TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

A Universal Scheme for Modeling Energy Systems

Ciechanowicz, David, Aydt, Heiko, Lees, Michael, Knoll, Alois and Hamacher, Thomas

TUM-I1341

A Universal Scheme for Modeling Energy Systems

David Ciechanowicz[a,d], Heiko Aydt[a], Michael Lees[b], Alois Knoll[c], Thomas Hamacher[d]

[a]TUM CREATE Ltd.
Singapore
{david.ciechanowicz | heiko.aydt}@tum-create.edu.sg

[b]Nanyang Technological University (NTU)
School of Computer Engineering
Singapore
mhlees@ntu.edu.sg

[c]Technische Universität München (TUM)
Institute for Informatics VI
Robotics and Embedded Systems
Germany
knoll@in.tum.de

[d]Technische Universität München (TUM)
Energy Economy and Application Technology
Germany
thomas.hamacher@tum.de

Abstract

For the modeling of energy systems a language is needed that is conceptually well-defined and gives a graphical representation. The most prominent approach, the *extended Reference Energy System* (eRES) suffers from shortcomings in the conceptual definition. Therefore, the alternative language *Universal Scheme for modeling Energy Systems* (USES) is being developed. The advantage of USES is its clearly defined concepts and their relationships between each other. USES is minimal in respect to its concepts meaning no unnecessary exceptions are introduced. A complete formal description of the syntax of USES taking advantage of both, graph theory and the UML class diagram language is created in this paper while also showing its usage on a real-world example.

Keywords: Modeling Energy Systems, Universal Scheme, USES, Formal specification, Meta model

1 Introduction

Energy is critical to the social, economic, and technical development of mankind. Different stages have to be run through to get the primary energy occurrences of nature in a usable form to its consumers. Together, all involved parties in this process form a network of the various technologies in which extraction, refinement, conversion, transportation, distribution, and utilization of different forms of energy take place to provide a set of services [BHZ04]. Such a network is called an *energy system*. It involves multiple interconnected energy chains which are to some extent competing against each other since more than one path along the energy chains leads to the provision of the same energy service. An energy chain consists of sequential series of linked stages, alternating commodities (e.g. energy goods) and processes (e.g. energy conversions). The latter include the above mentioned links starting from extraction and ending with utilization, but are not limited to them.

A simple example of a general energy chain is depicted in Figure 1. The illustrated steps that are to be taken to transform any primary energy (in this example crude oil) into a useful energy (heat) are neither exhausting nor are all of them necessary in every case. The storage process for example is non-compulsory, whereas more than one utilization process may be chained one after another. It is the purpose of an energy system to fulfill the demand for energy services which are, according to Figure 1, a combination of various input factors. These include the used technologies, infrastructure, labor, materials and energy carriers which are, as stated above, partly substitutable against each other.

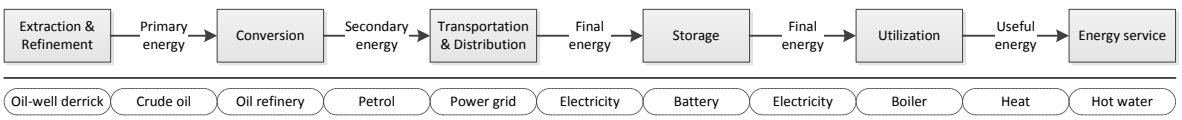

Figure 1: General energy chain from the extraction to the provision of energy services.

Energy systems can be of paramount complexity, especially when investigating high-resolution, large-scale energy systems with a huge amount of interconnected energy chains. Nowadays, the traditional architecture of a

power system, a specialized variant of an energy system only considering energy chains involving electrical energy, goes to its limits. It is therefore superseded by smart grids with diverse, intermittent and decentralized energy sources. In [YaIs02] super distributed energy systems are described which are controlled decentralized and autonomously. As stated in [Hi07] the paradigms of (energy) systems are changing from a system to a *system-of-systems* (SoS) approach, from being disciplinary to multidisciplinary and from an optimal to an adaptive approach. To understand, investigate, and plan the functionality of such complex energy systems simulations are needed which are based on computational models

Large-scale simulations of complex energy systems are compute intensive and thus require efficient simulation models. The most common existing modeling scheme, the *extended Reference Energy System* (eRES) approach [BSS98], gives a simplified image of the reality in a conceptual and representational way. Its syntax, however, is inconsistent, complex, and bears a lot of exceptions. In this paper the *Universal Scheme for modeling Energy Systems* (USES), an alternative language with a simple graphical representation and a well-defined syntax, is thus proposed.

