ?receiver a ?receivertype.
        ?decrementalevent my:nextEvent ?nextevent.
        ?nexteventtype rdfs:subClassOf* my:EventType.
        ?nextevent a ?nexteventtype.
}
```

Query 1: SPARQL-query retrieving the relevant information from the REA-ontology for creating message structures

Decremental event type	Message field	Sender type	Receiver type	Next event type

Table 11: Structure of the output table of the SPARQL-query for retrieving message structures

4.3 NOTATIONS USED BY THE METHODOLOGY

Several steps in the methodology require the design of a diagram or an overview table. This section specifies how exactly these diagrams and tables should be developed. The following subsections each represent a step that involves the creation of a diagram or table.

4.3.1 IDENTIFY AGENTS AND RESOURCES

To enumerate and give a detailed description of the relevant agents and resources in the business domain, an overview table can be created. Table 12 is a template overview table. In the left column one simple name of the type of agent/resource should be specified, with in the right column an elaborate description of its role, purpose, demands, etc. in natural language.

Agent / resource	Description
Agent1	
Resource1	

Table 12: Template overview table for agents and resources

4.3.2 IDENTIFY COMMITMENTS

The best way to identify commitments is to draw an existence dependency diagram. The identified agents and resources of the previous step can already be drawn, after which commitments can be added to create links between 2 actors and 1 resource type. Figure 16 shows a template existence dependency diagram with on the left 2 agents (“Agent1” and “Agent2”) and on the right 1 resource (“Resource1”). In the center, commitments can be drawn (in the example “Commitment1”) that connects with the agents and resource with existence dependencies. To maintain overview on the existence dependency diagram it is recommended to depict all agents on the left side of the diagram, resources on the right and commitments in the center. For each commitment a short description should be made, to give context and insights into the meaning of the commitments. This can be done in a table formatted like Table 13.

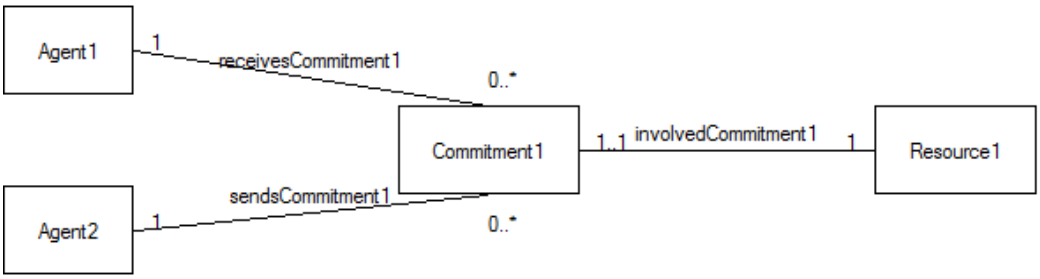


Figure 16: Template existence dependency diagram

Commitment	Description
Commitment1	

Table 13: Template overview table for commitments

4.3.3 IDENTIFY EVENTS

Commitments are defined as agreements between 2 agents to exchange a resource type in the near future. Events occur to fulfil these events. Therefore, for each commitment at least two events should be specified (one incremental and one decremental). This event needs to be linked to the resource type it is involved with and to the agent type that participates in this event. By creating an Object-Event-Table (OE-Table), the identification of events and their related objects is supported.

There are some rules the events have to respect, which are defined by MERODE, enumerated in section 2.4.1. The relationships that can be specified in an OE-Table are listed in Table 15. Table 14 shows an OE-Table template. For the ease of explanation, “Event1”, “Agent1”, “Commitment1” and “Resource1” represent the location of the events, agents, commitments and resources respectively in the table. Commitments that are a subclass of another commitment are depicted in the way shown as SubComm.1 is subclass of Commitment 2 in the example. The same goes for events; as depicted in the way shown as SubEvent1 is subclass of Event2 in the example. When there exists an interaction between an event and an agent or resource, their junctions in the table should have one or more letters. These relationship types are listed in Table 15. A tool like MERMAID can be used to facilitate the development of an OE-Table (see also section 4.2.1).

	Agent1		Commitment1	Commitment2	SubComm.1		Resource1	
Event1								
Event2								
SubEvent1								

Table 14: Template AOET

Letter	Relationship type
I	Internal actor
R	Responsible actor
P	(Only) participating actor
C	Object creation
E	Object ending
M	Object modifying (updating)
O	Owned participation in event
A	Acquired participation in event from superclass

Table 15: Relationship types for the AOET

4.3.4 MAKE UML ACTIVITY DIAGRAM

In a domain events occur only in a predefined order. To ensure these event sequences are respected in the domain model, MOSES uses Jackson Event Sequence diagrams (Schrier et al., 2012). An alternative to this notation is a UML activity diagram. This diagram type is more frequently used and has the same functionality as a Jackson Sequence diagram.

For each sequence of events that happen in the domain, an activity diagram can be drawn (Lethbridge & Laganière, 2005). Figure 17 shows an example activity diagram with all possible elements. A black circle represents the start state, which immediately induces the followed activity (in the example Event A). A black circle with a ring around it represents an end state, which not necessarily is ever reached. If it is reached, the sequence of events is over. Rectangles represent the activities or events themselves.

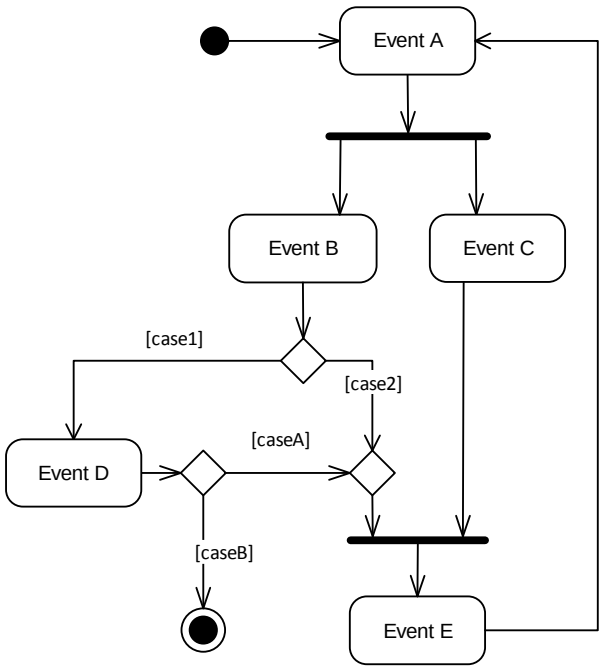


Figure 17: Example UML activity diagram

Two events can be performed at the same time after a fork, which is a horizontal line with one incoming transition and multiple outgoing transitions (in the example between Event A, B and C). A join is the same, but with one outgoing transition and multiple incoming transitions (in the example before Event E). The outgoing transition is triggered only when all incoming transactions have been triggered. A combination of both is called a rendezvous. Based on a decision, a decision node can trigger one of multiple outgoing transitions (in the example, after Event B, Event D can be triggered when case1 is true; if case B is true after Event D, the sequence ends. Split transitions paths by a decision node can be merged again by a merge node.

4.3.5 SEQUENCE DIAGRAMS

Determining the messages, their sender and receiver and their sequence order is essential for the last step of the methodology, where the message structures of the semantic standard are determined. UML sequence diagrams prove useful for gaining insights in all this (Bell, 2004; Lethbridge & Laganière, 2005). The notation of these diagrams are relatively simple.

In Figure 18 an example sequence diagram is shown. The rectangles with sender and receiver in them represent the agents. In specific, this example depicts an instance of AgentType called sender and another instance of AgentType called receiver. These instances have a lifeline, depicted by the dashed vertical line connected to them. The start of the sequence order is defined as the top of this lifeline, where the end of the sequence order is at the bottom of the lifeline.

When an agent shows activity, a rectangle is drawn around the lifeline. During the activity of an agent, messages can be sent and received. The arrows drawn between the two lifelines represent the direction in which messages are sent. For example “message1" is sent from “sender” to “receiver”. After this message has been received, “receiver” sends “message2” to “sender”; probably a response to “message1”. It is recommended to draw a separate diagram for each set of interactions between two agents.

A sequence diagram can also model alternative cases. The set of alternatives need to be framed by a box labeled “alt”. Figure 18 shows an example of this, where “message3” and “message4” are exchanged when the conditions of “case1” apply and where “message5” and “message6” are exchanged when the conditions of “case2” apply. While not pictured in the example, loops of message sequences can be modeled by drawing a similar frame with the label “loop” on it.

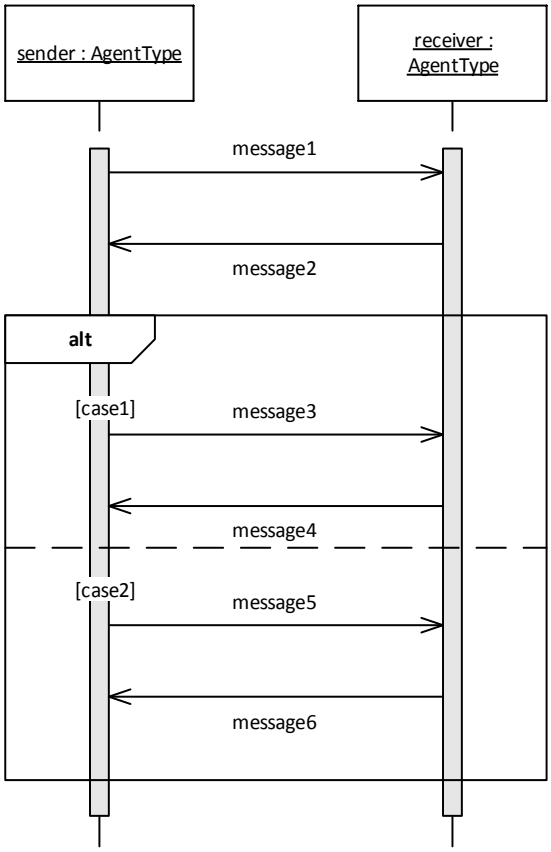


Figure 18: Example UML sequence diagram

Because the message sequences to be modeled are deducted from an ontology based on the REA upper ontology, all sequence diagrams can be deducted using the result of the SPARQL-query as defined in Query 1, with a result in the form of Table 11. This means all information about the instances of the built domain ontology is retrieved related to all “DecrementalEventTypes” and their related “message”, “sender”, “receiver” and “next: EventType”. The last piece of information indicates the next event/message involved with the current event.

Combining this information, sequence diagrams can be drawn showing the message exchanges. Figure 19 gives insights in how this information can be translated into a sequence diagram. Currently the translations need to be done by hand, but this is easily automated by writing several translations from OWL to a document type supporting the drawing of sequence diagrams.

The “sender” of a “message” has a “DecrementalEventType” that initiates the sending. The “receiver” has an “IncrementalEventType” connected to this “message”. To indicate what is the next EventType (either receiving or sending a message), the “next: EventType” plays a role.

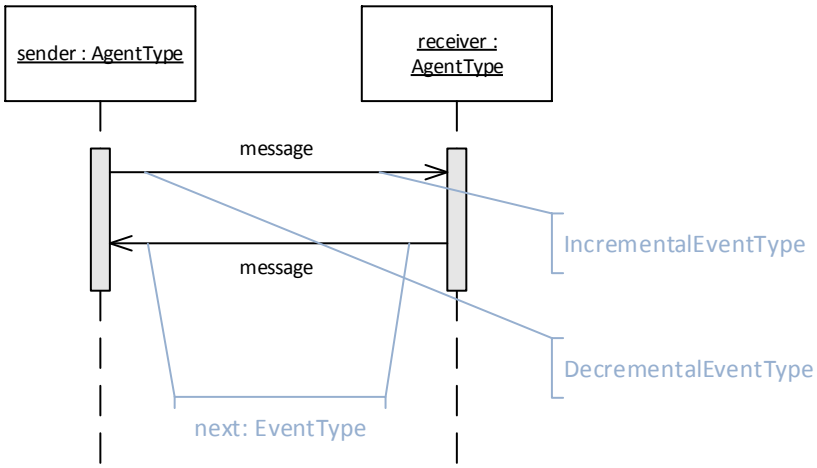


Figure 19: Example UML sequence diagram with REA elements

5 AN ONTOLOGY FOR SMART GRIDS

To demonstrate the new methodology, it will be applied on a theoretical case about the energy domain. As shortly introduced in chapter 2, the smart grid operability problem will be taken as a use case. The chapter elaborates on the philosophy to be used using the method, the methodology steps taken as well as the notations used to reach the technology-specific solution. This chapter starts with an overview of the energy sector, focused on the situation of smart grids in this sector. After that the designed methodology will be applied on the theoretical case of a micro grid. At the end, the application of the methodology will be discussed.

5.1 THE MICROGRID DOMAIN

Insights in the general structure of the energy sector are required to adequately apply the methodology on smart grids. This section will first elaborate on the most important recent developments in the energy sector, followed by an overview of the actors involved in the sector. After that follows an analysis of the concept of flexibility in the energy sector, which we want to address in particularly in this case. At the end of the section literature is reviewed about making the energy sector more flexible and what information models already exist for the energy sector.

5.1.1 WHAT IS A MICROGRID?

In conventional power systems, system status and control signals are exchanged using Supervisory Control and Data Acquisition (SCADA) systems (Pipattanasomporn, Feroze, & Rahman, 2009). These systems, however, require continuous monitoring activities by a human to intervene and take actions when necessary (i.e. the system is reactive). To automatically reason and take actions on power systems, a smart grid can be introduced. This is an autonomous system, meaning it can operate without human interventions and can even respond to emerging problems before humans could detect them (proactive system). A microgrid is nothing more than a smart grid on a local level; what a smart grid can do on high scale, a microgrid can do on a local level, which makes it easier to implement.

As mentioned earlier, a smart grid (and therefore also a microgrid) can be introduced for maintaining the balance on energy grids in the future, when more and more intermittent energy sources (especially renewable energy sources) will be connected to the grid. A smart grid can be defined as an electricity network that intelligently integrates the actions of all users connected to it (Ardito, Procaccianti, Menga, & Morisio, 2012). It improves the reliability, security and efficiency of a power grid by using “smart” technologies for intelligent monitoring, controlling, communicating and self-healing.

The current energy grid is designed and built decades ago and does not facilitate the new developments in the energy domain, as stated earlier. For example, currently grid operators have to continuously monitor the energy grid and have to perform changes in the operation of the grid manually (Bakker, 2012). For these reasons the design and control over the grid should be adapted to this new situation. The grid should make more use of ICT systems that help better matching energy demand and supply and facilitating more renewable energy sources while maintaining a dependable grid. Bakker states that a more robust and fault tolerant grid can be achieved, when a grid can continuously monitor and manage energy flows and energy quality parameters.

Tabors et al. (2010) define three “pillars”, or properties, a smart grid should have: (1) smart customer: the technologies enabling consumers to observe and control their consumption, (2) smart utility: the utilities implementing functionality for monitoring, controlling, pricing and demand-response systems and (3) smart

market: a market structure facilitating decision making and information on the energy of the smart grid. Real-time pricing (RTP) can be seen as a tool to realize this market.

One of the important aspects of a smart grid is to provide a smart metering infrastructure (Ardito et al., 2012). Information on when and how energy is consumed plays an important role for energy generators in the smart grid to allow automatic configuration of the network when a fault emerges somewhere in the network, utility billing and other applications. Also, forecasting is a key aspect of a smart grid system. It is used for ensuring the balance in energy demand and supply on the grid. Next to (traditional) day-ahead energy use profile predictions, the amount of power produced by renewable energy sources can also be predicted by using various parameters, like climate conditions and energy source location.

The need for information integration and communication is stressed by Ardito et al. (2012) for a successful realization of a smart grid. They argue that energy generators need a constant flow of real-time updates on energy demands in order to provide the precise amount of energy required. The authors also argue that an energy storage can substantially improve the efficiency of a smart grid. In cases when energy overproduction happens, the storage can be used to absorb this production peak . When underproduction occurs, the storage can fill the energy gap.

An ontology could be of good use in a smart grid. King (2008) states that an ontology “provides the glue to hold everything together” and that it drives decision support tools for the electric grid by standardizing the semantics and developing taxonomies.

5.1.2 TRENDS

Not only organizations, but also individuals are increasingly participating in the complex business network of the energy market. They install renewable energy sources like solar panels on their roof, or wind turbines in their neighborhood to provide themselves with their own energy supply and, more importantly, sell the remainder of the generated power to the rest of the power grid. Consumers are turning into so-called “prosumers”. The use of renewable energy sources currently is a big trend that is also enforced and encouraged by national and international regulations (Jamasb & Pollitt, 2005). Drivers for these policies include the mitigation of climate changes due to emission of greenhouse gasses, reducing the dependency on fossil fuel reserves and liberalizing and improving the energy market (European Commission, 2006).

The energy sector in the Netherlands, as well as other countries of the European Union is undergoing a change towards more use of renewable energy sources. In the past two decades the production and consumption of renewable energy rapidly increased. Jacobsson & Bergek (2004) indicate that in the Netherlands, among other countries in the EU, the diffusion of wind turbines increased (from 49 MW in 1990 to 519 MW in 2001), solar cells (from 1.3 MW in 1990 to 12.8 MW in 2001) and solar collectors (from 11 GW in 1990 to 226 GW in 2001). Also, Energie-Nederland & Netbeheer Nederland (2011) indicate an increase of total renewable energy generation in the Netherlands from around 3,000 GWh in 2001 to over 10,000 GWh in 2010.

The disadvantage of renewable energy sources is that their energy supply is on irregular basis (intermittent). Energy is only produced when certain meteorological conditions apply, e.g. a windmill only produces energy when there is sufficient wind. When and how much energy is supplied is therefore uncertain. To keep energy networks up and running, it is of vital importance to maintain the balance between the amount of energy consumed and supplied. When this uncertainty of power supply emerges, it becomes difficult to maintain this balance in energy networks because manually controlled power supplies need to adapt to these, uncertain, power supply changes. This leads to a need for mitigation possibilities for the supply of energy to guarantee the essential balance on the energy network.

Also, the political push to unbundle the energy market is causing a decentralization of the energy market (European Commission, 2005; Jamasb & Pollitt, 2005). This enables prosumers (i.e. stakeholders that consume as well as produce energy) to offer their energy surplus to the rest of the energy grid (Pagani & Aiello, 2011). Because of this change in the energy market, more and more stakeholders need to have interaction with the power grid. This results in the need for interfacing capabilities for equipment connected to the power grid (European Commission, 2006). These interfacing capabilities ensure the interoperability of automation and control arrangements of the equipment connected to the grid, like power generators, metering systems or control and automation architectures.

Next to that, Kok et al. (2010) note that there is an increase of distributed generators (DG, i.e. generation capacity in the medium- and low-voltage segments of the power grid). This leads to more power generators that generate relatively small amounts of energy. This, in turn, leads to the increased importance of intelligent distributed coordination of the power grid. The European Commission (2006) foresees a trend towards microgrids. They define microgrids as "low voltage networks with DG sources, together with local storage devices and controllable loads (e.g. water heaters and air conditioning)." Microgrids can automatically be set to islanded mode when necessary, for example in case of a fault in the upstream network, and can be resynchronized with the rest of the grid after the fault has been restored. Because microgrids include local energy storage devices (DGs), the energy network (in the microgrid) can be balanced by drawing from or storing small amounts of energy on these devices. Also, a microgrid can supply energy back to the rest of the energy grid. Currently, energy grids are not fully prepared to support this yet. Figure 20 shows an example of how a microgrid looks like. Houses with solar panels (i.e. the prosumers), some energy storages and a distributed generator are part of the microgrid. The microgrid shows two connections to the rest of the energy grid, from where, if necessary, power can be drawn from or supplied to.

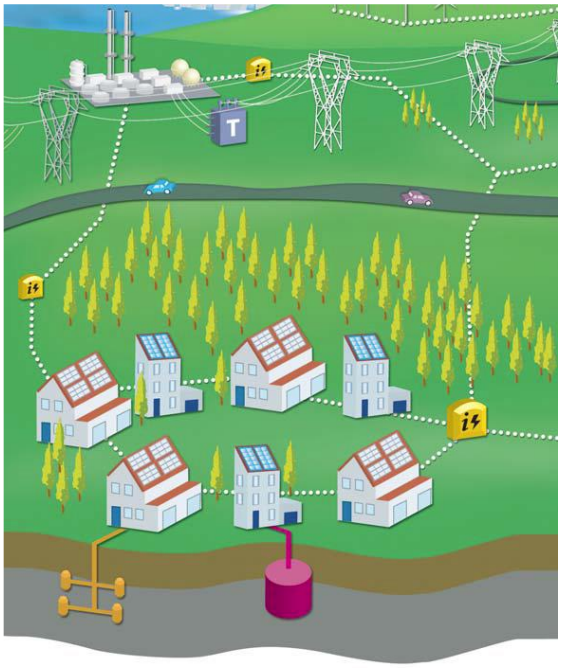


Figure 20: Example of a microgrid (European Commission, 2006)

5.1.3 OVERVIEW OF ACTORS

Different authors describe actors of the energy sector in more or less detail. This section provides an overview that integrates as much as possible relevant actors for supporting information exchange between them to facilitate a microgrid. An overview of the actors and their dependencies is given in Figure 21. The lines between the actors represent their interactions, which can be either the exchange of information or the physical exchange of energy in form of electricity. A description of these actors follows (Bakker, 2012; Energie-Nederland & Netbeheer Nederland, 2011; European Commission, 2006; european network of transmission system operators for electricity, 2011; J. K. Kok et al., 2010; Pagani & Aiello, 2011; Smart Grid Coordination Group, 2012). It should be noted that in the current situation of the energy domain more actors are involved, such as suppliers, traders and an energy trading market, but these are out of scope for the microgrid domain.

Energy producers: responsible for the generation of energy on large scale. Also includes the distributed generation operators that involve distributed energy generation on smaller scale. Large energy producers are typically inflexible in adjusting their production output on short term, while smaller energy producers are more flexible although the production costs are higher.

Transmission System Operator (TSO): represents the infrastructure and organization which transports electricity over long distances. It controls the high voltage energy network and facilitates the connection to energy networks abroad and the energy import. In the Netherlands TenneT is the national transmission system operator, which is a government-owned organization.

Distribution System Operator (DSO): represents the infrastructure and organization which distributes electricity to customers. It controls the medium and low voltage distribution networks. Receives the meter readings from the metering responsibles and passes these on to the suppliers and balance responsables.

Distributed Electrical Resources (DER): represents small-scale power generation technologies (typically in the range of 3 - 10.000 kW) directly connected to the medium and low voltage distribution networks. Typically, DSOs control these resources.

Customer: the end-user of the energy. The customer can be industrial, commercial or a home facility. It is expected that in the near future an increasing number of customers will want a connection to the grid that supports in-house generation and the ability to sell their surplus energy generation back to the grid. Most authors see this development only in the future, while we see customers, or prosumers, currently already transporting and selling their energy surplus to the rest of the energy market.

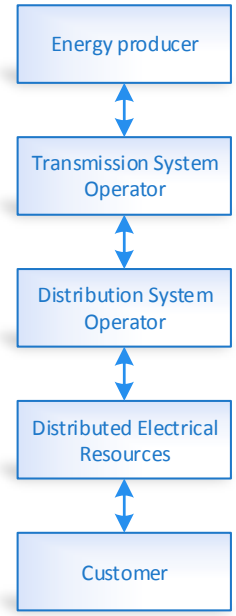


Figure 21: Overview of actors relevant for a microgrid

5.1.4 MICROGRID INTEROPERABILITY

The technology in the energy sector can be divided in three layers (Pagani & Aiello, 2011). The lowest layer is the *physical layer* (or, technical layer), which directly interacts with electrical apparatus and other physical components of power plants and distribution substations. The second layer, called the *data layer* (or, informational layer), contains control data that remotely supervises and interacts with physical equipment. Therefore, this data can control energy production, transmission and distribution in the energy network. The *business layer* (or, organizational layer) is the third layer, which uses information from the data layer to form business information. This information is used for indicating the current performance in the energy network, to make forecasts and to control the network.

The GridWise Architecture Council (GWAC) extends this model by allocating eight interoperability categories to these 3 layers, as shown in Figure 22 (Smart Grid Coordination Group, 2012). The emphasis of the first category, *basic connectivity*, lies at the physical distribution of all participating components in the smart grid context. *Network interoperability* and *syntactic interoperability* comprise of protocols and mechanisms for exchanging information between smart grid components. The information exchanged itself resides in GWAC categories 4 and 5. *Business context* facilitates interoperability in information on a high level, where the *semantic understanding* category provides a shared understanding of the information exchanged in the smart grid domain. *Business procedures* describe functions and services in the smart grid and their relationships with the information defined on the categories below it. *Business objectives* and *Economic/regulatory policy*

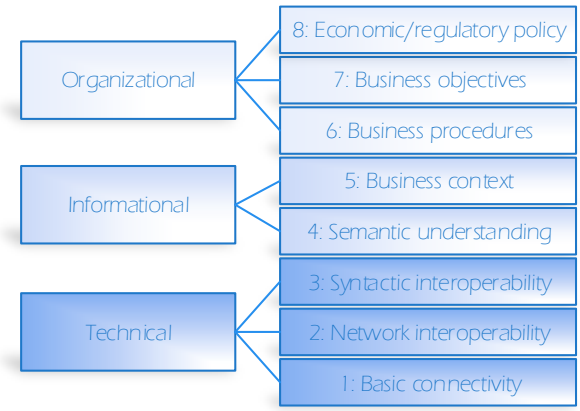


Figure 22: GWAC interoperability categories (Smart Grid Coordination Group, 2012)

represent the business view on the information exchanges occurring in a smart grid. For example regulatory markets and policies can be defined on this level.

The improved MOSES methodology to be carried out in this case has the goal to create an interoperability standard that strives towards semantic understanding between all domain stakeholders. Therefore, the methodology focuses on GWAC category 4 (semantic understanding), although to completely develop semantic understanding in the microgrid domain knowledge about the events and other dynamic behavior in the domain is required (see also section 4.2.1.3 about the mindset behind modeling business events).

5.1.5 FLEXIBILITY IN ENERGY DEMAND AND SUPPLY

The identified trend towards the large use of renewable energy sources increased the need for coping with flexibility in energy demand and supply. Specifically, flexibility is needed in the amount of energy produced and consumed and flexibility in the moment of the energy production or consumption. There are several approaches to achieve this flexibility (Verhoosel, Rothengatter, Rumph, & Konsman, 2012):

One approach is to hand over some control over one or more devices (e.g. a large energy consuming device like an air conditioner) by a third party in return for some discount for the consumer. When it is better for the energy network, the device can be switched off or turned to a lower energy consuming mode. A second approach is to use dynamic pricing to elicit some demand-response behavior, whether or not automated, to influence the behavior of energy usage of the consumer. Another approach is to enable the trading of energy in smaller volumes. This can be an effective means to adapt energy demand to the intermittent supply, as shown by the PowerMatcher concept, which is a multi-agent system that facilitates this energy trading (K. Kok et al., 2008).

The approach taken by Verhoosel et al. (2012) is called the flex-offer approach. This approach lets participants explicitly specify the flexibility they are willing to offer other parties on the energy market themselves. This is in close relation with the three main pillars of smart grids as defined by Tabors et al. (2010) (Smart customer (demand), smart utility (supply) and smart market (platform provider)). In this case we will also take this perspective. Flexibilities in energy demand and supply are deemed to be achieved by temporal shifts of energy consumption, temporary reductions in energy load and the spread of energy load over time. This combines the possibility for fine grained control by the party buying flexibility with the autonomy of the party selling the energy that stays unaffected. A disadvantage of this approach is the introduction of extra complexity, because all flexibility constraints set need to be satisfied.

Next to flexibility in demand, we see possibilities to make use of the flexibility in supply as well. Especially for renewable energy sources that currently generate electricity when they can (e.g. when the sun is shining, or when the wind is blowing), in some cases the extra energy produced is not counted on and could potentially cause an imbalance on the energy network. In these special cases it should also be possible to turn off the energy generation of renewable energy sources.

The embedding of energy resource flexibility is expected to have several benefits to the microgrid itself and its other connecting power grids (CEN/CENELEC/SG-CG/M490/E, 2012). A benefit is that the microgrid can be assured of voltage frequency stability by preventing large voltage deviations between the actual and scheduled power in a grid. Next to that, resource flexibility results in a higher safety against grid capacity exceeding. Also, by using available flexibility a network can be restored more easily after a failure, or even facilitate a black start after a complete blackout.

5.1.6 SMART GRID INFORMATION MODELS

In order to achieve flexibility in energy demand and supply in a power grid, information can be exchanged between the involved parties using energy information models. This section investigates the current state-of-the-art of these information models used. Table 16 shows an overview of the information models reviewed. It appears that all, except the MIRABEL model and the upper ontology for power engineering applications, do not include an ontology.

Information model	Author	Includes ontology	Goal
Common Information Model (CIM)	IEC	No	Drive interface and data exchange design
OASIS Energy Interoperation	OASIS Open	No	Interoperable and standard exchange of dynamic price, reliability and emergency signals, bids, load predictability and enterprise interaction with energy markets
Facility Smart Grid Information Standard	NIST	No	Enabling energy consuming devices and control systems to manage energy loads and energy sources in response to communications with the smart grid
MIRABEL	Verhoosel et al. (2012)	Yes	Enabling flexibility in energy loads in a given timeframe
PowerMatcher	J. K. Kok, Warmer, & Kamphuis (2005)	No	Market-based control of an energy network by matching energy supply and demand
Upper ontology for power engineering applications	Catterson, Baker, Davidson, & McArthur (2010)	Yes	Monitoring the condition of power systems

Table 16: Overview of the energy information models reviewed

The *Common Information Model (CIM)* is a widely accepted energy information model that includes vendors and customers from around the world (Crapo et al., 2009). It is adopted by the International Electrotechnical Commission (IEC) for power transmission and distribution (Simmins, 2011). The main objectives of this model is to develop a platform independent data model for enabling better smart grid interoperability and to define a set of standards describing a "framework for energy market communications" (Uslar, Specht, Rohjans, Trefke, & Vasquez González, 2012). These market communications include the exchange of information between market participants and market operators as well as communication between market operators.

The requirements of data exchanges are documented and maintained in a UML model, which is mapped to existing other CIM UML models. Simmins (2011), on the other hand, states that next to this UML model the CIM defines a common vocabulary and basic ontology for aspects of the electric power industry. The CIM is one of the priority action plans of the Smart Grid Interoperability Panel (SGIP) of the National Institute of Standards and Technology (NIST) in the United States, which has the goal to support the development of smart grids (National Institute of Standards and Technology, 2012a).

The CIM is not implemented in an ontology model yet, while several authors attempted to map and transform the information models of the CIM to ontology models. Majewska, Kryza, & Kitowski (2007) developed a 1-on-1-mapping of the modeling concepts to OWL concepts. They argue that high simplification of the CIM is currently required in order to perform this 1-on-1-mapping, which, in fact, removes most of the reasoning behind the model. Quirolgico, Assis, Westerinen, Baskey, & Stokes (2004) reasoned the CIM classes can be

transformed by making use of mappings between the information model classes and ontology classes, but reasoned the mappings themselves are currently too simplified to map all knowledge from the CIM to an ontology variant of the CIM. More research on this mapping is therefore required.

OASIS Energy Interoperation (OASIS Open, 2012a) is an information model that enables collaborative and transactive use of energy in a power network and defines XML vocabularies for the interoperable and standard exchange of information supporting the functioning of an energy grid. These include information about energy prices and bids (demand and response), network reliability and emergency signals and the prediction of loads. This information is derived from the "WS-Calendar" and "EMIX" specification. The first defines how to specify and communicate the duration and time of a schedule. The Energy market Information Exchange (EMIX) specifies the semantics of energy markets. It specifies in an information model and an XML vocabulary the definitions of price and products in transitive energy markets (OASIS Open, 2012b). This model facilitates the price communications and negotiations in a smart grid market, which include information on prices, bids, time for use or availability, units, quantity to be traded and characteristics of what is traded (Cox & Considine, 2011). The WS-Calendar, EMIX and OASIS Energy Interoperation are part of priority action plans set by the SGIP (National Institute of Standards and Technology, 2012a).

At the moment, the *Facility Smart Grid Information Standard* is being developed (National Institute of Standards and Technology, 2012b). The purpose of this standard is to enable energy consuming devices and control systems in the customer premises to manage electrical loads and energy sources in response to communications with the smart grid. The information models to be created are intended to provide a common basis to describe, manage and communicate information on aggregate electrical energy consumption and forecasts. In the end an abstract, object-oriented information model will be defined supporting a wide range of energy management applications and electrical service provider interactions. This standard is also part of a priority action plan (PAP17) set by the SGIP (National Institute of Standards and Technology, 2012a).

Specifically to enable flexibility in an energy grid the *MIRABEL* system has been developed (Verhoosel et al., 2012). This system introduces offers on energy supply and demand that incorporate power profile flexibilities. Consumers as well as producers of energy can use these so-called flex-offers to specify their own flexibility in energy amount offered or required over a given timeframe. Verhoosel et al. (2012) developed their own ontology of the energy domain to fit in their flex-offers. The energy domain is expressed in five main classes: device, actor, energy profile, constraint and flex-offer. Here, a device is an energy consuming or producing device that has a specific energy load over a certain time span (energy profile). Actors have minimum or maximum demands (constraints) on their energy load, price and time. These constraints are issued by an actor for the devices that are owned by the actor. By developing this ontology, the authors try to aid in the conceptualization of the domain semantics and the development of a shared understanding of the domain. This adds value for the smart grid domain, because at the moment tools and components are distributed and their interfaces have to be standardized in order to work in a plug-and-play concept. Also, the use of an ontology is expected to improve the amount of intelligence, security, maintainability, testability, management and development in the smart grid domain.

PowerMatcher is a software framework that entails market-based control of an energy network by matching energy supply and demand (J. K. Kok et al., 2005). The model of the PowerMatcher mainly consists of devices that can be categorized by their type of controllability and SD-matchers (supply-demand-matchers). Devices can be stochastic (non-controllable), shiftable (within certain limits; e.g. a domestic washing machine), external resource buffering (producing a resource that has to keep a certain buffer; e.g. a domestic fridge that needs to maintain a temperature within set limits) and electricity storage, freely controllable or operated by user actions. These devices have different possibilities and constraints to control them. SD-matchers control a specified part of an electricity grid. The SD-matchers are interconnected using a tree-structure, where there is one root SD-matcher (corresponding with the energy trading market in 5.1.3) that defines the properties of the energy trading market and other SD-matchers forming a price for matching energy demand and supply and

aggregating the demand functions of the devices and other SD-matchers below them (K. Kok et al., 2008). Based on their flexibilities, devices will produce or consume more or less energy at a given timeframe. This way fewer imbalances on the energy network occur (K. Kok et al., 2012). The PowerMatcher concept has been proven by Jötten, Weidlich, Filipova-Neumann, & Schuller (2011) to effectively lower the balancing costs of a medium sized regional grid.

Based on the Common Information Model, V. M. Catterson, Baker, Davidson, & McArthur (2010) developed the *upper ontology for power engineering applications*. This ontology is mainly aimed at one application; the monitoring of the condition of power systems (i.e. devices in an energy network). Although this ontology is focused on one application, it remains an ontology of the energy domain. The main concepts of the model are system resource (devices in the energy network), measurement (e.g. a sensor reading or historical data), data interpretation (the derived meaning of a measurement; e.g. a measurement indicating a defect), value, timestamp and agent action (actions an agent can request from another agent; e.g. calculate the fault ratio). The model supports the exchange of messages between agents, although not explicitly defined. Lower ontologies could extend this upper ontology in order to, for example, maintain the balance between energy demand and supply in the energy network.

5.2 THE METHODOLOGY APPLIED

The defined methodology in the previous chapter is applied on the microgrid flexibility case, as described by section 5.1. The case is a theoretical case, which means the case relies mostly on available domain literature. To gain practical experience with involving domain experts in the methodology, the development methodology has been iterated and validated by several domain experts from TNO. The first subsection elaborates on the domain experts involved in this first application of the developed methodology. Each other subsection describes one step of the methodology that has been executed.

5.2.1 DOMAIN EXPERTS

To gain some practical experience involving domain experts in the methodology, the development methodology has been iterated twice by involving the domain knowledge of two domain experts. These experts were guided through the methodology steps and were asked to provide relevant additional domain information for each step involving domain knowledge as input. These are the steps from the first phase of the methodology determining the basic shared domain model, the steps determining the properties of agents and resources, determining domain constraints and class instances. The basic shared domain model and ontology were initially developed using relevant literature, after which each domain expert iterated the methodology to add missing relevant domain knowledge. This way a strong shared domain model and ontology can be built.

After the methodology was applied, the developed domain model and messages defined were reviewed by a third domain expert on validity and completeness. This expert was therefore asked whether the developed domain model can represent a microgrid, in its simplified form, and, taking into account the simplifications of the case, whether the domain concepts modeled are complete. It was initially concluded by this expert that the presence of the role of the energy supplier was missing. After the addition of the supplier and its commitment in the microgrid, the domain expert concluded that the defined resources, agents and their commitments were valid and complete taking into account the simplifications of the case.

The number of domain experts involved is limited due to time limitations on this research. This means the number of domain experts involved is insufficient for drawing conclusions for the practical usage of the methodology involving domain experts, but it can still serve the goal to create an initial insight into how domain experts can be involved in the development methodology.

5.2.2 IDENTIFY SCOPE

As mentioned before, the scope of this case is limited to microgrids in energy networks. To prevent misinterpretations, in this case the term “energy” refers to “electrical energy” only. The role renewable energy sources play and the required energy demand and supply flexibility that comes with them are also within the scope of the case. The latter also touches the purpose of the case; building an ontology and providing a base for an ICT solution that facilitates flexibility in energy demand and supply, but also alleviates interoperability between the involved stakeholders of the microgrid. In order to realize this, balancing the energy demand and supply in blocks of 15 minutes. Blocks of 15 minutes are chosen, because these provides enough space for other actors to maintain the quality of the energy in the grid (well within the 50-60 Hz range), while there is still a lot of flexibility in energy load over time.

The scope of the solution excludes the forecasting of energy demand on longer terms. The microgrid modeled comprises of enough information the producers themselves can use for their own forecasting activities. This case assumes energy producers and consumers to make their requests for energy consumption or production ad-hoc.

Therefore the solution should be able to answer the following competency questions:

1. Can the solution facilitate the exchange of energy between energy consumer and supplier?
2. Can the solution cope with renewable energy sources to influence their energy production?
3. Can the solution influence the energy demand over time of energy consumers?
4. Can the solution influence the energy production of energy producers?
5. Can the solution balance energy demand and supply in blocks of 15 minutes?
6. Can the solution interact with the external power grid of the microgrid?

5.2.3 DETERMINE SHARED BUSINESS DOMAIN MODEL

5.2.3.1 IDENTIFY AGENTS AND RESOURCES

Agent / resource	Description
Consumer	Consumes energy. The load of energy has a certain flexibility over a given time period. It wants the best (cheapest) price for energy on the market. It can also be a hybrid solution such as a short term energy storage that can consume an amount of energy from the grid that can help balancing the energy consumption and production on the grid. Also, prosumers who do not produce enough energy for themselves need to consume additional energy.
Producer	Produces energy. The load of energy has a certain flexibility over a given time period. It wants the best (highest) price for energy on the market. It can also be a hybrid solution such as a short term energy storage. Some producers in the microgrid can be identified as DERs (distributed energy resources), which produce energy on a small or medium scale. Other types include short term energy storages that can put energy stored earlier back on the grid and prosumers with e.g. solar panels that have excess energy they can put in the grid.
FlexMarket	Negotiates an energy price between consumers and producers with the goal to reach an optimal balance in energy production and consumption at any given point in time. It is the balance responsible party of the grid. (Future situation:) its (price) negotiations are influenced by the congestion constrictions imposed by the DSO, leading to energy flows of only the highest bidder in a specific part of the power grid that certain congestion constrictions are imposed on. In case matching consumption and production flexibilities leads

	to an impossible balance, energy import or export to the external power grid is requested at a supplier.
DSO	(Future situation:) based on its own information, it can impose constrictions to the FlexMarket for specific quantities of energy production and/or consumption of a specific consumer or producer.
Supplier	Imports or exports energy from or to the microgrid based on requests from the FlexMarket. (The administration on contracts and price agreements is out of scope for this case.)
EnergyFlexibility	Represents the information a consumer and producer share with the FlexMarket to negotiate about their future energy consumption and production. This information comprises of the time range the offer is valid for, the total power to be consumed or produced and the minimum and maximum power load in a block of 15 minutes.
MarketOfferFlexibility	Represents the information the FlexMarket shares with consumers and producers after an initiated flex offer negotiation. The information comprises of an indication whether the offer can be accepted at all (in case congestion restrictions apply), for each block of 15 minutes the exact amount of energy to be consumed or produced and the energy price it can offer for this offer.
EnergyAcquisition	Represents the acquisition of a MarketOffer. The information comprises of an indication whether the MarketOffer is accepted and also relates to the MarketOffer itself.
CongestionInformation	Represents the information of the DSO about congestion limitations that should be imposed on the microgrid, which comes down to the FlexMarket having to take these limitations into account. This information is based on other information available to the DSO.
EnergyRequest	Represents a request to a supplier for energy import or export from or to the microgrid. The information comprises of the exact amount of energy to be imported or exported, the block of 15 minutes it should take place and an indication whether the energy needs to be imported from or exported to the microgrid.