Having a clear and well-defined formal syntax definition of a language offers several advantages. From an implementation perspective, simulation models can be easily verified against the formal specification. Furthermore, a standardized method for modeling energy systems can be established which facilitates interoperability of different simulation models and exchange of models and data between different research groups. Another advantage of having a formal specification is the ability to easily generate synthetic power systems that can be used for simulation-based what-if scenario analysis.

Section 2 focuses on the conceptual and representational definition of USES. Besides an informal syntax definition two formal ones, using graph theory and the UML class diagram language, are given in Section 3. After exemplarily converting a real-world energy system into USES in Section 4 its advantages as well as possible applications of USES regarding model verification and building software systems are deliberated in Section 5 which concludes this paper and gives an outlook on possible further application contexts of USES.

2 Conceptual and Representational Description

One modeling language has exactly one conceptual aspect while having one to many representational aspects. In this section the conceptual description as well as the default graphical representation of USES is depletive presented. Other graphical representations may be introduced keeping the conceptual description. By means of an abstract energy system all of the concepts and most of their relationships among each other are depicted by concrete elements in Figure 2 taking advantage of the graphical USES representation.

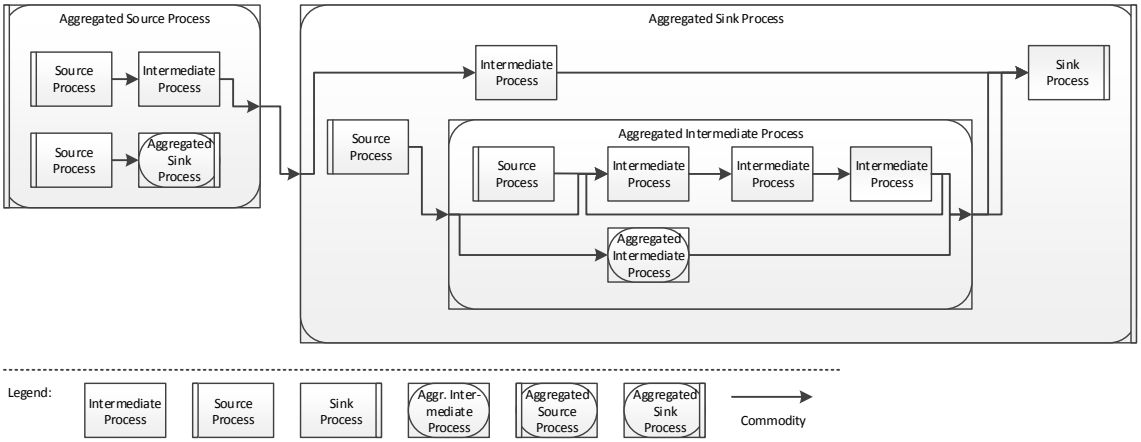

Figure 2: Abstract energy system showing the relationships between all of the defining elements of USES including their sub-elements by taking advantage of the default graphical USES representation.

Since the syntax of USES is based on graph theory networks of any context can be modeled with USES. It is indeed universal in respect to the use of its expressions and therefore in its semantic but the description of the semantics in this context is limited to the use-case of modeling energy systems. USES avails itself of the concept of inheritance [Se12]. Elements can be refined or specialized, respectively. The specialized element always inherits the properties and attributes of their respective aggregated element, and may possesses additional properties or attributes to provide the desired specialization.