Table 17: Overview of the actors and objects in the microgrid domain

5.2.3.2 IDENTIFY COMMITMENTS

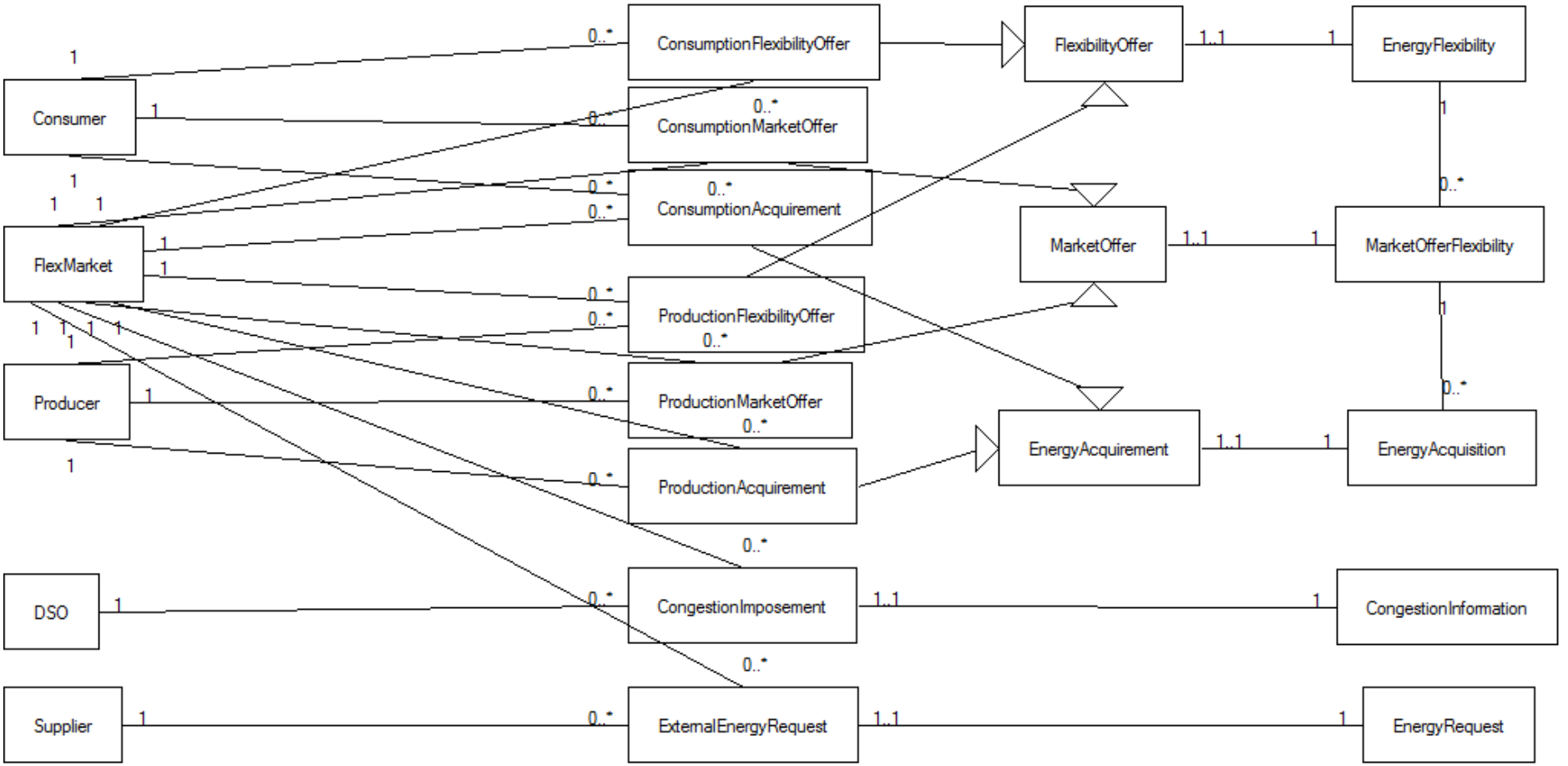


Figure 23: Identified commitments in the basic shared domain model

Commitment	Description
ConsumptionFlexibilityOffer	In the microgrid consumers will send their current flexibility in power demands for a specific range of time in the future, on which the FlexMarket will reply a ConsumptionMarketOffer.
ProductionFlexibilityOffer	In the microgrid producers will send their current flexibility in power supply for a specific range of time in the future, on which the FlexMarket will reply a ProductionMarketOffer.
ConsumptionMarketOffer	Based on the current energy availability and congestion information, the FlexMarket will send a reply to a consumer who sent a FlexibilityOffer. This reply states the price it costs the consumer to consume energy and gives a schedule of consumption (i.e. amount of energy to consume per time interval of 15 minutes). The consumer will have to reply on this offer with a ConsumptionAcquirement.
ProductionmarketOffer	Based on the current energy demand and congestion information, the FlexMarket will send a reply to a producer who sent a FlexibilityOffer. This reply states the price it can give the producer to produce energy and gives a schedule of production (i.e. amount of energy to produce per time interval of 15 minutes). The producer will have to reply on this offer with a ProductionAcquirement.
ConsumptionAcquirement	Based on the received MarketOffer and reasoning about the stated price, the consumer will send a reply to the MarketOffer to state it is willing to acquire the offer. If the consumer declines the offer, the consumer could issue a new FlexibilityOffer with a higher or lower flexibility to check how the price varies.
ProductionAcquirement	Based on the received MarketOffer and reasoning about the stated price, the producer will send a reply to the MarketOffer to state it is willing to acquire the offer. If the producer declines the offer, the producer could issue a new FlexibilityOffer with a higher or lower flexibility to check how the price varies.
CongestionImposement	In the microgrid congestion limits apply. The DSO will send the FlexMarket updates about the limitations it should apply on the production or consumption of energy of a specified consumer or producer in the grid.
ExternalEnergyRequest	When the FlexMarket cannot find a balance in energy demand and supply utilizing the available flexibility, the FlexMarket can request the import or export of energy from the external power grid the microgrid is connected to.

Table 18: Overview of the commitments of the agents in the microgrid domain

5.2.3.3 IDENTIFY EVENTS

In MERODE for each commitment in the domain model, at least one event exists, as can be derived from the object-event table (OET) in Figure 24. Table 19 relates the commitments identified in Table 18 to MERODE events that fulfil them. In this domain every commitment is fulfilled by only one event.

	Consumer	FlexMarket	Producer	DSO	Supplier	CongestionImposement	ExternalEnergyRequest	FlexibilityOffer	ConsumptionFlexibilityOffer	ProductionFlexibilityOffer	MarketOffer	ConsumptionMarketOffer	ProductionMarketOffer	EnergyAcquirement	ConsumptionAcquirement	ProductionAcquirement	EnergyFlexibility	MarketOfferFlexibility	EnergyAcquisition	CongestionInformation	EnergyRequest
imposeCongestion		A/M		A/M		O/C														A/M	
requestExternalEnergy		A/M			A/M		O/C														A/M
negotiateFlexibilityOffer	A/M	A/M A/M	A/M					O/C	I/C	I/C							A/M				
negotiateConsumptionFlexibilityOffer	A/M	A/M							S/C												
negotiateProductionFlexibilityOffer		A/M	A/M							S/C											
negotiateMarketOffer	A/M	A/M A/M	A/M								O/C	I/C	I/C				A/M	A/M			
negotiateConsumptionMarketOffer	A/M	A/M										S/C									
negotiateProductionMarketOffer		A/M	A/M										S/C								
acquireEnergy	A/M	A/M A/M	A/M											O/C	I/C	I/C	A/M	A/M	A/M		
acquireEnergyConsumption	A/M	A/M													S/C						
acquireEnergyProduction		A/M	A/M													S/C					

Figure 24: OET of the microgrid basic shared domain model

Event	Commitment
imposeCongestion	ConsumptionImposement
requestExternalEnergy	ExternalEnergyRequest
negotiateConsumptionFlexibility	ConsumptionFlexibilityOffer
negotiateProductionFlexibility	ProductionFlexibilityOffer
negotiateConsumptionMarketOffer	ConsumptionMarketOffer
negotiatieProductionmarketOffer	ProductionMarketOffer
acquireEnergyConsumption	ConsumptionAcquirement
acquireEnergyProduction	ProductionAcquirement

Table 19: Events identified fulfilling the commitments in the domain model

5.2.3.4 MAKE UML ACTIVITY DIAGRAMS

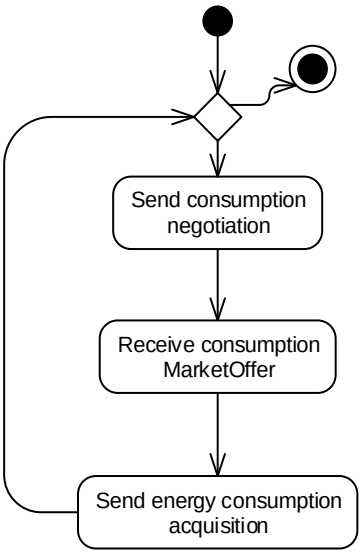


Figure 25: Activity diagram consumer

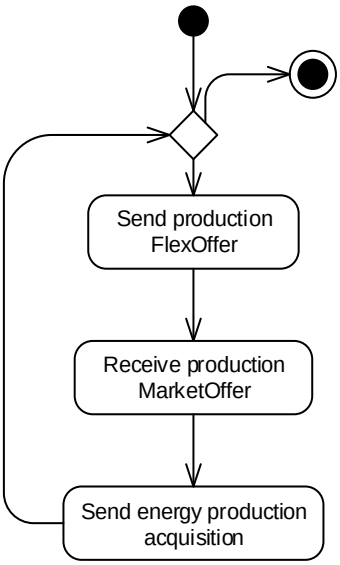


Figure 26: Activity diagram producer

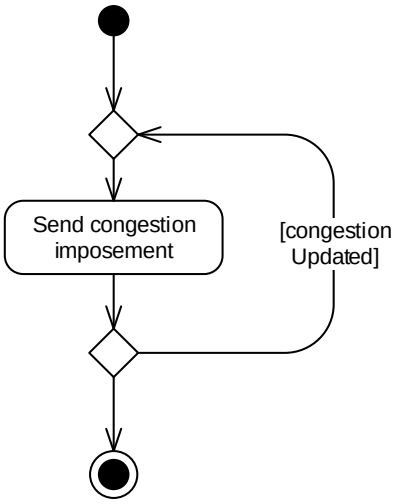


Figure 27: Activity diagram DSO

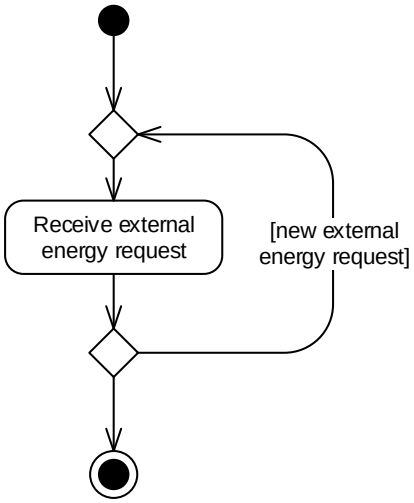


Figure 28: Activity diagram supplier

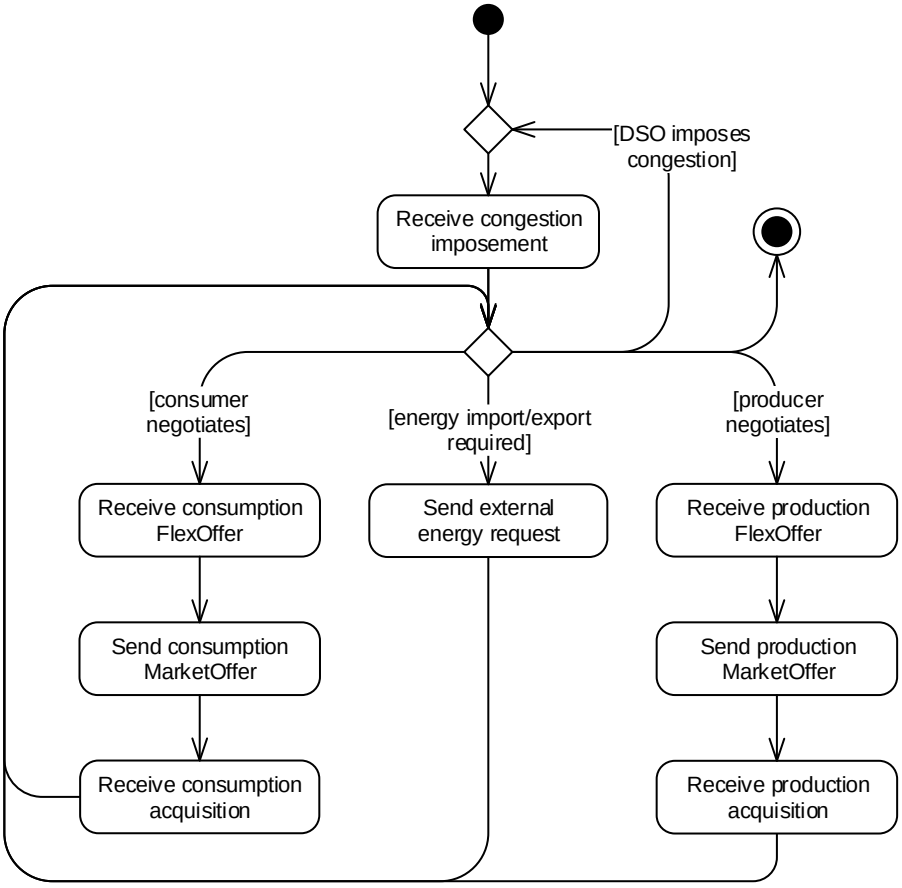


Figure 29: Activity diagram flex market

5.2.4 BUILD ONTOLOGY BASE

All concepts defined in the previous phase have been directly linked to the REA-ontology. The MERODE events have been redefined in terms of incremental and decremental event types. Table 20 shows how the events are redefined. Participations and stockflows have been defined to relate the defined events with the relevant resource and agent types.

MERODE events	REA events
imposeCongestion	SendCongestionImposement
	ReceiveCongestionImposement
requestExternalEnergy	SendExternalEnergyRequest
	ReceiveExternalEnergyRequest
negotiateConsumptionFlexibilityOffer	SendConsumptionFlexibilityOffer
	ReceiveConsumptionFlexibilityOffer
negotiateProductionFlexibilityOffer	SendProductionFlexibilityOffer
	ReceiveProductionFlexibilityOffer
negotiateConsumptionMarketOffer	SendConsumptionMarketOffer
	ReceiveConsumptionMarketOffer
negotiateProductionMarketOffer	SendProductionMarketOffer
	ReceiveProductionMarketOffer
acquireEnergyConsumption	SendConsumptionAcquirement
	ReceiveConsumptionAcquirement
acquireEnergyProduction	SendProductionAcquirement
	ReceiveProductionAcquirement

Table 20: Mapping MERODE events on REA events of the microgrid case

5.2.5 DEVELOP ONTOLOGY

5.2.5.1 REUSE AND INTEGRATE EXISTING ONTOLOGIES

The domain models found in the domain literature either did not comprise of an ontology yet, or did not provide a published or accessible ontology repository. While some ontology domain models were available of the energy sector, smart grids or microgrids, these were either in form of a traditional information model or an unpublished ontology. Therefore this step cannot be demonstrated in this case.

5.2.5.2 DETERMINE PROPERTIES OF AGENTS AND RESOURCES

The following properties were specified:

Concept	Property	Description
CongestionInformation	(min 1) ciHasLoad Constraint	For each consumer and producer the congestion constraints can be shown with a (new) concept “LoadConstraint” (see next row).
LoadConstraint	loadConstraintApplies ToAgent	The (one) AgentType (either consumer or producer) to which this constraint applies.
	maximumLoad Congestion	Indicates the maximum amount of energy that can be sent or received by the related agent. (this is a non-negative integer)
EnergyAcquisition	relatedMarketOffer	Refers to the related MarketOffer to which this EnergyAcquisition is a reply to.
	acceptMarketOffer	Indicates whether the related MarketOffer is accepted or not (Boolean).
EnergyFlexibility	minimumFlexLoad	The lower bound of energy load offered per 15 minutes (in Watt). (this is a non-negative integer)
	maximumFlexLoad	The upper bound of energy load offered per 15 minutes (in Watt) (this is a non-negative integer)
	totalPower	The total amount of energy to be offered in the given time range (in Watt) (this is a non-negative integer)

	startTimeRange	The lower bound of the time range the flexibility offer is valid. (dateTime)
	endTimeRange	The uper bound of the time range the flexibility offer is valid. (dateTime)
MarketOfferFlexibility	acceptEnergyFlexibility	Indicates whether the related EnergyFlexibility is accepted or not (Boolean).
	price	The price that can be offered for this MarketOffer. (this is a non-negative number)
	(min 1) mofHasEnergy Profile	For each block of 15 minutes the exact amount of energy to consume or produce is indicated by using a new sub-resource “EnergyProfile” (see next row)
EnergyProfile	epHasTimeBlock	The start time of a 15 minute time block for which the related energy quantity applies (dateTime)
	epHasEnergyQuantity	The amount of energy to consume or produce in the related time block (in Watt) (this is a non-negative integer)
EnergyRequest	energyQuantity	The amount of energy requested to either import or export from or to the microgrid (in Watt) (this is a non-negative integer)
	energyImportTo Microgrid	Indicates whether the request is to import the specified quantity of energy to the microgrid, or if not, export the specified quantity of energy from the microgrid to the grid outside the microgrid. (Boolean)
	timeBlock	Specifies to which time block of 15 minutes this request applies. (dateTime)

Table 21: Properties specified of each concept

5.2.5.3 DETERMINE INFORMAL CONSTRAINTS

- A flex offer is declined when the offering AgentType (consumer or producer) has a minimumFlexLoad that is higher than the maximumLoadCongestion imposed on this AgentType.
- Of a flex offer, the minimumFlexLoad is always smaller than the maximumFlexLoad.
- Of a flex offer, the startTimeRange is always smaller than the endTimeRange.

5.2.5.4 DETERMINE FORMAL CONSTRAINTS

In SWRL:

- EnergyFlexibility(?flexoffer), minimumFlexLoad(?flexoffer, ?minload), maximumFlexLoad(?flexoffer, ?maxload), resHasSfOutflow(?flexoffer, ?fooutflow), sfOutflowHasDecrEvent(?fooutflow, ?decrevent), decrEventHasPProvide(?decrevent, ?providep), pProvideHasAgent(?providep, ?agent), LoadConstraint(?lc), loadConstraintAppliesToAgent(?lc, ?agent), maximumLoadCongestion(?lc, ?maxload) -> lessThanOrEqual(?minload, ?maxload)
- EnergyFlexibility(?flexoffer), minimumFlexLoad(?flexoffer, ?minflex), maximumFlexLoad(?flexoffer, ?maxflex) -> lessThanOrEqual(?minflex, ?maxflex)
- EnergyFlexibility(?flexoffer), startTimeRange(?flexoffer, ?starttime), endTimeRange(?flexoffer, ?endtime) -> lessThanOrEqual(?starttime, ?endtime)

5.2.5.5 DETERMINE CLASS INSTANCES

In this step instances of the defined domain concepts can be determined based on the situation in the real domain. As this case is a theoretical case, the instances determined are fictional but should represent a possible real-world scenario. Table 22 shows the exact instances determined in the ontology.

REA concept	Domain concept	Individual
AgentType	Consumer	house1, house2, battery1
	DSO	holidayPark1
	FlexMarket	flexMarket1
	Producer	windTurbine1, battery1
	Supplier	Eneco
Commitment	CongestionImposement	ci1
	ConsumptionAcquirement	caHouse1, caHouse2, caBattery1
	ProductionAcquirement	paWindTurbine1, paBattery1
	ExternalEnergyRequest	eer1
	ConsumptionFlexibilityOffer	cfoHouse1, cfoHouse2, cfoBattery1
	ProductionFlexibilityOffer	pfoWindTurbine1, pfoBattery1
	ConsumptionMarketOffer	cmoHouse1, cmoHouse2, cmoBattery1
	ProductionMarketOffer	pmoWindTurbine1, pmoBattery1
(Decremental/Incremental) EventType	(Send/Receive) CongestionImposement	sci1 rci1
	(Send/Receive) ConsumptionMarketOffer	scmoHouse1, scmoHouse2, scmoBattery1, rcmoHouse1, rcmoHouse2, rcmoBattery1
	(Send/Receive) ConsumptionNegotiation	scnHouse1, scnHouse2, scnBattery1, rcnHouse1, rcnHouse2, rcnBattery1
	(Send/Receive) EnergyConsumptionAcquisition	secaHouse1, secaHouse2, secaBattery1, recaHouse1, recaHouse2, recaBattery1
	(Send/Receive) EnergyProductionAcquisition	sepaWindTurbine1, sepaBattery1, repaWindTurbine1, repaBattery1
	(Send/Receive) ExternalEnergyRequest	seer1, reer1
	(Send/Receive) ProductionMarketOffer	spmoWindTurbine1, spmoBattery1, rpmoWindTurbine1, rpmoBattery1
	(Send/Receive) ProductionNegotiation	spnWindTurbine1, spnBattery1, rpnWindTurbine1, rpnBattery1
Participation (Provide/Receive) Type	(Sends/Receives) CongestionImposement	provideci1, receiveci1
	(Sends/Receives) ConsumptionMarketOffer	providecmoHouse1, providecmoHouse2, providecmoBattery1, receivecmoHouse1, receivecmoHouse2, receivecmoBattery1
	(Sends/Receives) ConsumptionNegotiation	providecnHouse1, providecnHouse2, providecnBattery1, receivecnHouse1, receivecnHouse2, receivecnBattery1
	(Sends/Receives) EnergyConsumptionAcquisition	provideecaHouse1, provideecaHouse2, provideecaBattery1, receiveecaHouse1, receiveecaHouse2, receiveecaBattery1
	(Sends/Receives) EnergyProductionAcquisition	provideepaWindTurbine1, provideepaBattery1, receiveepaWindTurbine1, receiveepaBattery1
	(Sends/Receives) ExternalEnergyRequest	provideeer1, receiveeer1

	(Sends/Receives) ProductionMarketOffer	providepmoWindTurbine1, providepmoBattery1, receivepmoWindTurbine1, receivepmoBattery1
	(Sends/Receives) ProductionNegotiation	providepnWindTurbine1, providepnBattery1, receivepnWindTurbine1, receivepnBattery1
ResourceType	CongestionInformation	cinfo1
	LoadConstraint	lcHouse1, lcHouse2, lcBattery1, lcWindTurbine1
	EnergyAcquisition	eaHouse1, eaHouse2, eaBattery1c, eaBattery1p, eaWindTurbine1
	EnergyRequest	er1
	EnergyFlexibility	efHouse1, efHouse2, efBattery1c, efBattery1p, efWindTurbine1
	MarketOfferFlexibility	mofHouse1, mofHouse2, mofBattery1c, mofBattery1p, mofWindTurbine1
	EnergyProfile	epHouse1, epHouse2, epBattery1c, epBattery1p, epWindTurbine1
Stockflow (Inflow/Outflow) Type	CongestionInformation (Inflow/Outflow)	inflowci1, outflowci1
	EnergyAcquisition (Inflow/Outflow)	infloweaHouse1, infloweaHouse2, infloweaBattery1c, infloweaBattery1p infloweaWindTurbine1, outfloweaHouse1, outfloweaHouse2, outfloweaBattery1c, outfloweaBattery1p, outfloweaWindTurbine1
	EnergyFlexibility (Inflow/Outflow)	inflowefHouse1, inflowefHouse2, inflowefBattery1c, inflowBattery1p, inflowefWindTurbine1, outflowefHouse1, outflowefHouse2, outflowBattery1c, outflowBattery1p, outflowefWindTurbine1
	EnergyRequest (Inflow/Outflow)	inflower1, outflower1
	MarketOfferFlexibility (Inflow/Outflow)	inflowmofHouse1, inflowmofHouse2, inflowmofBattery1c, inflowmofBattery1p, inflowmofWindTurbine1, outflowmofHouse1, outflowmofHouse2, outflowmofBattery1c, outflowmofBattery1p, outflowmofWindTurbine1

Table 22: Instances defined in the ontology of the microgrid case

5.2.6 DETERMINE TECHNOLOGY-SPECIFIC SOLUTION

As stated in section 4.2.4 due to time limitations to this thesis, the transformation of OWL to XML messages is not covered. A general translation from OWL (XML) to a technology-specific solution (e.g. XML messages) should be used taking into account the event sequences. In Query 2 is shown a SPARQL-query that can be executed on the built ontology to retrieve a table that takes into account the instances of concept types. Executing this query on the built ontology results in Figure 30.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX my: <http://www.rea-ontology.com/rea-smartgrid#>
SELECT ?eventtype ?messagefield ?sendertype ?receivertype ?nexteventtype
WHERE {
       ?decrementalevent rdf:type ?eventtype.
       ?eventtype rdfs:subClassOf* my:DecrementalEventType.
       ?allsfs rdfs:subPropertyOf my:eventHasSf.
       ?decrementalevent ?allsfs ?relatedsfs.
       ?allrelatedresources rdfs:subPropertyOf my:sfHasRes.
       ?relatedsfs ?allrelatedresources ?resource.
       ?messagefield rdfs:subPropertyOf* my:message.
       ?resource ?messagefield ?message.
       ?allparticipations rdfs:subPropertyOf my:evHasP.
       ?decrementalevent ?allparticipations ?participations.
       ?allagents rdfs:subPropertyOf my:pHasA.
       ?participations ?allagents ?sender.
       ?sendertype rdfs:subClassOf my:AgentType.
       ?sender a ?sendertype.
       ?allDualities rdfs:subPropertyOf my:dualEvent.
       ?decrementalevent ?allDualities ?dualevent.
       ?dualparticipations rdfs:subPropertyOf my:evHasP.
       ?dualevent ?dualparticipations ?dualparticipation.
       ?alldualagents rdfs:subPropertyOf my:pHasA.
       ?dualparticipation ?alldualagents ?receiver.
       ?receivertype rdfs:subClassOf my:AgentType.
       ?receiver a ?receivertype.
       ?decrementalevent my:nextEvent ?nextevent.
       ?nexteventtype rdfs:subClassOf* my:EventType.
       ?nextevent a ?nexteventtype.
}
```

Query 2: SPARQL-query to retrieve the relevant information required to develop message structures

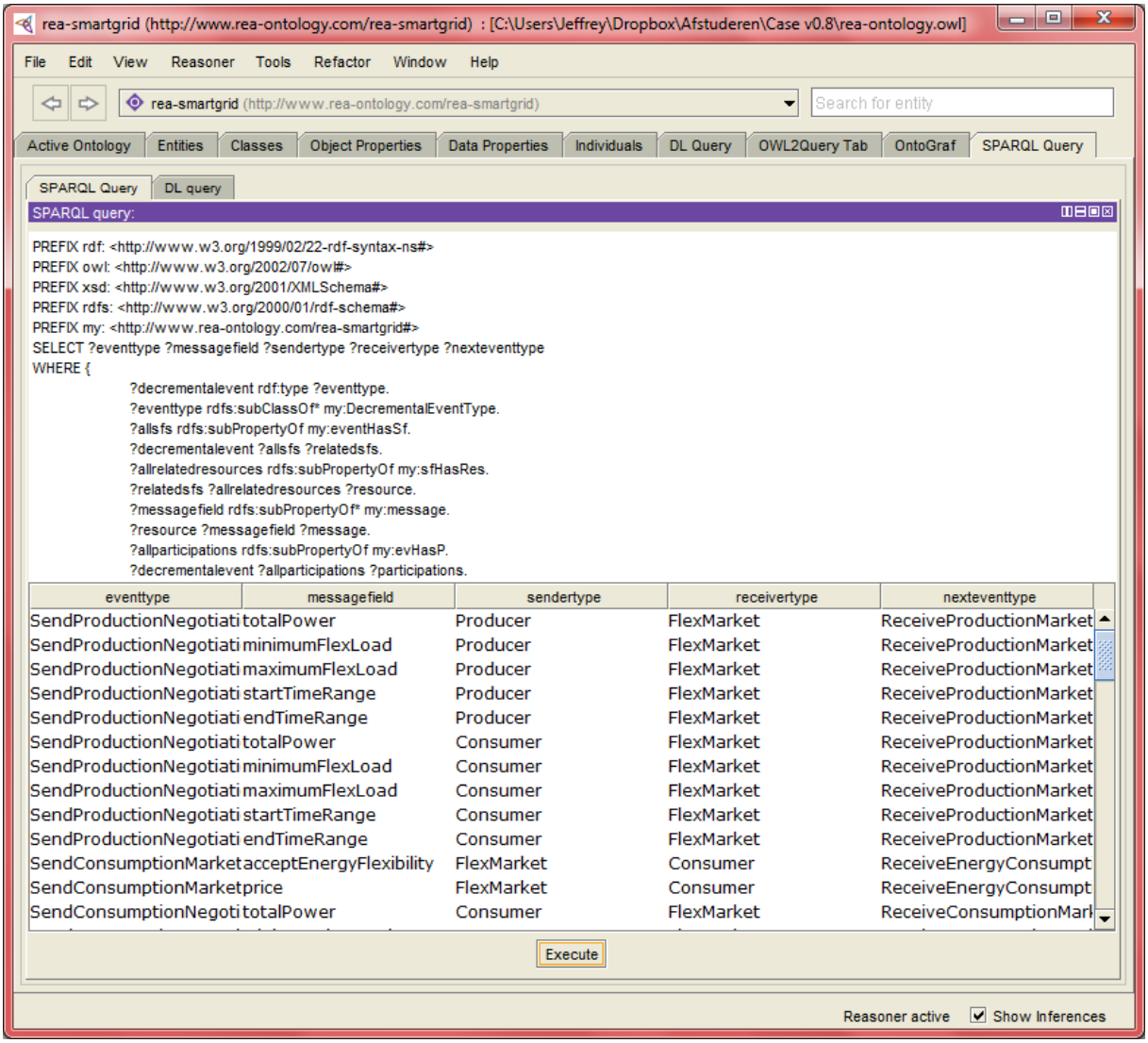


Figure 30: Screenshot of the SPARQL message query output in Protégé

5.2.6.1 AN ADDITIONAL APPLICATION

Additional information can be retrieved from the information already present in the ontology developed. One of many possible applications is creatig an overview of the congestion impositions of the DSO. Query 3 shows the query that can be asked to the ontology to retrieve the relevant domain information. Figure 31 shows the output table from this query.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX my: <http://www.rea-ontology.com/rea-smartgrid#>
SELECT ?gridparticipant ?participanttype ?maximumload
WHERE {
        ?loadconstraint rdf:type my:LoadConstraint.
        ?loadconstraint my:loadConstraintAppliesToAgent ?gridparticipant.
        ?participanttype rdfs:subClassOf my:AgentType.
        ?gridparticipant rdf:type ?participanttype.
        ?loadconstraint my:maximumLoadCongestion ?maximumload.
}
```

Query 3: SPARQL-query to retrieve the relevant information to gain insights into the congestion impositions in the microgrid

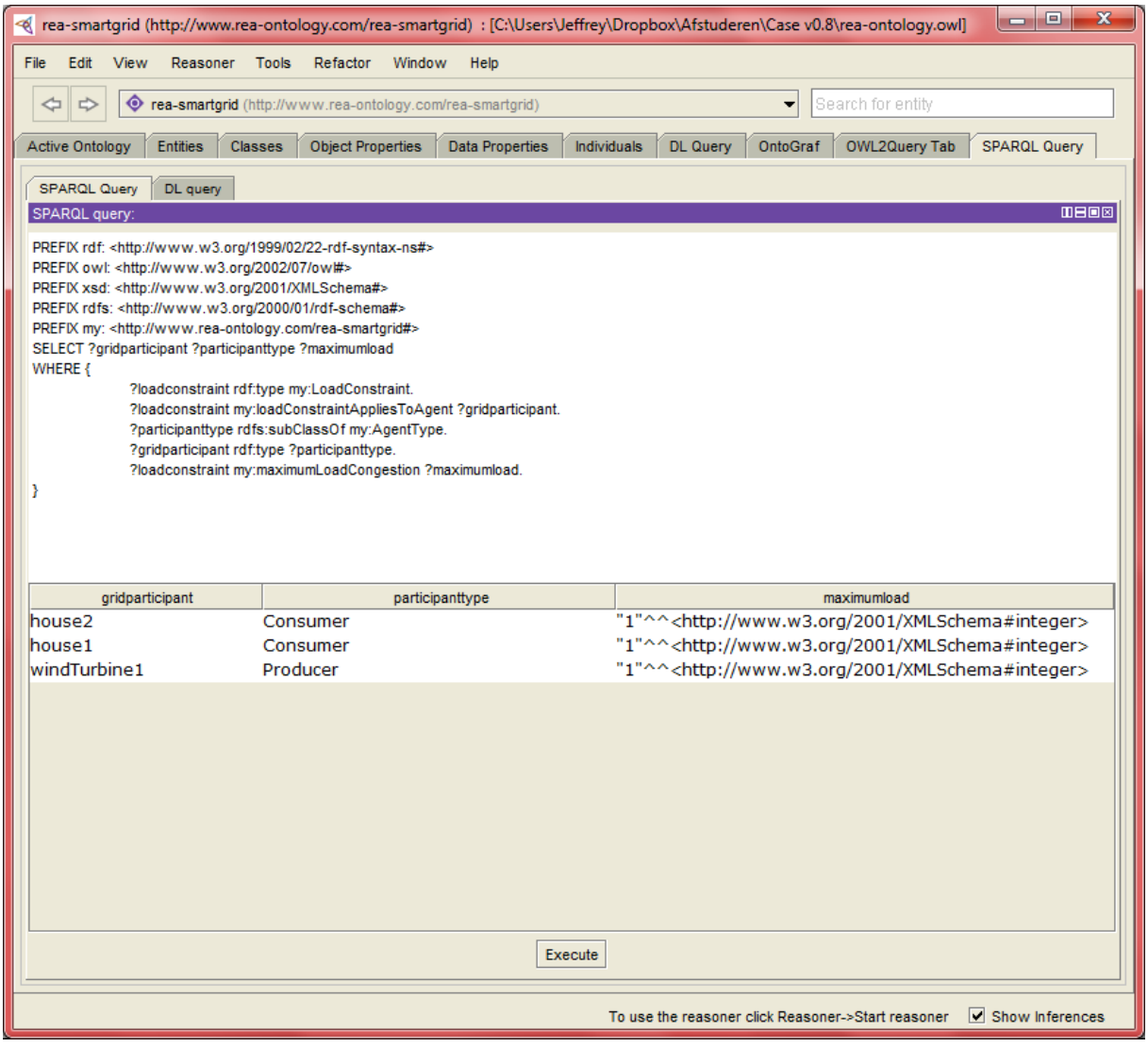


Figure 31: Screenshot of the SPARQL congestion query output in Protégé

5.3 DISCUSSION

By applying the newly designed methodology to the microgrid domain, some hands-on experience on the methodology was gained. The application of the methodology in the previous section involved domain experts, for which their expected and actual contributions to the domain model are discussed in the first following subsection. After that the applicability of the developed methodology is discussed by looking into the positive and negative properties of the methodology. Then the useful, and less useful, applications of the methodology are discussed by answering the question about in which circumstances a methodology using an ontology or an information model should be used. Based on this discussion, in the next chapter several conclusions can be drawn and lessons learned can be stated.

5.3.1 DOMAIN EXPERTS

The application of the methodology on a microgrid involved two domain experts who were involved in the relevant steps of the development methodology. A third domain expert reviewed the domain model and messages defined after the methodology steps were executed. This low number of domain experts involved is insufficient for drawing conclusions for the practical usage of the methodology involving domain experts. Yet the involvement of several domain experts can create initial insights into how domain experts can be involved in developing a domain model and a semantic standard.

The methodology steps determining the concepts of the basic shared domain model, determining the properties of agents and resources and determining domain constraints involved the knowledge from domain experts. As two domain experts were asked to add domain knowledge to the domain model and ontology throughout the methodology steps, the experts gave a lot of additional knowledge to extend the models with. In the phase where the basic shared domain model concepts were defined, additional competency questions were set. These are the competency questions 5 and 6 in section 5.2.2. Furthermore the price negotiation commitments involving the flex market and producers and consumers was defined in a better way than previously was defined based on domain literature. Another improvement the domain experts gave the domain model concepts were the definition of the DSO and the temporary energy storage devices. The properties of an energy profile that is sent by the flex market were also adjusted to be conform time blocks of 15 minutes. One domain expert also stated an extra domain constraint that allows the declining of flex offers.

The third domain expert also provided valuable knowledge input. As he was asked to review the domain model, he came to the conclusion the supplier agent was missing from the domain model, which plays an important role for the flex market allowing the microgrid to be balanced by importing or exporting energy from or to the microgrid. After this lack of knowledge was added to the domain, the expert had a second look at the models and, taking into account the case simplifications, concluded the defined resources, agents and their commitments were valid and complete.

This involvement of domain experts in the methodology clearly shows benefits as added domain knowledge and completeness checks of the domain model and properties. Even though this use case can only provide initial insights into how domain experts can be involved in the development approach, this application of the methodology in the case of a microgrid gives clear indications domain experts provide additional values to the domain model, and therefore also the ontology and semantic standard.

5.3.2 POSITIVE PROPERTIES

Actor and object modeling for the basic domain model is limited with MERODE because it does not allow *multiple inheritance*. Making use of an ontology allows the *reasoning about subsets of classes* and therefore provides a useful addition to the modeling possibilities of MERODE. As an example, a battery is both an energy

consumer as well as an energy producer. In MERODE this device should be defined as two separate classes; one for energy production and another for energy consumption. Reasoning about the battery device also becomes easier using an ontology, as it allows a battery to be of both the consumer and producer type.

When creating classes and instances of the developed ontology, these are checked on the posed assertions on the ontology classes before and during any instance is created and on the ontology superclasses before and during any subclass is created (provided a reasoner is activated). This *asserts no incorrect instances can exist at any time*. We think that when traditional information models are used, the assertion of model instances cannot be as well-guaranteed as this is possible with an ontology, because the instances created for an information model with constraints are expected to be checked on their correctness after they have been created.

The use of *informal knowledge capturing* before formalizing knowledge proved to be convenient to prepare for formal requirements as well as defining model components. Because there were no language boundaries (expressing requirements in a formal computer language is very limited), it was easy to state domain requirements and also the actors, resources and their dynamic behavior in the domain. By taking this approach, the formalization of each requirement and component was only a translation process and therefore simplified.

The use of *competency questions* proved to be convenient for defining a very specific scope on the domain. The competency questions also proved useful for evaluating the completeness of the basic domain model as well as the ontology domain model, during the iterating through the development phases of these models.

Because an ontology can be queried for specific information (Gasevic & Djuric, 2006), *a query language* like SPARQL can be used to *quickly retrieve very specific parts of the domain ontology to gain extra insights*. In the method it is used to derive the message flows, but it can also be used to design other types of semantic standards or other insights. For example, with a query the events a given actor is involved in can be found to gain insights into its doings.

Throughout the development process steps of the designed methodology it was asserted that no concepts can exist at any time that do not comply with any of the posed rules and restrictions. This was asserted by making use of both the MERMAID and Protégé tool. Therefore, even in domains where many restrictions are required, a domain model and ontology can be developed effortless without any concept violating the domain restrictions.

Ontologies are designed to be reasoned with (Henderson-Sellers, 2011). For example, the Protégé editor comes with a built-in reasoner. If the ontology for some reason needs to be extended with, e.g., more concepts or information to be exchanged, *old and/or new restrictions on the ontology can automatically include this new extension* and an updated platform-specific solution can be automatically derived.

Because the REA upper ontology is central in the methodology's approach, during the whole development methodology the developers and domain experts need to think continuously in terms of resources, events and agents. This prevents the defining of ambiguous concepts and ambiguous model constructions.

5.3.3 NEGATIVE PROPERTIES

The new methodology adopts the REA ontology foundation. This means the structure of the whole domain model needs to be fit into this upper ontology, leading to an even *higher coupled model structure* than would be necessary in the original MOSES approach. As a high coupling could lead to a low system interoperability (Daclin et al., 2006), a higher coupling between classes in this new method could result in a lower system interoperability potential compared with the original MOSES methodology.

Mapping the resources, events, agents and relationships from the basic domain model to the REA ontology foundation *needs to be done by hand*. This mapping cannot be automated as extra domain knowledge needs to

be added during the transformation. For example relationship cardinalities need to be specified in more detail and set theories and their constraints need to be added during this transformation.

The REA ontology foundation *does not fully match the MERODE methodology*. Therefore several extra assertions need to be created in the ontology models in order to comply fully with the MERODE methodology. (See also section 4.2.2)

When agents in the domain of discourse make a commitment, a specified resource type is linked to this commitment, as are an incremental and decremental event type. All these concept types have to be created to support the modeling of one commitment. A disadvantage of using the (REA-) ontology is therefore that *the model expands very fast*, because of the many concepts that are interconnected with each other to describe the commitments, resources, stockflows, agents, participations and events in the domain. An advantage of an ontology is that it can be queried to look up specific parts of the ontology. In the situation where an overview of all domain concepts at once is required, using this (REA-) ontology is not a good option.

5.3.4 WHEN TO USE WHICH MEANS?

An alternative methodology for MOSES has been developed and applied on a case in this thesis. Its advantages and disadvantages have also been discussed in the sections above. Based on this information an indication can be given in which situations it is better to develop a semantic standard using the original MOSES methodology making use of a (traditional) information model, and when to use the newly developed methodology making use of an ontology.

Information model	Ontology
Low coupling between classes is essential for the interoperability potential	Reasoning about subsets of classes is useful
Domain knowledge is not available throughout the whole modeling process	Business logic can get complex or changes frequently
Having an overview of all domain concepts at once is required	Multiple inheritance of concepts is useful
The only intentions are to generate standard message structures	It is essential to have no loss of domain knowledge
	Need for fast and easy overviews to gain insight in certain parts of the domain
	Many and complex rules are required to assert a consistent model of the domain in reality
	Future extensions to the domain model are to be expected
	Integration with the “online web of data”: Open Linked Data

Table 23: Situations when to use either an information model or an ontology

5.3.4.1 INFORMATION MODEL

Building and using an information model for deriving semantic standards, i.e. the original MOSES methodology, is best suitable in cases where the domain concepts to be described need to be constructed in a loosely coupled way. This means that when the model architecture gets highly complex or unstructured if it were mapped on, for example, the REA upper ontology, it should be better to use an information model as means for deriving a semantic standard instead.