2.1 Process

A *Process* is a representation of a physical device which transforms commodities (see below) into other commodities. It can be distinguished between energy supply processes and processes that make use of the supplied energy. The first group may consist of power plants, refineries, transmission and distribution infrastructure, energy storage devices among others. The second group is comprised of all electronic devices, such as refrigerators, electric vehicles, or computers, for example. They can have multiple inputs and multiple outputs. Processes are generalized. For a usage they have to be further specialized into intermediate, source, and sink processes (see below) due to a different set of properties. Generalized processes are only conceptually used in this form. Only the specialized variants have a graphical representation.

- **Intermediate Process**
  An *intermediate process* may have multiple inputs and multiple outputs. Intermediate processes are used whenever the transformation of the inputs to the outputs can be easily described, e.g. in form of one or more equations in a mathematical model. They are also generalized and can be used as such but can indeed be further specialized into aggregated intermediate processes (see below). Figure 3 shows the default representation of an intermediate process.

Figure 3: Default representation of an intermediate process.

- **Source Process**
  In a USES diagram every modeled energy chain has to start with a *source process* having only outputs but no inputs. The energy chain ends with a sink process (see below). Source processes therefore provide the outgoing commodity by e.g. harvesting or generating it. They are also generalized and can be used as such but can indeed be further specialized into aggregated source processes (see below). Figure 4 shows the default representation of a source process.

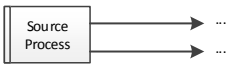

Figure 4: Default representation of a source process.

- **Sink Process**
  In a USES diagram every modeled energy chain has to end with a *sink process* having only inputs but no outputs. The energy chain starts with a source process. Sink processes therefore consume the ingoing commodity by e.g. converting it into a service. They are also generalized and can be used as such but can indeed be further specialized into aggregated sink processes (see below). Figure 5 shows the default representation of a sink process.

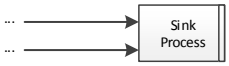

Figure 5: Default representation of a sink process.

- **Aggregated Process**
  In a USES model an *aggregated source/intermediate/sink process* behaves as a process in respect to the relationship with other elements on the same hierarchical level but additionally serves as a host for an

energy sub-system which resides on a below lying layer as regards content. The energy sub-system also consists of interconnected commodities and processes following the same USES rules and therefore forms an own USES model. The usage of aggregated processes is transparent without any exceptions in the syntactical rule set. Aggregated processes, against its non-aggregated counterparts, should be used when the transformation of the inputs to the outputs is more complex and cannot or should not be expressed in simple mathematical equations. Figure 6 shows the default representation of an aggregated source, intermediate, and sink process.

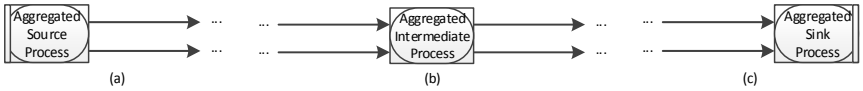

Figure 6: Default representation of an aggregated (a) source, (b) intermediate, and (c) sink process.

2.2 Commodity

Commodities are produced and consumed by other processes. They are defined as material streams or sets that are quantifiable. Commodities can be distinguished into energy goods or carriers that flow in the system, emissions, or waste, but are not limited to them. Energy goods may be primary energy sources like coal or crude oil or up to final energy vectors like gasoline or heat. Each commodity is of a specific type only allowing a connection between two processes capable of providing or consuming this commodity type. The connection is directed without having to introduce dedicated process interfaces. Figure 7 shows the default representation of a commodity.

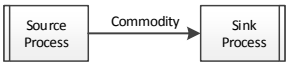

Figure 7: Default representation of a commodity.

When representing bi-directional commodity flows the concept of having directed connections, where inputs are connected to the left and outputs are connected to the right of a process, quickly becomes cumbersome. By introducing an alternative graphical representation without having to change the conceptual definition of USES, commodities may also be drawn as bi-directional connections. The graphical representation shown in Figure 8a and 8b are therefore conceptually equal. Without any information loss the arrows in bi-directional connections can be omitted like in Figure 8c.

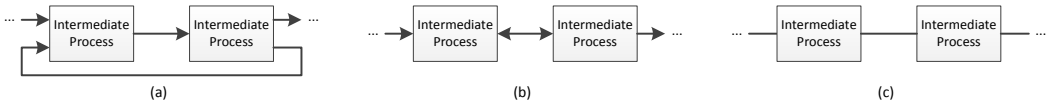

Figure 8: Alternative representation of a bi-directional commodity.