Another development situation where the use of information models is preferred over using an ontology as means is when domain knowledge sources are not available throughout the whole modeling process. In the original MOSES approach, domain knowledge is only appealed in the first phase, where the shared business domain is developed. The subsequent phases (deriving an information model and developing a shared solution model) are dependent on and influence the business domain model. The methodology developed in this thesis relies on the addition of domain knowledge throughout the development process; in each phase new domain knowledge is added to reach the final outcome step by step.

When overview on all domain concepts at once is required during the development or maintenance of the domain model, using an information model is a better alternative than an ontology. An information model provides an abstract overview of only the information that needs to be exchanged, while an ontology provides an overview of all domain concepts and their properties. Especially when the REA upper ontology is used, the number of classes expands very fast, as for one modeled commitment, also stockflows, participations and events have to be modeled. To mitigate this problem, standard queries can be designed for the ontology in order to develop specific overviews of the domain modeled.

The first phase of the developed methodology involves all steps necessary to provide a basic shared model of the domain; agents and resources are identified, but also the commitments between agents and the events that fulfil these commitments. Also part of this phase is to define the sequence of events happening in the domain. This basic shared domain model almost already provide enough information to determine all messages to be exchanged between the agents involved. The only thing missing is the content of the messages itself. While ontologies provide much more benefits (see also section 5.3.4.2) such as business logic and restrictions and extensibility, a simpler approach using only traditional information models would suffice and save effort, if purely the message structures need to be determined.

5.3.4.2 ONTOLOGY

In the case where the domain of discourse includes many different types of concepts and includes many constraints on subsets of these concepts, the use of an ontology is highly recommended. Both the micro grid case and theory indicate the usefulness of this ability of ontologies. This use of restrictions can also be applied on a wider business context; the business logic existing in a domain can already be modeled using this type of restrictions. Therefore, when the business logic is complex, business logic can already be implemented. Also, if the business logic changes over time, the modeling of business logic and changing these restrictions can result without much effort in a new, slightly adapted, semantic standard.

The same can be stated for the ability of ontologies to support multiple inheritance of concepts. Where the MERODE ontology only supports single inheritance, the microgrid case shows the usefulness of multiple inheritance for several concepts within the microgrid domain. An example is the temporary energy storage device, which can act both as an energy producer as well as consumer.

The new methodology is designed in such a way to prevent any loss of domain knowledge. By first stating informal requirements to the domain before formalizing them ensures no information about the domain is lost in a translation to formal requirements language before the informal requirements have been described.

It is possible to query ontologies. Therefore, by developing several standard queries for a developed ontology, very specific overviews of domain information can be created. Because these queries can give different views on the domain, the use of an ontology in combination with ontology queries can provide fast and highly relevant insights in the domain.

Many business domains consist of a lot of rules their participating agents, their commitments and resources are applying to. When modeling a business domain, all these rules have to be taken into account to assert the

model to represent the domain in reality well. Ontologies have native support for appending axioms to model concepts, whereas traditional information models require extensions, of which OCL is the most known. The most important downside of this is that when an existing constraint is added or a new constraint is added later to the traditional information model, other constraints or concepts also need to be updated by hand based on the changes in the model. Ontologies, on the other hand, can process these changes automatically, which gives it a large advantage over traditional information models. Also other extensions made to the domain model in a later stage can be processed without much effort.

An important aspect of an ontology is the possibility to integrate with the “online web of data”, also called Open Linked Data (Bizer, Heath, & Berners-Lee, 2009). The main focus of the methodology developed in this thesis is creating interoperability standards using an ontology model. While the generation of a technology-specific solution for interoperability messages can also be generated in almost the same ease using more traditional information models (see also section 5.3.4.1), ontologies can serve more purposes than generating semantic standards alone. A lot of domain information is stored in an ontology model, which is created throughout the developed methodology. Bizer et al. (2009) reason that once an ontology is published on the internet, other ontology models and other applications crawling the “web of data” can make use of this information for their own purposes and to form universal consensus on definitions of concepts.

For example, the ontology of a microgrid generated in the case study can be included in a wider ontology about all possible energy grids, which covers microgrids among other things. An example that could make use of this information is an application that can visualize flows of energy in power grids. When the application zooms in on the type of microgrid the ontology that was built in the microgrid use case, the information from this ontology will be used.

6 CONCLUSIONS

To conclude this thesis, this section will answer the main research question of this thesis by answering the sub-questions supporting the main research question. The main question was "*How can the MOSES methodology be extended with the development and use of an ontology?*". First, the answers on the sub-research questions are provided. After that, a reflection on the limitations of this research is made, followed by some recommendations for future work that lies in extension of this research.

RQ1: What is the state-of-the-art on ontology development methodologies?

This research question is aimed at providing an overview of ontology development methodologies. Context about the properties and possibilities of ontologies and the available ontology languages is given from the literature in section 2.2 and 3.1. The most important characteristics of ontologies are their conceptualizing property of all aspects of a real-world domain and their ability to include restrictions on their contents to assert consistency with concepts of the real-world domains. Other aspects of ontologies include the open-world assumption, which means it is assumed that not everything modeled with an ontology will restrict its interpretation of the domain modeled. Also, an ontology always provides concept definitions that are unambiguous and understandable and interpretable for all stakeholders involved with the ontology, as all stakeholders share this same ontology to collaborate. Next to that, an ontology can contain rules and semantics of the domain concepts, facilitating automated reasoning about these concepts.

In section 3.2 the most important ontology development methodologies are discussed. These include the methods of Enterprise Ontology, Methontology, Cyc, TOVE, Ontology Development 101 and DILIGENT. Each of these methodologies has a different origin of discipline and take different perspectives. To reach a more general domain ontology development methodology, the aspect of reusability of existing ontologies has been taken into account. The notion domain knowledge elicited from domain experts should initially also be recorded in a semi-informal way to prevent loss of knowledge and by forming competency questions at the start of development, an indication of the completeness of the ontology can be given.

RQ2: What are the benefits of the development and use of an ontology for interoperability in a domain?

The aim of this research question is to provide an overview of all potential benefits the use of an ontology offers to achieve interoperability. In this case, interoperability refers to the ease of exchange of information between domain agents. Section 3.1 lists many aspects of ontologies benefiting domain interoperability. The most important aspects are that an ontology provides a shared vocabulary with unambiguous concept descriptions for all stakeholders in the domain. By means of validity rules, the shared ontology can also rule out any configurations possible that do not square with the real world domain. Next to that, it is possible for ontologies to include dynamic behavior (e.g. events) of stakeholders by modeling their processes and operation rules.

RQ3: What are good additions to the MOSES methodology for improving interoperability in a domain using ontologies?

Taking into account the interoperability benefits ontologies can bring (see section 3), the MOSES methodology has been adapted to include the development and usage of an ontology. The exact methodology steps, mindsets and notations can be found in section 4. One of the main additions to the methodology is the use of the REA upper ontology to which each domain concept has to be mapped. This is an addition throughout the whole development process, which means the domain ontology developer continuously needs to think about the domain in terms of resources, events and agents, which leads to a clearer model as no ambiguous constructions can be modeled.

To include an ontology in the methodology, the business information modeling phase is replaced by two phases building an ontology base from the first phase and another to further specify the ontology model. In the latter phase, next to determining agent and resource properties, ontology constraints are determined. This is done by initially capturing the constraints in a semi-informal way to facilitate domain experts, followed by a constraints formalizing step. Another addition to MOSES is the method's initial step, where the scope of the development is identified. This sets clear boundaries of the domain to be described and the level of detail required. Especially for domain experts sharing their domain knowledge this can be of help.

RQ4: How can the extended methodology be applied and evaluated in the energy domain?

Based on literature and domain experts on microgrids and energy flexibility, the developed methodology was applied on the future scenario of a generic microgrid supporting flexibility in energy demand and supply. During the development several positive and negative experience was gained. The use of informal knowledge capturing proved to be convenient before formalizing the knowledge. Stating competency questions setting the boundaries and granularity of detail to be described in the case, also provided clarity for the domain experts. On the downside, the number of concepts in the ontology model expands rapidly per commitment described. This might pose an inconvenience when overviewing the whole domain model.

Based on all findings, this research concludes with a recommendation whether to use an ontology or information model based on the requirements. An information model can better be used in scenarios when low coupling between classes is essential for the interoperability potential of the model, domain knowledge is not available throughout the whole modeling process, a need exists to be able to overview all domain concepts at once and when the only purpose is to generate standard message structures. An ontology is a better choice when reasoning about subsets of concepts is useful, the domain described contains complex or frequently changing business logic, specific parts of the domain need to be overviewed in fast and easily generated overviews, future model extensions are to be expected and, perhaps the most important argument, the ontology has to be able to integrate with the "online web of data": Open Linked Data. The latter argument is of high value for semantic standards that want to reach a high level of interoperability in a wider domain scope or across different domains. The integration of the ontology with other ontologies in the "online web of data" leads to the usage and use of shared semantics of concepts of the current, but also other domains.

6.1 LIMITATIONS

One of the main focuses of this thesis was the development of a development methodology for semantic standards using an ontology as means. This methodology includes two transformation steps; one transformation from a MERODE model to a shared basic ontology model in REA and another from the REA ontology model to a technology-specific solution. Due to the time restrictions set for this thesis, only the development methodology steps could be determined. While the technologies used (MERODE domain model, OWL ontology and XML messages) allow for automatic translation due to their compatibility with XML, no transformations have been written for both transformations. Currently the transformations have to be done by hand.

The application of the designed methodology on microgrids gives some hands-on experience with the methodology. Nonetheless an execution and evaluation of only one use case does not empirically validate the designed methodology. More use cases should be executed and evaluated to empirically validate the methodology.

The ontology model and semantic standard developed for the microgrid are developed based on literature on microgrids, smart grids and interviews with domain experts. Unfortunately the ontology model is not designed for a real problem or need. The literature as well as the domain experts provided knowledge about microgrids

and flex offers in general. Therefore, to use the designed ontology model and semantic standard developed in a real scenario, they should be revised to meet the exact situation in which they will be applied.

6.2 REFLECTION

This section takes a look at the lessons learned by looking in retrospect to the whole process of developing this thesis and its final products (i.e. an improved methodology for developing semantic standards using an ontology as means and a domain ontology and message structures for a microgrid with flexibility in energy demand and supply). First, the strengths of this research are stated, followed by the weaknesses. The section concludes with a summarizing list of all lessons learned in this thesis.

6.2.1 STRENGTHS

The structure of this thesis, and the execution of the research involved, is based on the design science research methodology as prescribed by Peffers et al. (2007) (see also Figure 1). After starting with an elaborate literature research on ontologies, ontology development methods and foundational ontologies facilitating business domains, the development methodology was designed and developed. The mindsets and methodologies of the development methodologies formed the basis of the designed methodology, next to the original MOSES methodology and mindsets for achieving high interoperability in a domain model. The design of the methodology was a highly iterative process involving many reviews with experts on developing semantic standards. *The many review sessions with experts were of high value as throughout the development process the methodology steps became more and more coherent*. Also, as new aspects were added to the methodology, the review iterations helped greatly in embedding these aspects in the steps.

After the choice was made to use REA as upper ontology, developing a version of a REA upper ontology that perfectly suits the needs involved many iterations and feedback from ontology experts. To grasp the exact definitions of both the MERODE concepts as well as the REA concepts was difficult, but in the end this resulted in a highly coherent REA model. Next to that a one-on-one mapping needed to be created with the MERODE methodology of the first phase, which was derived from MOSES. Also here, *a good grasp of the definitions of MERODE and REA concepts was essential*. *By developing a mapping table and diagram* (see also section 4.2.2), *the concept definitions and concept mappings were elucidated*.

The technique in developing the development methodology focused on three main points: the mindset, the steps and the notation. The technique mainly focuses on the literature on ontologies, ontology development methods and interoperability aspects. For the mindset it is important to understand the thoughts and intentions behind the literature and to take or derive the most important aspects for the methodology developed. For the steps, the steps of the other methodologies and the reasoning why these steps need to be performed are taken into consideration. The notation mainly relied on the original MOSES methodology with a few additions of the more recent UML modeling techniques. The goal of developing the methodology was clear (deriving semantic standards pursuing high interoperability). *Because of having a clear goal, the most important mindsets, steps and notations could be drawn up without much effort*.

During the expert sessions developing an ontology for the microgrid case also several lessons were learned. During the sessions, domain experts sometimes referred back to the scope identified in the first step, which clarified how much detail they should provide in describing the domain, what simplifications are made in the domain model and which concepts to include and which not to include. *Therefore this step identifying the domain scope is of high value for the rest of the development process*. During the expert sessions is also found that even the more technical experts had trouble understanding the concept “commitment” and its “executes”-relationship with the concept “event”. A lesson that can be learned here is that *the terms involving the development methodology should be clarified to the domain experts before starting or during the start of*

the expert sessions. Possible options to do this include providing clear definitions for each term beforehand or briefly give an explanation on the most difficult terms at the start of an expert session.

6.2.2 WEAKNESSES

Looking back on the decision to use the MERODE methodology for determining the basic shared domain model *it would have been a better solution to directly determine this basic model in the same upper ontology as the rest of the methodology is using (REA)*. This would save the steps building the ontology base in REA, as this base would be automatically generated when the agents, resources, commitments and events were identified.

The developed methodology comprises of four main phases. The first results in a basic shared domain model in MERODE. The second transforms this model into a base ontology model (OWL) embedded in the REA upper ontology. Additional details are added to the ontology in the third phase, after which a technology-specific solution is derived in the fourth phase. As mentioned earlier, *if in the first phase not MERODE, but directly REA was used for determining the basic shared domain model, the second phase could be removed*.

6.2.3 LESSONS LEARNED

Based on the strengths and weaknesses stated above the following lessons can be learned:

- The many review sessions with experts were of high value as throughout the development process the methodology steps became more and more coherent. By involving domain experts, additional domain knowledge could be captured over the knowledge derived from relevant domain literature.
- To prevent obscurity in the domain scope, before developing the basic shared domain model and ontology, competency questions should be a formulated covering as much domain properties and boundaries as possible, as domain experts would need more detail in order to develop the domain model. In particular competency questions covering the level of detail the domain should get, what simplifications can be made and which parts to include or exclude from the domain.
- Domain experts need a thorough understanding of the meaning of the REA-concepts. Therefore, when performing the methodology, beforehand the REA-concepts (resource, event, agent, commitment, stockflow and participation) and their relations should be understood by the modeling expert to develop an adequate ontology model. The terms can, for example, be clarified to the domain experts before starting or during the start of the expert sessions.
- The conversion steps from MERODE to REA take up a considerable amount of effort. This could have been prevented if the basic shared domain model would have been developed initially in REA. A lesson learned for the future development is therefore to directly develop the basic shared domain model in terms of REA.

6.3 FUTURE WORK

The future work that lies in extension of this thesis should focus mainly on the automation of the developed methodology and on the validation of the developed methodology. Where the first area of research focuses on developing automation procedures, the second requires use cases for this method to be applied on and proper methodology evaluation techniques.

Due to time restriction on this thesis the designed methodology is currently not automated in the sense of model transformations. Currently transformations from MERODE to REA and from REA to a technology-specific solution (a common solution is XML messages) are described as “steps”, while these could also be automated using a transformation. To facilitate this methodology, these transformations could be developed in a language like XSLT or XPath (W3C, 1999a, 1999b), which can cope well with XML transformations. When these

transformations can be embedded in the methodology, the steps involving manual transformations can therefore be removed as it is automatized.

More future work in automatizing the methodology lies in developing an application that facilitates the whole development process described in this thesis. The application can be of high practical value for ontology developers, who can easily perform all steps of the method in this one application. The application development itself would not require more theoretical knowledge, but would only have to facilitate the methodology elaborated upon in chapter 4. Eventually, the identification of MERODE elements in the first methodology phase can be changed to directly identifying REA elements, skipping the MERODE methodology, as earlier was concluded that this would be a less laborious solution.

The developed methodology is not fully validated in this thesis, due to restrictions in time as well as no practical use case(s) were available. To fully validate the methodology it should be evaluated with proper evaluation techniques. It is suggested to apply the methodology on more use cases of different business domains, as well as the application of methodology evaluation techniques.

6.4 IMPLICATIONS AND RECOMMENDATIONS FOR PRACTICE

One of the goals of this study was to provide a development methodology for TNO to develop semantic standards making use of an ontology and all the benefits it brings. This study offers a methodology that takes the advantages of ontologies and embeds it in a development process for semantic standards. The generic design enables practitioners to develop semantic standards in any domain. While the building of a software tool supporting the methodology would improve the practical use is future work, the methodology as it is designed is already of high practical value as it provides detailed descriptions of the steps practitioners have to take. Appendix A provides a practical guide for practitioners who want to execute the developed methodology.

Executing the methodology steps results in a semantic standard, but also a complete domain ontology. The semantic standard has a clear purpose; standardizing communication between parties in the domain of discourse and improving their interoperability. Whether this methodology is easier to use than the original MOSES methodology was not assessed in this study, but several additions of this methodology to the original methodology facilitate the development process. The consistency rules embedded in ontologies can at least facilitate the possible difficulties of maintaining a consistent domain model. The added steps to clearly define the methodology's scope and to define semi-informal requirements before formalizing them also helps to overcome difficulties domain experts can cope with.

The ontology can be of practical use in numerous ways, as it contains a lot of information about both the static and dynamic parts of the domain, which is shared among all stakeholders. The information can be of use for management purposes, such as overseeing all activity in the domain, or a part of it. Also for other applications the ontology can be of interest. The information about agents, their commitments, resources and/or events can be of use. An example of this can be seen in section 5.2.6.1.

7 REFERENCES

Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., et al. (2006). Towards a Reference Ontology for Business Models. (D. Embley, A. Olivé, & S. Ram, Eds.)*Conceptual Modeling - ER 2006 SE - 36*, *4215*, 482–496. doi:10.1007/11901181_36

Arango, G., & Prieto-Diaz, R. (1991). Domain analysis concepts and research directions. *Domain analysis and software systems modeling*, 9–26. Retrieved from http://www.docstoc.com/docs/document-preview.aspx?doc_id=93912037

Ardito, L., Procaccianti, G., Menga, G., & Morisio, M. (2012). A Survey on Smart Grid Technologies in Europe. *ENERGY 2012 : The Second International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies* (pp. 22–28). St. Maarten, The Netherlands Antilles.

Aßmann, U., Zschaler, S., & Wagner, G. (2006). Ontologies, Meta-models, and the Model- Driven Paradigm. In C. Calero, F. Ruiz, & M. Piattini (Eds.), *Ontologies for Software Engineering and Technologies* (pp. 249–273). Springer-Verlag.

Bakker, V. (2012). *Triana: a control strategy for Smart Grids: Forecasting, planning & real-time control*. University of Twente, Enschede.

Bell, D. (2004, February 16). UML basics: The sequence diagram. Retrieved May 4, 2013, from http://www.ibm.com/developerworks/rational/library/3101.html

Bezivin, J., & Gerbe, O. (2001). Towards a precise definition of the OMG/MDA framework. *Automated Software Engineering, 2001. (ASE 2001). Proceedings. 16th Annual International Conference on* (pp. 273–280). doi:10.1109/ASE.2001.989813

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. *International Journal on Semantic …*. Retrieved from http://www.igi-global.com/article/linked-data-story-far/37496

Burton-Jones, A., & Weber, R. (1999). Understanding relationships with attributes in entity-relationship diagrams. *Proceedings of the 20th international conference on Information Systems* (pp. 214–228). Atlanta, GA, USA: Association for Information Systems. Retrieved from http://dl.acm.org/citation.cfm?id=352925.352946

Catterson, V. M., Baker, P. C., Davidson, E. M., & McArthur, S. D. J. (2010). An upper ontology for power engineering applications. Retrieved from http://sites.ieee.org/pes-mas/

CEN/CENELEC/SG-CG/M490/E. (2012). *SG-CG/M490/E Smart Grid Use Case Management Process*.

Chen, D, & Daclin, N. (2006). Framework for enterprise interoperability. *Proc. of IFAC Workshop EI2N*. Retrieved from http://chen33.free.fr/M2/Elearning/CIGI2009.Chen.final.pdf

Chen, David, Vallespir, B., & Daclin, N. (2008). An approach for enterprise interoperability measurement. *Proceedings of MoDISE-EUS*, 1–12. Retrieved from http://ceur-ws.org/Vol-341/paper1.pdf

Corcho, O, Fernandez-Lopez, M., & Gomez-Perez, A. (2007). Ontological engineering: what are ontologies and how can we build them? *Semantic Web Services* (pp. 44–70). Premier Reference Source. Retrieved from http://oa.upm.es/5456/

Corcho, Oscar, Fernández-López, M., & Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data & Knowledge Engineering*, *46*(1), 41–64. doi:10.1016/S0169-023X(02)00195-7

Cox, W., & Considine, T. (2011). Energy , Micromarkets , and Microgrids. *Grid-Interop Forum 2011*.

Crapo, A., Wang, X., Lizzi, J., & Larson, R. (2009). The semantically enabled smart grid. *The Road to an Interoperable Grid (Grid-Interop)*. Retrieved from http://www.gridwiseac.org/pdfs/forum_papers09/crapo.pdf

Daclin, N., Chen, D., & Vallespir, B. (2006). Enterprise interoperability measurement-Basic concepts. *Proceedings of the EMOI*, (1), 1–5. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.7162&rep=rep1&type=pdf

Energie-Nederland, & Netbeheer Nederland. (2011). *Energy in the Netherlands 2011*. *Chemistry & …*.

European Commission. (2005). *Towards Smart Power Networks* (pp. 1–40).

European Commission. (2006). *European SmartGrids Technology Platform* (pp. 1–37).

european network of transmission system operators for electricity. (2011). *The harmonised electricity market role model* (pp. 1–28). Brussels.

Evans, A., & Kent, S. (1999). Core Meta-Modelling Semantics of UML: The pUML Approach. In R. France & B. Rumpe (Eds.), *«UML»'99 — The Unified Modeling Language* (Vol. 1723, pp. 140–155). Springer Berlin Heidelberg. doi:10.1007/3-540-46852-8_11

Falbo, R. de A., Guizzardi, G., & Duarte, K. C. (2002). An ontological approach to domain engineering. *Proceedings of the 14th international conference on Software engineering and knowledge engineering* (pp. 351–358). New York, NY, USA: ACM. doi:10.1145/568760.568822

Fernández-Lopéz, M., & Gómez-Pérez, A. (2002). Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*, *17*(02), 129–156. doi:10.1017/S0269888902000462

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering. *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series* (pp. 33–40). Stanford: American Asociation for Artificial Intelligence. Retrieved from http://oa.upm.es/5484/

Ford, T. C., Colombi, J. M., Graham, S. R., & Jacques, D. R. (2007). A Survey on Interoperability Measurement. *Twelfth International Command and Control Research and Technology Symposium*. Newport, RI.

Gailly, F., & Poels, G. (2007). Towards Ontology-Driven Information Systems: Redesign and Formalization of the REA Ontology. In W. Abramowicz (Ed.), *Business Information Systems SE - 19* (Vol. 4439, pp. 245–259). Springer Berlin Heidelberg. doi:10.1007/978-3-540-72035-5_19

Gasevic, D., & Djuric, D. (2006). *Model Driven Architecture and Ontology Development*. Berlin Heidelberg: Springer-Verlag. doi:10.1007/3-540-32182-9

Geerts, G. L., & Mccarthy, W. E. (2000). The Ontological Foundation of REA Enterprise Information Systems. *Annual Meeting of the American Accounting Association* (pp. 127–150). Philadelphia, PA.

Gordijn, J., & Akkermans, J. M. (2003). Value-based requirements engineering: exploring innovative e-commerce ideas. *Requirements Engineering*, *8*(2), 114–134. doi:10.1007/s00766-003-0169-x

Gordijn, J., Osterwalder, A., & Pigneur, Y. (2005). Comparing two business model ontologies for designing e-business models and value constellations. *18th Bled eConference eIntegration in Action*. Bled, Slovenia.

Guizzardi, G. (2007). On ontology, ontologies, conceptualizations, modeling languages, and (meta) models. *Frontiers in artificial intelligence and applications*, *155*, 18.

Guizzardi, Giancarlo. (2005). *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, Enschede. Retrieved from http://doc.utwente.nl/50826/

Guizzardi, Giancarlo, & Wagner, G. (2005). Towards Ontological Foundations for Agent Modelling Concepts Using the Unified Fundational Ontology (UFO). In P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, & M. Winikoff (Eds.), *Agent-Oriented Information Systems II* (Vol. 3508, pp. 110–124). Berlin Heidelberg: Springer Berlin Heidelberg. doi:10.1007/11426714_8

Henderson-Sellers, B. (2011). Bridging metamodels and ontologies in software engineering. *Journal of Systems and Software*, *84*(2), 301–313. doi:10.1016/j.jss.2010.10.025

Hesse, W. (2008). Engineers Discovering the "Real World" — From Model-Driven to Ontology-Based Software Engineering. In R. Kaschek, C. Kop, C. Steinberger, & G. Fliedl (Eds.), *Information Systems and e-Business Technologies* (Vol. 5, pp. 136–147). Springer Berlin Heidelberg. doi:10.1007/978-3-540-78942-0_16

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), pp. 75–105. Retrieved from http://www.jstor.org/stable/25148625

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Retrieved April 29, 2013, from http://www.w3.org/Submission/SWRL/

IEEE. (1990). IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries.

Jacobsson, S., & Bergek, A. (2004). Transforming the energy sector: the evolution of technological systems in renewable energy technology. *Industrial and Corporate Change*, *13*(5), 815–849. doi:10.1093/icc/dth032

Jamasb, T., & Pollitt, M. (2005). *Electricity Market Reform in the European Union : Review of Progress toward Liberalization & Integration*.

Jarrar, M., & Meersman, R. (2009). Ontology Engineering – The DOGMA Approach. In T. Dillon, E. Chang, R. Meersman, & K. Sycara (Eds.), *Advances in Web Semantics I* (Vol. 4891, pp. 7–34). Springer Berlin Heidelberg. doi:10.1007/978-3-540-89784-2_2

Jötten, G., Weidlich, A., Filipova-Neumann, L., & Schuller, A. (2011). Assessment of flexible demand response business cases in the smart grid. *21 st International Conference on Electricity Distribution Frankfurt* (pp. 6–9). Frankfurt.

Kalfoglou, Y. (2001). Exploring Ontologies. *Handbook of Software Engineering and Knowledge Engineering: vol. 1: Fundamentals* (pp. 863–887). World Scientific Publishing. Retrieved from http://eprints.soton.ac.uk/260528/

King, R. L. (2008). Information services for smart grids. *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE* (pp. 1–5). IEEE.

Kok, J. K., Scheepers, M. J. J., & Kamphuis, I. G. (2010). Intelligence in Electricity Networks for Embedding Renewables and Distributed Generation. In R. R. Negenborn, Z. Lukszo, & H. Hellendoorn (Eds.), *Intelligent Infrastructures SE - 8* (Vol. 42, pp. 179–209 LA – English). Springer Netherlands. doi:10.1007/978-90-481-3598-1_8

Kok, J. K., Warmer, C. J., & Kamphuis, I. G. (2005). PowerMatcher: multiagent control in the electricity infrastructure. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (pp. 75–82). New York, NY, USA: ACM. doi:10.1145/1082473.1082807

Kok, K., Derzsi, Z., Gordijn, J., Hommelberg, M., Warmer, C., Kamphuis, R., & Akkermans, H. (2008). Agent-Based Electricity Balancing with Distributed Energy Resources, A Multiperspective Case Study. In R. H. Sprague (Ed.), *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)* (pp. 173–173). Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/HICSS.2008.46

Kok, K., Roossien, B., Macdougall, P., Pruissen, O. Van, Venekamp, G., Kamphuis, I. g., Laarakkers, J. A. W., et al. (2012). Dynamic Pricing by Scalable Energy Management Systems - Field Experiences and Simulation Results using PowerMatcher. *IEEE Power and Energy Society General Meeting 2012, IEEE*.

Kosanke, K. (2006). ISO Standards for Interoperability: a Comparison. In D. Konstantas, J.-P. Bourrières, M. Léonard, & N. Boudjlida (Eds.), *Interoperability of Enterprise Software and Applications SE - 6* (pp. 55–64). Springer London. doi:10.1007/1-84628-152-0_6

Lee, Y. T. (1999). Information modeling: From design to implementation. *Proceedings of the second world manufacturing congress* (pp. 315–321).

Lethbridge, T. C., & Laganière, R. (2005). *Object-Oriented Software Engineering* (2nd ed.). Berkshire: McGraw-Hill.

Majewska, M., Kryza, B., & Kitowski, J. (2007). Translation of Common Information Model to Web Ontology Language. In Y. Shi, G. Albada, J. Dongarra, & P. A. Sloot (Eds.), *Computational Science – ICCS 2007 SE - 53* (Vol. 4487, pp. 414–417). Springer Berlin Heidelberg. doi:10.1007/978-3-540-72584-8_53

Maniraj, V., & Sivakumar, D. (2010). Ontology Languages - A Review. *International Journal of Computer Theory and Engineering, 2010*, *2*(6), 1793–8201.

National Institute of Standards and Technology. (2012a). NIST Framework and Roadmap for Smart Grid Interoperability Standards.

National Institute of Standards and Technology. (2012b). PAP17: Facility Smart Grid Information Standard. Retrieved December 12, 2012, from http://collaborate.nist.gov/twiki-sggrid/bin/view/SmartGrid/PAP17FacilitySmartGridInformationStandard

Nguyen, V. (2011). *Ontologies and information systems: a literature survey*. Edinburgh, Australia. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA546186

Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Medical Informatics*. Retrieved from http://liris.cnrs.fr/alain.mille/enseignements/Ecole_Centrale/What is an ontology and why we need it.htm

OASIS Open. (2012a). Energy Interoperation Version 1.0. Retrieved from http://docs.oasis-open.org/energyinterop/ei/v1.0/energyinterop-v1.0.html

OASIS Open. (2012b). Energy Market Information Exchange (EMIX) Version 1.0. Retrieved from http://docs.oasis-open.org/emix/emix/v1.0/emix-v1.0.html

Object Management Group. (2012). Introduction to OMG UML. Retrieved November 29, 2012, from http://www.omg.org/gettingstarted/what_is_uml.htm

Pagani, G., & Aiello, M. (2011). Towards a Service-Oriented Energy Market: Current State and Trend. In E. M. Maximilien, G. Rossi, S.-T. Yuan, H. Ludwig, & M. Fantinato (Eds.), *Service-Oriented Computing* (Vol. 6568, pp. 203–209). Springer Berlin Heidelberg. doi:10.1007/978-3-642-19394-1_22

Panetto, H., & Molina, A. (2008). Enterprise integration and interoperability in manufacturing systems: Trends and issues. *Computers in Industry*, *59*(7), 641–646. doi:10.1016/j.compind.2007.12.010

Peffers, K., Tuunanen, T., Rothenberger, M. a., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. doi:10.2753/MIS0742-1222240302

Pinto, H.Sofia, Tempich, C., & Staab, S. (2009). Ontology Engineering and Evolution in a Distributed World Using DILIGENT. In S. Staab & R. Studer (Eds.), *Handbook on Ontologies* (pp. 153–176). Springer Berlin Heidelberg. doi:10.1007/978-3-540-92673-3_7

Pinto, Helena Sofia, & Martins, J. P. (2004). Ontologies: How can They be Built? *Knowledge and Information Systems*, *6*(4), 441–464. doi:10.1007/s10115-003-0138-1

Pipattanasomporn, M., Feroze, H., & Rahman, S. (2009). Multi-agent systems in a distributed smart grid: Design and implementation. *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES* (pp. 1–8). doi:10.1109/PSCE.2009.4840087

Quirolgico, S., Assis, P., Westerinen, A., Baskey, M., & Stokes, E. (2004). Toward a Formal Common Information Model Ontology. In C. Bussler, S. Hong, W. Jun, R. Kaschek, Kinshuk, S. Krishnaswamy, S. Loke, et al. (Eds.), *Web Information Systems – WISE 2004 Workshops SE - 2* (Vol. 3307, pp. 11–21). Springer Berlin Heidelberg. doi:10.1007/978-3-540-30481-4_2

Schrier, A., Van Bekkum, M., Krukkert, D., Verhoosel, J., & Roes, J. (2012). *MOSES : Model gebaseerde Ontwikkeling van SEmantische Standaarden*.

Schuster, R., & Motal, T. (2009). From e3-value to REA: Modeling Multi-party E-business Collaborations. *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on* (pp. 202–208). doi:10.1109/CEC.2009.58

Shanks, G., Tansley, E., & Weber, R. (2003). Using ontology to validate conceptual models. *Commun. ACM*, *46*(10), 85–89. doi:10.1145/944217.944244

Simmins, J. J. (2011). The impact of PAP 8 on the Common Information Model (CIM). *Power Systems Conference and Exposition (PSCE), 2011 IEEE/PES* (pp. 1–2). doi:10.1109/PSCE.2011.5772503

Smart Grid Coordination Group. (2012). *SG-CG / M490 / C _ Smart Grid Reference Architecture*.

Smith, B., & Welty, C. (2001). Ontology: towards a new synthesis. *Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001* (pp. 3–9). New York, NY, USA: ACM. doi:10.1145/505168.505201

Snoeck, M., Dedene, G., Verhelst, M., & Depuydt, A.-M. (1999). *Object-Oriented Enterprise Modelling with MERODE* (1st ed.). Leuven, Belgium: Leuven University Press.

Snoeck, M., Michiels, C., & Dedene, G. (2003). Consistency by Construction: The Case of MERODE. In M. Jeusfeld & Ó. Pastor (Eds.), *Conceptual Modeling for Novel Application Domains SE - 11* (Vol. 2814, pp. 105–117). Springer Berlin Heidelberg. doi:10.1007/978-3-540-39597-3_11

Stahl, T., & Voelter, M. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. (Wiley, Ed.)*John Wiley and Sons ISBN9780470025703* (p. 444). Wiley. Retrieved from

http://www.amazon.com/Model-Driven-Software-Development-Technology-Engineering/dp/0470025700

Tabors, R. D., Parker, G., & Caramanis, M. C. (2010). Development of the Smart Grid: Missing Elements in the Policy Process. *System Sciences (HICSS), 2010 43rd Hawaii International Conference on* (pp. 1–7). doi:10.1109/HICSS.2010.148

Uschold, M, & Gruninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge engineering review*, *11*(2), 93–136.

Uschold, Michael, & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, *33*(4), 58–64. doi:10.1145/1041410.1041420

Uslar, M., Specht, M., Rohjans, S., Trefke, J., & Vasquez González, J. (2012). The IEC Common Information Model. *The Common Information Model CIM* (Vol. 66, pp. 75–106). Springer Berlin Heidelberg. doi:10.1007/978-3-642-25215-0_3

Verhoosel, J. P. C., Rothengatter, D., Rumph, F. J., & Konsman, M. (2012). An Ontology for Modeling Flexibility in Smart Grid Energy Management. *Proceedings of the 3rd Workshop on Energy Efficient Buildings 2012*. Reykjavik.

W3C. (1999a). XSL Transformations (XSLT). Retrieved May 3, 2013, from http://www.w3.org/TR/xslt

W3C. (1999b). XML Path Language (XPath). Retrieved May 3, 2013, from http://www.w3.org/TR/xpath/

W3C. (2013). SPARQL 1.1 Overview. Retrieved April 9, 2013, from http://www.w3.org/TR/sparql11-overview/

APPENDIX A: PRACTICAL GUIDE FOR PRACTITIONERS

This appendix is intended to provide a practical manual for practitioners who want to apply the developed methodology. The steps should be executed in order and the phases can be approached in an iterative manner. Before the phases and steps are outlined, the tools and files that support the development process are described. Some helpful screenshots are added at the end of this appendix.

A.1 USEFUL TOOLS AND FILES

Two pieces of software are highly recommended to be used during the execution of the development methodology. The first software tool is called MERMAID. This tool can be found at http://merode.econ.kuleuven.ac.be/mermaid.aspx. It helps developing and drawing the dependency diagram and supports the building of the Object-Event-Table (OET). Both are part of the first development phase.

Although there are many other ontology (OWL) editor tools, Protégé is one of the most popular. Also, this tool was used in developing our own version of the REA upper ontology and in developing the microgrid ontology of chapter 5. In this guide also some screenshots are included from this tool, Protégé version 4.2.

The REA upper ontology as described in section 4.2.2, can be found in the attached file “rea-ontology.owl”. This ontology is the base of the ontology to be developed in the end. It can be loaded in Protégé to link concepts to the base concepts defined in this file.

A.2 PHASE 1: DETERMINE BASIC SHARED DOMAIN MODEL

A.2.1 IDENTIFY SCOPE

To limit the scope and to determine the granularity of detail this domain model will describe, first, competency questions should be posed. These questions should comprise of all the things the solution should be able to fulfil, and eventually explicitly do not have to be able to fulfil or to what extent concepts have to be defined. Two example competency questions from the microgrid case are shown in Example 1.

```
1. Can the solution facilitate the exchange of energy between energy consumer and
   supplier?
2. Can the solution balance energy demand and supply in blocks of 15 minutes?
```

Example 1: Competency questions

A.2.2 IDENTIFY AGENTS AND RESOURCES

All agents and resources involved in the domain of discourse should be identified and described in this step. An agent is a party involved in the interactions in the domain. A resource supports the business activities of the involved agents. Try to define the names and descriptions as unambiguous as possible to prevent misinterpretations. Example 2 shows a table describing some actors and resources identified in the microgrid case.

Agent / resource	Description
Consumer	Consumes energy. The load of energy has a certain flexibility over a given time period. It wants the best (cheapest) price for energy on the market.
Producer	Produces energy. The load of energy has a certain flexibility over a given time period. It wants the best (highest) price for energy on the market.
Energy	The (physical) energy to be delivered for consumption from producer to consumer. It can also be routed through a power grid via “energy transmissions”.
EnergyFlexibility	Represents the information a consumer and producer share with the FlexMarket to negotiate about their (near) future energy consumption and production. This information comprises of information about the power load required, the timespan that is required within the power needs to be produced or consumed and the actual duration the consumption or production once started.

Example 2: Descriptions of domain agents and resources

A.2.3 IDENTIFY COMMITMENTS

When the agents and resources have been identified, the commitments agents have in the domain need to be identified. The commitments should be able to be described in the form of the exchange of information (or information about an exchange of a physical concept), between exactly 2 agents: a sender and a receiver. In the cases where more than 2 agents or more than 1 information resource are involved, it is possible to decompose this commitment into 2 or more sub-commitments.

Example 3 shows a table describing some commitments identified in the microgrid case. After the commitments have been described, an existence dependency diagram can be drawn, eventually with help of the MERODE tool (see also Figure 32). Here, on the left side of the diagram the agents are drawn, on the right side the resources and in between the commitments. The agents, commitments and resources can be drawn in MERODE by using the “add object-type” tool from the toolbar. The commitments are existence dependent on two agents and one resource. Existence dependencies can be drawn using the “add dependency” tool from the toolbar. Example 4 shows an existence dependency diagram of the commitments of Example 3. After drawing the diagram, don’t close MERODE yet, as it can also be used for the next step.

Commitment	Description
ConsumptionFlexibilityOffer	In the smart grid consumers will send their current flexibility in power demands for a specific range of time in the future, on which the FlexMarket will reply a ConsumptionMarketOffer.
CongestionImposement	In the smart grid congestion limits apply. The DSO will send the FlexMarket updates about the exact congestions per transmission link in the smart grid.

Example 3: Description of commitments

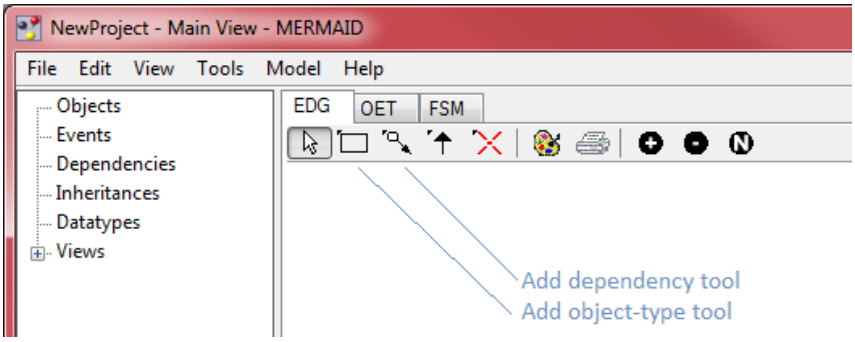
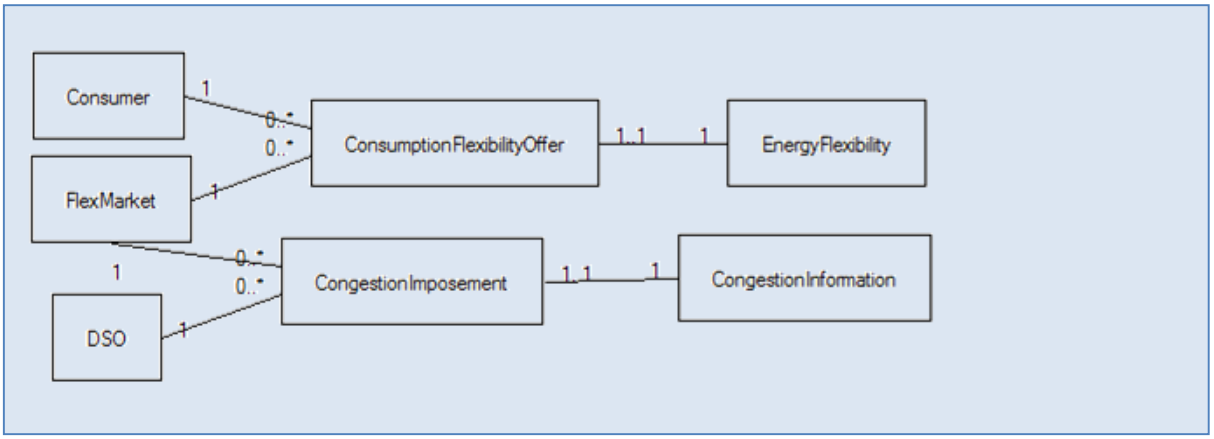


Figure 32: Developing an existence dependency diagram with MERMAID



Example 4: Existence dependency diagram

A.2.4 IDENTIFY EVENTS

To fulfil the commitments identified in the previous step, agents execute events. The events to be identified include both the sending and receiving of information. In a later step, the events will be further specified into at least one incremental event and at least one decremental event.