Although a more intuitive way for drawing commodities can be found in *Sankey* diagrams this solution is not suitable when creating USES diagrams in the first place. Being a representation of a real model a USES diagram is independent from any data model and therefore from any concrete flow quantities that could be used for the width of the lines. Nevertheless, this modified Sankey-USES representation, which is a combination of a real model and quantitative data and no longer a pure real model, can be used in the second place after specifying the flow quantities to give a more detailed graphical representation of the modeled energy system together with its data. Likewise, using information of a *Geographic Information System* (GIS), a GIS-USES or a Sankey-GIS-USES representation can be drawn in the second place utilizing stored geographical information for each element. More information on Sankey diagrams and on GIS can be found in [An12, pp. 128] and [De09] respectively.

3 Syntax

A *language* or synonymously a *notation* is a possibly infinite set of expressions that is used to syntactically represent information or to communicate, respectively [HaRu00]. The two parts of a language are the syntactic notation (*syntax*) and their meaning (*semantic*). The syntax defines the structure of the language, its expressions as well as their relationships among each other. The semantic of a language deals with the meaning of each of its expressions. It assigns an unambiguous meaning to each syntactically allowed phrase. The remainder of this section deals with the description of the syntax on an informal and a formal level, the latter one using graph theory and the UML class diagram language.

3.1 Informal

The concepts of USES and their syntactical relationships among each other have to follow certain rules which have partly already been delineated in the description of each of the concepts in Section 2. It is vital for any formal syntax specification to have an overview of the whole rule set and not to omit any rule. Therefore, in the remainder of this section the syntax of USES is textually defined in a structured way as precisely as natural language permits. Relationships are denoted in the multiplicity-notation of the class diagram language of the UML which is also later used in the meta model in Section 3.2.2.

As, for any later implementation, the mentioned concepts can have multiple different attributes attached to them only attributes that are mandatory for the definition of the USES syntax are named. Among these attributes are the *id* (unique number identifying every element), the *shortName* (name identifying every element in its graphical representation), the *scope* (the id of the aggregated process or USES model the element is part of on the hierarchical above lying layer), and the *idCommodityType* (unique number for every different type of commodities). Any later extension regarding attributes can easily be done without conflicts at the appropriate place in the syntax definition given below and in the meta model presented in Section 3.2.2.

1. *USES Model*
    a. 1 USES model consists of $1..*$ processes and $1..*$ commodities.
    b. 1 USES model is part of $0..1$ aggregated elementary, $0..1$ aggregated source, or $0..1$ aggregated sink process.
    c. Attached attributes: id (integer, primary key, read-only), shortName (string).
2. *Processes*
    a. 1 process is part of $0..*$ USES models.
    b. Processes are specialized into elementary, source, and sink processes. The specialization is disjoint and total, meaning 1 process has a relationship with the multiplicity of 1 to a specialized process.
        i. Elementary, source, and sink processes can be specialized into aggregated elementary, source, and sink processes. The specialization is disjoint and partial, meaning 1 elementary, source, or sink process has a relationship with the multiplicity of $0..1$ to a specialized elementary, source, or sink process.
            1. 1 aggregated elementary, source, or sink process consists of 1 USES models.
    c. Attached attributes: id (integer, primary key, read-only), shortName (string), scope (integer), idInputCommodityTypes (List<integer>), idOutputCommodityTypes (List<integer>)
3. *Commodities*
    a. 1 commodity is part of $0..*$ USES models.
    b. 1 commodity connects 2 processes. Both processes have to have the same scope. The source/sink process has to include the same idCommodityType as the commodity in its idOutputCommodityTypes/idInputCommodityTypes attribute.

c. Attached attributes: id (integer, primary key, read-only), shortName (string), scope (integer), idCommodityType (integer).

   i. The scope equals the scope of either connected processes.

3.2 Formal Syntax Definition

One of the advantages of USES is its clear and minimal syntax definition without unnecessary complex structures, relationships, or exceptions. This makes an easy formal specification possible. The benefit of a formal syntax specification of a modeling language has an effect when implementing any model using this language in software. The topologic consistency of the created model can then easily be verified against its specification which increases confidence in the system. In addition, a standardized method for modeling energy systems can be established on a broad basis. Exact specifications also enhance the exchange of data between different systems and increase data quality. The remainder of this section deals with the formal description of the syntax of USES using graph theory and the UML class diagram language.

3.2.1 Graph Theory

A USES diagram is a finite, weighted, and directed graph G consisting of a set of vertices V and edges E. The graph G may have self-egdes, multi-edges, and/or cycles but is not necessarily connected. Hierarchical structures, represented by aggregated processes, are also possible and described in the end of this section. The explanations given in Section 2 suggest that vertices may be processes while edges are commodities. Starting with the general definition of G in (1), the mentioned properties of G will be consecutively described to form the formal syntax specification.

$$G = (V, E) \tag{1}$$

The graph G is finite because the cardinality of both sets, vertices V and edges E, are finite. This is depicted in (2).

$$|V| < |\mathbb{N}| \quad \text{and} \quad |E| < |\mathbb{N}| \tag{2}$$

In a USES diagram the set of vertices V consists of processes v_i as can be seen in (3).

$$V = \{v_0, v_1, \dots, v_{n-1}, v_n\} \tag{3}$$

One vertex can have multiple edges connected as input and output, respectively. The number of edges connected to a vertex is called degree [Ne10, pp.133]. In a directed graph the in-degree deg^{in} and the out-degree deg^{out} of each vertex can be distinguished and calculated as shown in (4) and (5). Equation (4) defines the in-degree based on the intensional definition of set theory as well as giving a mathematical definition based on the adjacency matrix of while (5) is doing the same of the out-degree. The purpose of the adjacency matrix is to represent the network mathematically instead of graphically [Ha95, pp.76 and Ne10, pp.110]. If the in-degree (out-degree) is zero, the respective vertex represents a source (sink) process. In any other case the vertex is an intermediate process.

$$deg^{in}(v_i) = |\{v_i' | (v_i' \rightarrow v_i) \in V\}| \quad \text{or} \quad deg^{in}(v_i) = \sum_{j=1}^{n} A_{ij} \tag{4}$$

$$deg^{out}(v_i) = |\{v_i' | (v_i \rightarrow v_i') \in V\}| \quad \text{or} \quad deg^{out}(v_j) = \sum_{i=1}^{n} A_{ij} \tag{5}$$

The graph G is a vertex- and edge-weighted graph. According to (6) each edge e_{ij} (see below) is assigned a number representing the value of its *idCommodityType* attribute a^{ct}. According to (7) each vertex v_i is assigned a number representing the value of its *scope* attribute a^{scp} and two lists of numbers, the *idCommodityType* attributes for possible input a^{ctIn} (*idInputCommodityTypes*) and output commodities a^{ctOut}

(*idOutputCommodityTypes*). The generation of one number uniquely representing the desired value(s) is done according to Gödel numbering [Fr05].

$$f_e^{enc}(a^{ct}): e_i \rightarrow \mathbb{N} \tag{6}$$

$$f_v^{enc}(a^{scp}, a^{ctIn}, a^{ctOut}): v_i \rightarrow \mathbb{N} \tag{7}$$

The Gödel number that is assigned to each vertex and each edge representing its weight can be clearly decoded using (8) and (9).

$$f_e^{dec}(\mathbb{N}): e_i \rightarrow \{a^{ct}\} \tag{8}$$

$$f_v^{dec}(\mathbb{N}): v_i \rightarrow \{a^{scp}, a^{ctIn}, a^{ctOut}\} \tag{9}$$

The set of edges E, where each edge describes an ordered pair of vertices, contain the links between all vertices in the graph as depicted in (10).

$$E \subseteq V \times V \tag{10}$$

An edge e_{ij} as defined in (11) can only exist between vertices having the same value for the *scope* attribute a^{scp}. In addition, the list of values of the *idCommodityType* attribute for possible output (input) commodities a^{ctOut} (a^{ctIn}) of the source (sink) vertex v_i (v_j) has to include the value of the *idCommodityType* attribute a^{ct} of the edge e_{ij}.