To provide an overview of all events in the domain and to relate them to the previously identified commitments, in MERODE an Object-Event-Table (OET) can be generated. By clicking on the “OET” tab above the toolbar, an empty OET will be shown with all entered agents, resources and commitments already in the right place. For each commitment one event should be created with the “add event” tool from the toolbar (see also Figure 33. By using the “add method” tool, an owned creation method should be created for the commitment the event describes. Automatically, methods of the involved agents and resource will appear after defining the owned creation method. An example OET from the microgrid case is shown in Example 5.

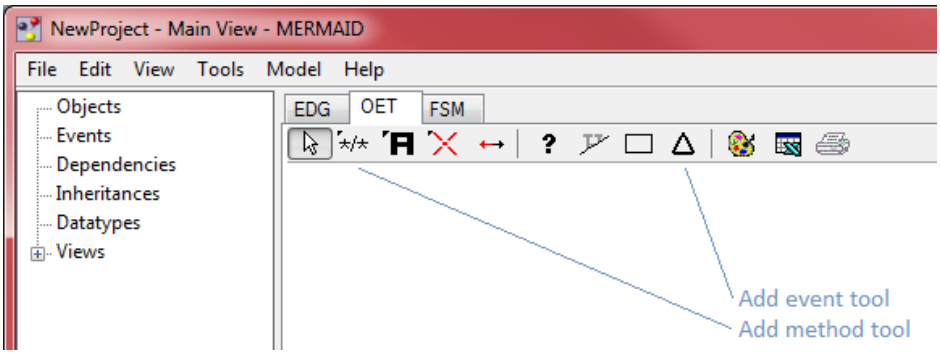


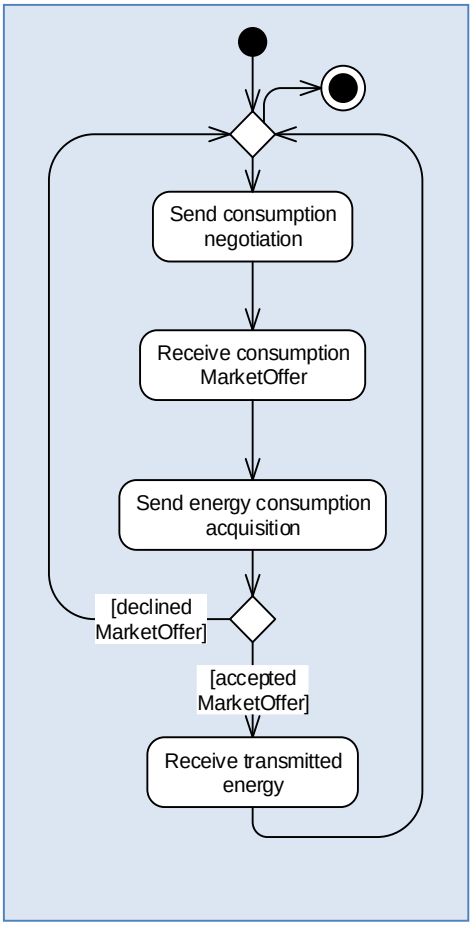
Figure 33: Developing an Object-Event-Table with MERMAID

	Consumer	FlexMarket	DSO	ConsumptionFlexibilityOffer	CongestionImposement	CongestionInformation	EnergyFlexibility
NegotiateConsumption	A/M	A/M		O/C			A/M
imposeCongestion		A/M	A/M		O/C	A/M	

Example 5: Event-Object-Table

A.1.5 MAKE UML ACTIVITY DIAGRAMS

The events identified in the previous step are bound to a specific sequence in the domain of discourse. For each domain agent an activity diagram should be drawn. It can happen that the same event occurs in more than one activity diagram. When the occurrence of an event is followed by a choice of one of a few events, the constraints when an event is executed needs to be defined. Example 6 shows an activity diagram from the microgrid case, where the consumer negotiates with the flex market the consumption of energy and in the end the receiving of the energy to be consumed.



Example 6: Activity diagram

A.2 BUILD ONTOLOGY BASE

A.2.1 IMPORT AND LINK AGENTS AND RESOURCES TO REA-ONTOLOGY

The agents and resources identified in the previous phase can be specified as subclasses of the rea-ontology classes AgentType and ResourceType. The easiest way to do this is to open the attached file containing the REA upper ontology in Protégé and use the “create class hierarchy” tool from the menu (see also Figure 34). In the tool the class AgentType or ResourceType can be selected as root class to create a class hierarchy. The names of all agents or resources can be entered in the next screen of the tool.

Please keep in mind that the reasoner (preferably HermiT) should run to ensure no incorrect concepts are defined. To select HermiT as reasoner, open the menu “Reasoner” and click on “HermiT”. The reasoner can be started by clicking on “Start reasoner” in the “Reasoner” menu. When incorrect information is entered in the ontology, this information will be marked red.

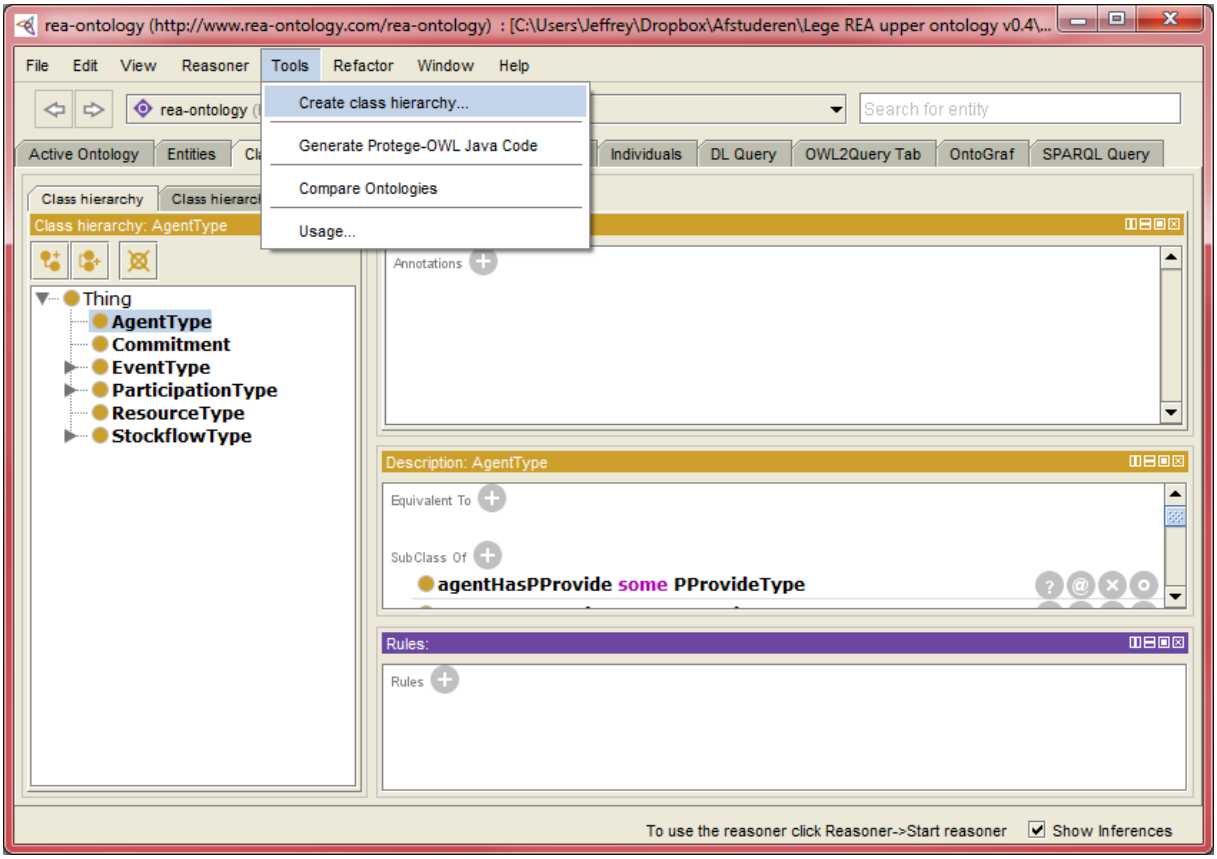


Figure 34: Create class hierarchy in Protégé

A.2.2 IMPORT AND LINK COMMITMENTS AND EVENTS TO REA-ONTOLOGY

The same steps need to be taken to link the commitments and events from the basic shared domain model. The events identified earlier now need to be "split" into one incremental event (receiving information) and one decremental event (sending information). Next to Commitments, DecrementalEventTypes and IncrementalEventTypes, also ParticipationTypes and StockflowTypes need to be specified. Participations indicate the relationship between agents and the events they participate in. Stockflows indicate the relationship between resources and the events they are sent or received. Figure 35 shows an example of the concepts of the microgrid ontology.

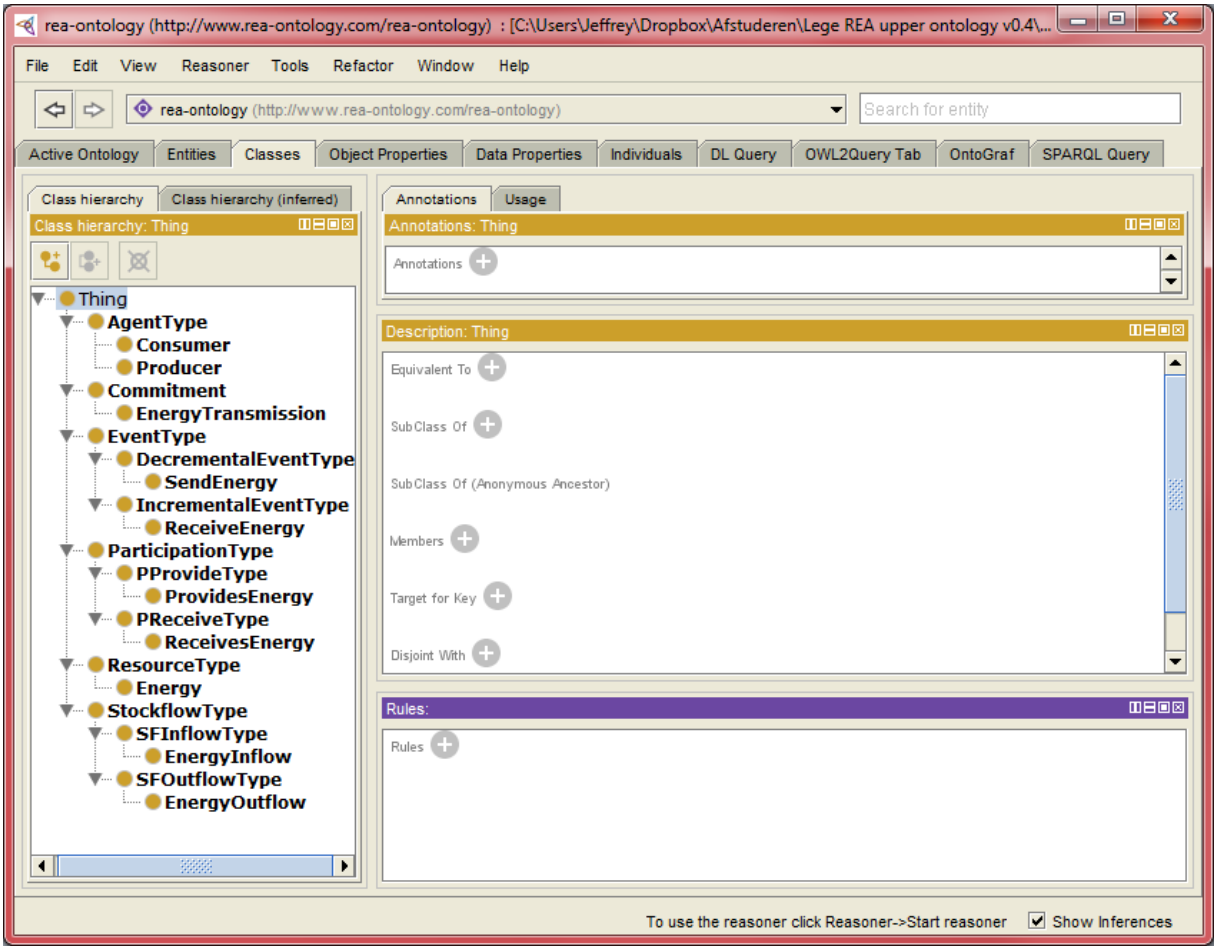


Figure 35: Example ontology in Protégé

After all concepts are added to the ontology, their properties inherited from the REA upper ontology need to be specialized to ensure only correct instances of the model can be created in the next phase. To add specialized properties, go to the “class” tab in Protégé and select a class on the left panel. To add a (specialized) property to this class press on the plus-sign next to “SubClass Of” in the right panel (see also Figure 36). Go to the “Object restriction creator” tab in the new window and select one of the properties inherited from the superclass in the left panel. In the right panel select the specialized concept that is related to the property. Make sure the right restriction type is set in the bottom panel. Figure 37 shows a screenshot. Repeat this until all REA-properties are specialized.

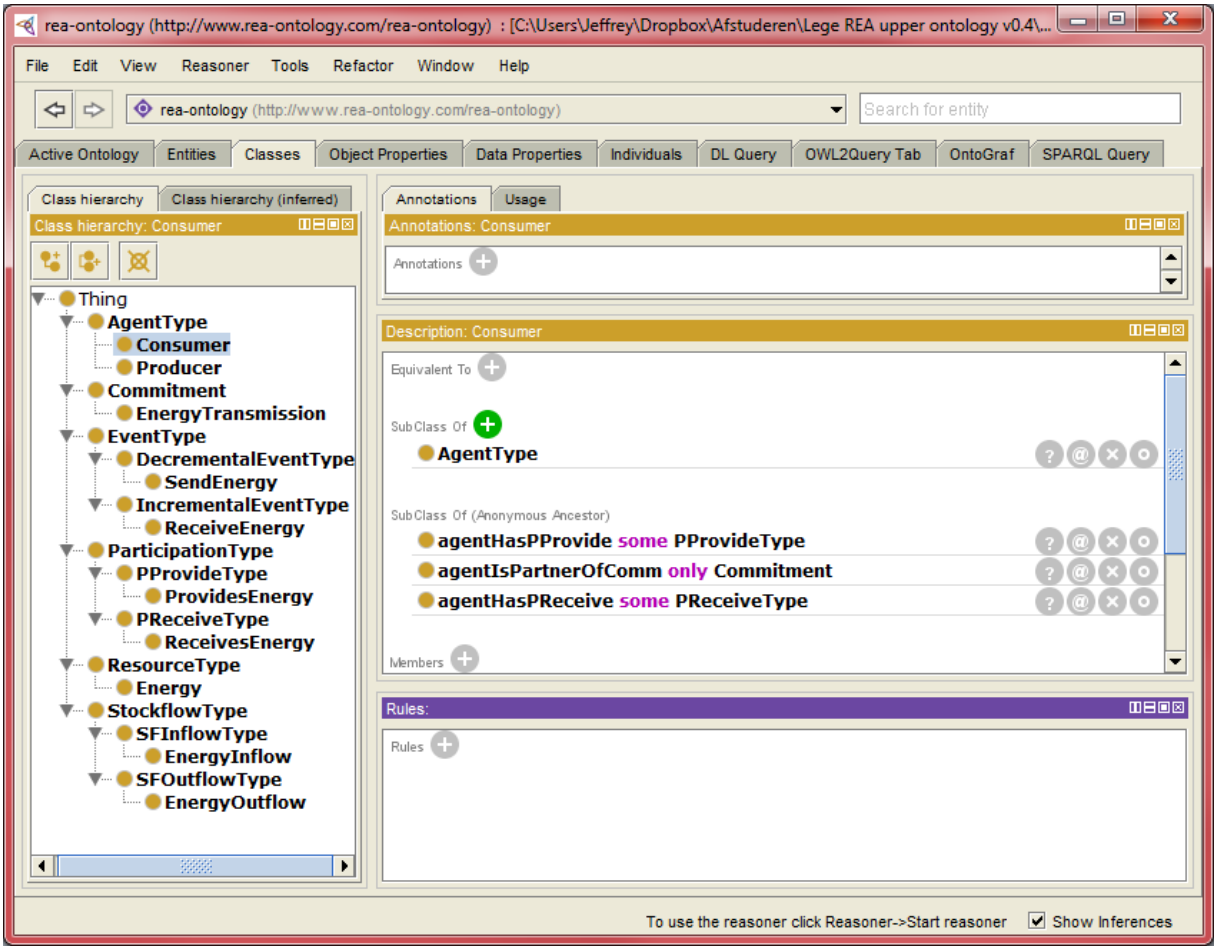


Figure 36: Specialize properties of a class in Protégé

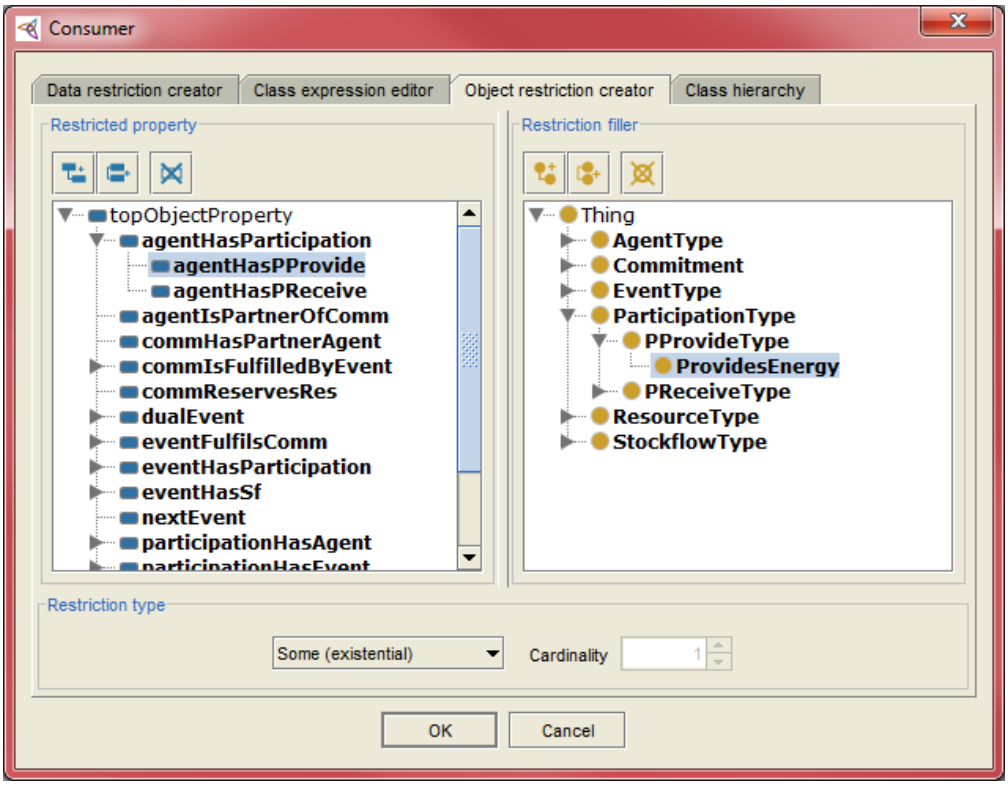


Figure 37: Add property to concept in Protégé

A.3 DEVELOP ONTOLOGY

A.3.1 REUSE AND INTEGRATE EXISTING ONTOLOGIES

In case concepts of another ontology can be used, these can be directly imported in the ontology and then linked to the REA upper ontology. In Protégé ontologies can be imported in the “Active Ontology” tab and pressing the plus-sign next to “Direct Imports” (see also Figure 38).

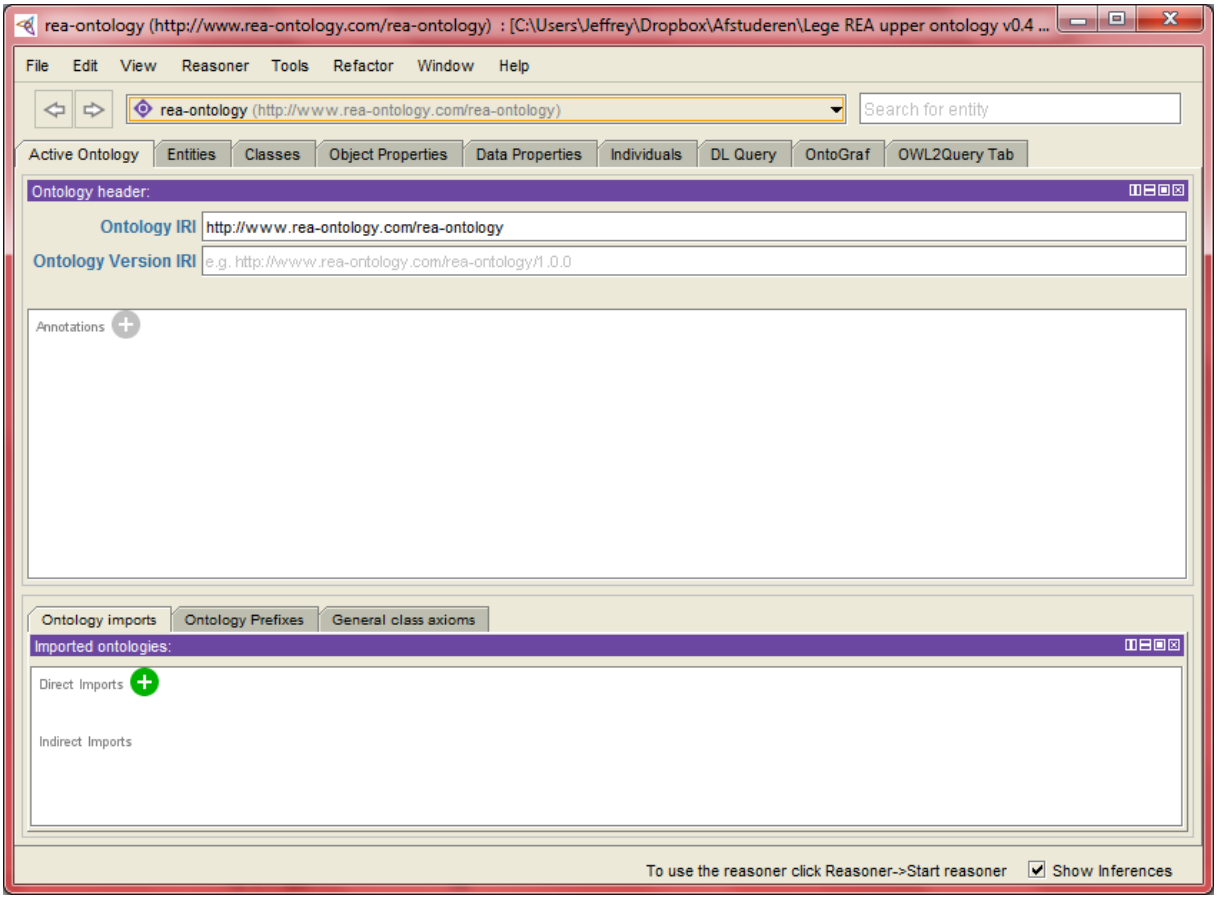


Figure 38: Import ontologies in Protégé

A.3.1.1 DETERMINE PROPERTIES OF AGENTS AND RESOURCES

Agent- and resource-specific properties relevant for the domain model can be specified as data properties. Based on these data properties rules can be created in the next step. Note that the data properties allocated to a resource are seen as information that needs to be send when the resource is exchanged.

Every data property has to be first defined in the “Data Properties” tab of Protégé. The overview of the data properties and the buttons to add or remove a data property are available in the left panel. In the right panel the Domain to which classes the property applies and the Range which data type the property is can be specified. The domain can be specified by selecting one or more concepts in the “Class hierarchy” tab of the selection window. The range can be specified by selecting a data type from the “Built in datatypes” tab. Eventually the data type can be restricted to certain values by editing the range (see also Figure 39). An example restriction is “long[>=0]” to restrict all values to be larger than or equal to 0. The defined data properties can be added in the same way as regular properties are added (see also Figure 36).

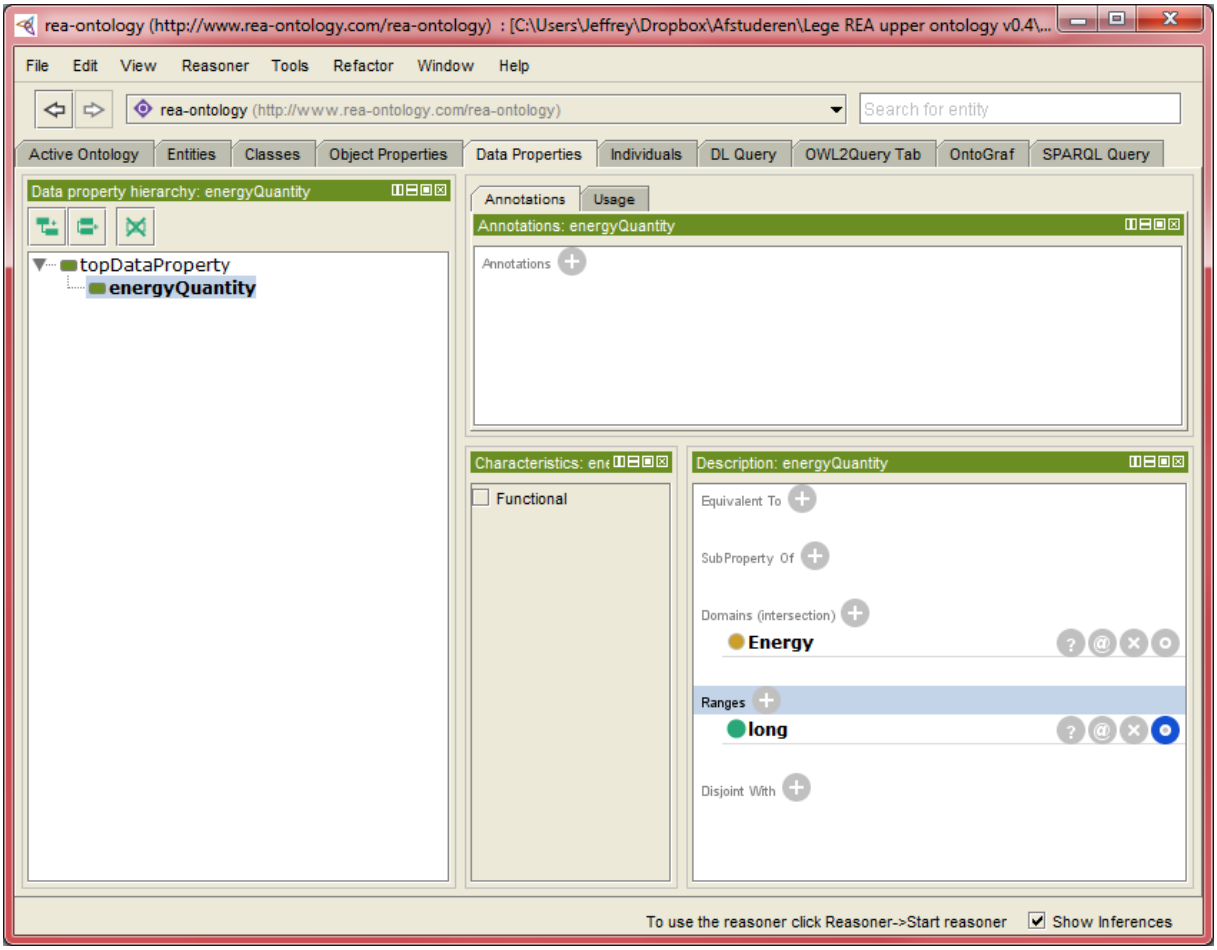


Figure 39: Data property in Protégé (range edit button highlighted)

A.3.1.2 DETERMINE INFORMAL CONSTRAINTS

To limit the instantiations of the developed ontology model to only configurations valid in the real world, constraints apply on the domain model. These constraints can be written down to informal constraints, which can be formalized in the next step. The constraints can be written down in a numbered list, as shown in Example 7.

1. `A flex offer is declined when the offering AgentType (consumer or producer) has a minimumFlexLoad that is higher than the maximumLoadCongestion imposed on this AgentType.`
2. `Of a flex offer, the minimumFlexLoad is always smaller than the maximumFlexLoad.`

Example 7: Informal constraints from the microgrid case

A.3.1.3 DETERMINE FORMAL CONSTRAINTS

The informal constraints from the previous step can be formalized into SWRL rules. The SWRL syntax can be found in http://www.w3.org/Submission/SWRL/. Example 8 shows the constraints of Example 7 formalized. Protégé has native support for SWRL. Before rules can be added in the editor, the "Rules" view should be added to the classes tab of Protégé. This can be done by first selecting the "Rules" view from the menu (see Figure 40 where to find it), then clicking on the bottom part of the Protégé window to add the view. With the plus-sign new rules can be added.

```
1. EnergyFlexibility(?flexoffer), minimumFlexLoad(?flexoffer, ?minload),
   maximumFlexLoad(?flexoffer, ?maxload), resHasSfOutflow(?flexoffer, ?fooutflow),
   sfOutflowHasDecrEvent(?fooutflow, ?decrevent), decrEventHasPProvide(?decrevent,
   ?providep), pProvideHasAgent(?providep, ?agent), LoadConstraint(?lc),
   loadConstraintAppliesToAgent(?lc, ?agent), maximumLoadCongestion(?lc, ?maxload) ->
   lessThanOrEqual(?minload, ?maxload)
2. EnergyFlexibility(?flexoffer), minimumFlexLoad(?flexoffer, ?minflex),
   maximumFlexLoad(?flexoffer, ?maxflex) -> lessThanOrEqual(?minflex, ?maxflex)
```

Example 8: Formal contraints from the microgrid case

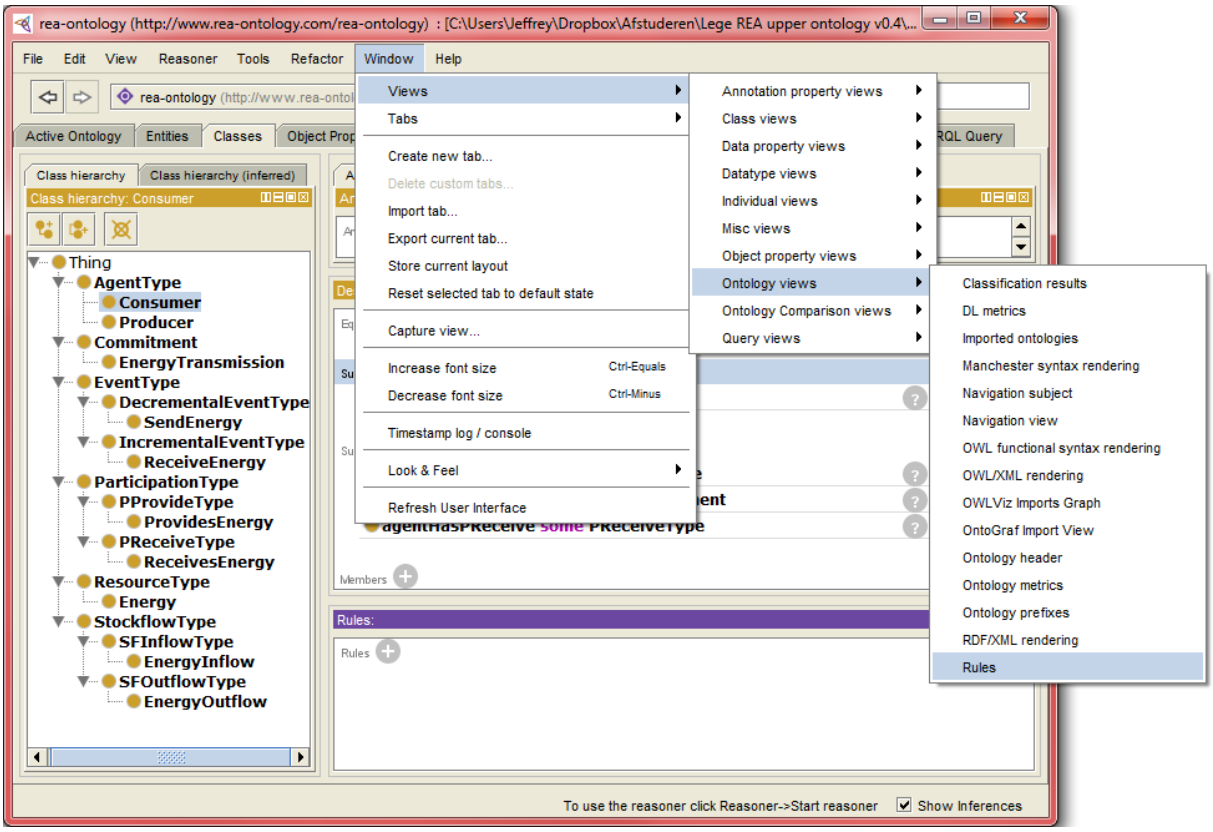


Figure 40: The "Rules" view in Protégé

A.3.1.4 DETERMINE CLASS INSTANCES

At this step the domain concepts should be fully specified. In this step instances of the domain concepts can be created. Instances specify the concrete instances of the class types specified in the ontology, e.g. “House1” is an instance of Consumer in the domain of discourse of the example in Figure 41. Protégé can work with individuals in the “Individuals” tab, as shown in Figure 41. For every instance a “Type”, “Object property assertions” and eventually “Data property assertions” need to be defined, in Protégé this can be done on the right panel of the “Individuals” tab.

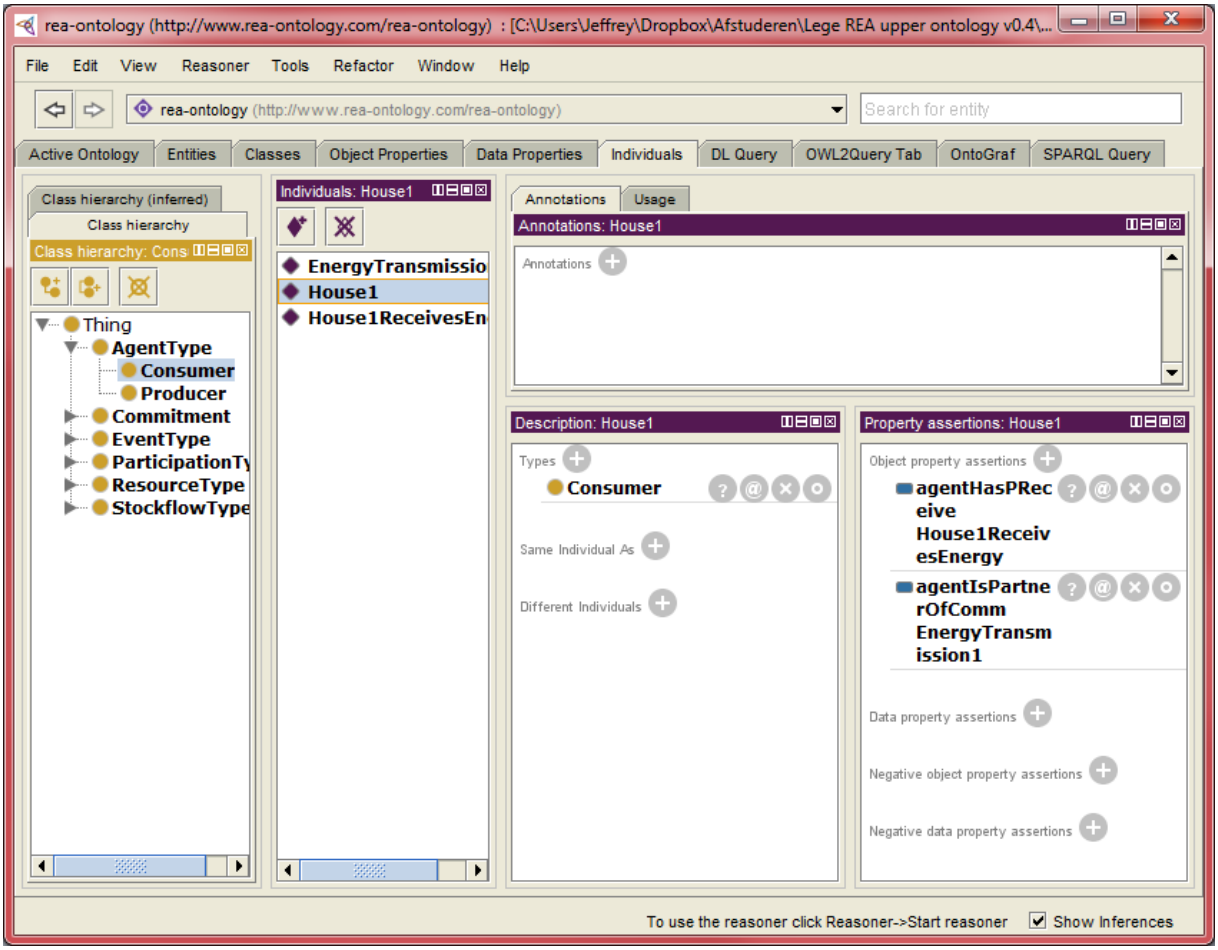


Figure 41: Individuals in Protégé

A.4 DETERMINE TECHNOLOGY-SPECIFIC SOLUTION

Message structures can be derived from the ontology using the SPARQL-query defined in Query 2, which is repeated below in Query 4. Protégé supports the execution of SPARQL-queries. To execute the query in Protégé, first the “SPARQL Query” tab needs to be enabled by selecting the tab from the menu, as shown in Figure 42.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX my: <http://www.rea-ontology.com/rea-smartgrid#>
SELECT ?eventtype ?messagefield ?sendertype ?receivertype ?nexteventtype
WHERE {
        ?decrementalevent rdf:type ?eventtype.
        ?eventtype rdfs:subClassOf* my:DecrementalEventType.
        ?allsfs rdfs:subPropertyOf my:eventHasSf.
        ?decrementalevent ?allsfs ?relatedsfs.
        ?allrelatedresources rdfs:subPropertyOf my:sfHasRes.
        ?relatedsfs ?allrelatedresources ?resource.
        ?messagefield rdfs:subPropertyOf* my:message.
        ?resource ?messagefield ?message.
        ?allparticipations rdfs:subPropertyOf my:evHasP.
        ?decrementalevent ?allparticipations ?participations.
        ?allagents rdfs:subPropertyOf my:pHasA.
        ?participations ?allagents ?sender.
        ?sendertype rdfs:subClassOf my:AgentType.
        ?sender a ?sendertype.
        ?allDualities rdfs:subPropertyOf my:dualEvent.
        ?decrementalevent ?allDualities ?dualevent.
        ?dualparticipations rdfs:subPropertyOf my:evHasP.
        ?dualevent ?dualparticipations ?dualparticipation.
        ?alldualagents rdfs:subPropertyOf my:pHasA.
        ?dualparticipation ?alldualagents ?receiver.
        ?receivertype rdfs:subClassOf my:AgentType.
        ?receiver a ?receivertype.
        ?decrementalevent my:nextEvent ?nextevent.
        ?nexteventtype rdfs:subClassOf* my:EventType.
        ?nextevent a ?nexteventtype.
}
```

Query 4: SPARQL-query to retrieve the relevant information required to develop message structures

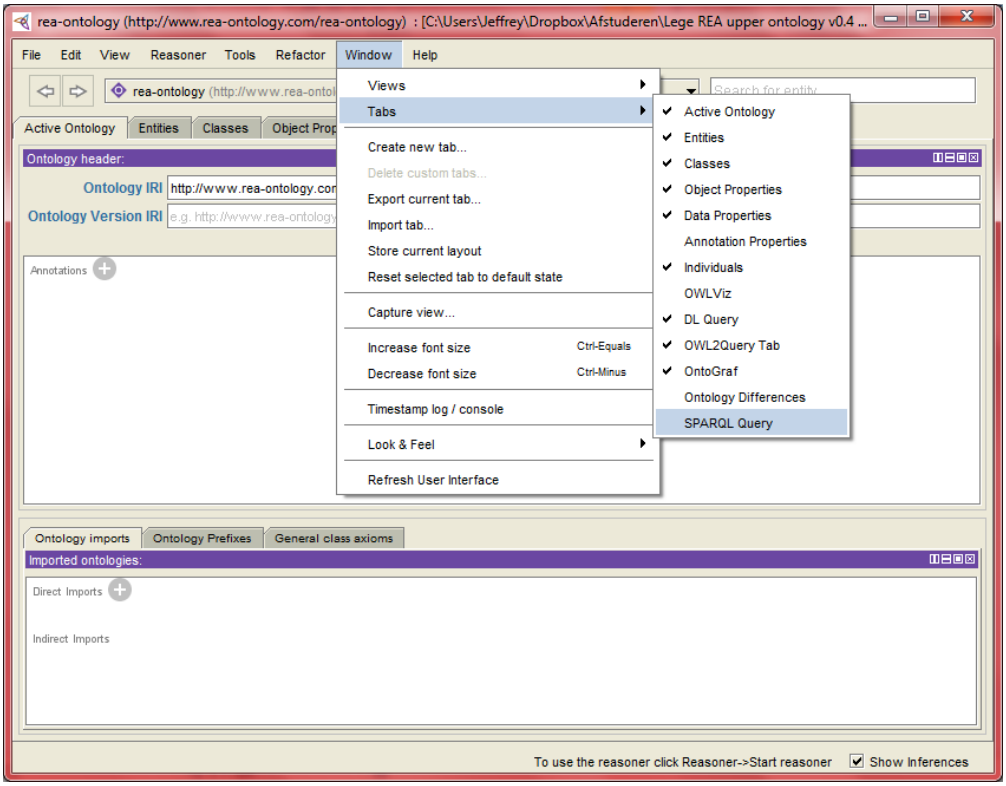


Figure 42: The SPARQL Query tab in Protégé