$$e_{ij} = (v_i, v_j) \tag{11}$$

with

$$e_{ij} \in E$$
$$v_i, v_j \in V$$
$$v_i: a^{scp} = v_j: a^{scp}$$
$$\bigcup_m v_i: a_m^{ctOut} \cup \bigcup_n v_j: a_n^{ctIn} \cup a^{ct} \neq \emptyset$$

Any vertex might represent an aggregated vertex whose sub-graph is a complete USES model according to this syntax specification. Figure 9 to 11 show an example of an energy system being partly aggregated and disaggregated to demonstrate how hierarchical structures can be implemented in G.

(a) (b)

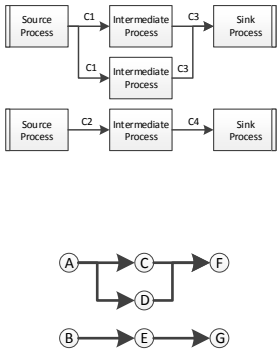

Figure 9: Abstract energy system in (a) USES (b) graph representation, fully disaggregated.

(a) (b)

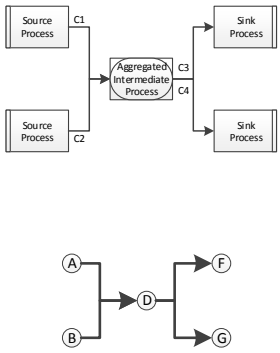

Figure 10: Abstract energy system in (a) USES (b) graph representation, fully aggregated.

(a) (b)

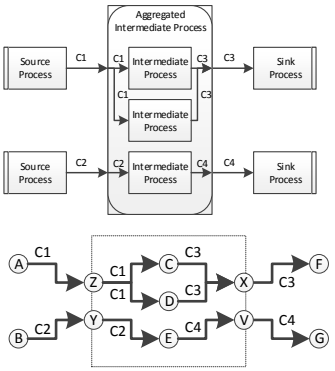

Figure 11: Abstract energy system in (a) USES (b) graph representation, aggregated and disaggregated.

The idea behind hierarchical structures is the insertion of additional vertices at those points in the graph where a commodity is connected to an aggregated process. These vertices take the function of merging (output) or splitting (input) a commodity and do not differ from other vertices in any manner. Each of the two lists a^{ctIn} and a^{ctOut} has only one entry that is the *idCommodityType* of the commodity.

3.2.2 Meta model

A model is, generally speaking, the outcome of a process in which one or more persons construct a representation of an original on the basis of his or their cognition for purposes of a subject using an object language, e.g. a natural or formal language. A *meta model* defines the object types which may be used for the creation of a model, their attributes, their semantic as well as the rules of their relationships among each other using a meta language.

The concepts of USES and their syntactical relationships among each other have to follow certain rules which have partly already been delineated in the description of each of the concepts in Section 2. In this section a meta model of USES using the semi-formal diagrammatic UML class diagram language is created and presented. The class diagram language, an excerpt of the UML, is applied because it is a widely-used and regarding its syntax fully formally specified language.

The three main classes *USES model*, *process*, and *commodity* as well as their attributes, already described in Section 3.1, and relationships among each other are shown in the upper half of Figure 12. The inheritance structure of processes as well as the associations among the specialized sub-classes is shown in the bottom half of the figure. Relationships are again denoted in the multiplicity-notation of the class diagram language of the UML.

Attributes and relationships are only illustrated once on the most generalized level and not repeated on lower specialized levels according to the general concept of inheritance [Se12]. Besides inheritance, compositions and associations are also used among the depicted classes and illustrated in Figure 12. Any association is assumed to be bi-directional. Classes whose names are written in italic are abstract classes according to the UML standard [OMG]. They cannot be instantiated and, as already stated in Section 2, do not have any graphical representation. They are abstract due to their inheritance scheme described in the previous section. They also indicate that any specialization to inherited classes is disjoint and total while being not abstract a disjoint and partial specialization is assumed.

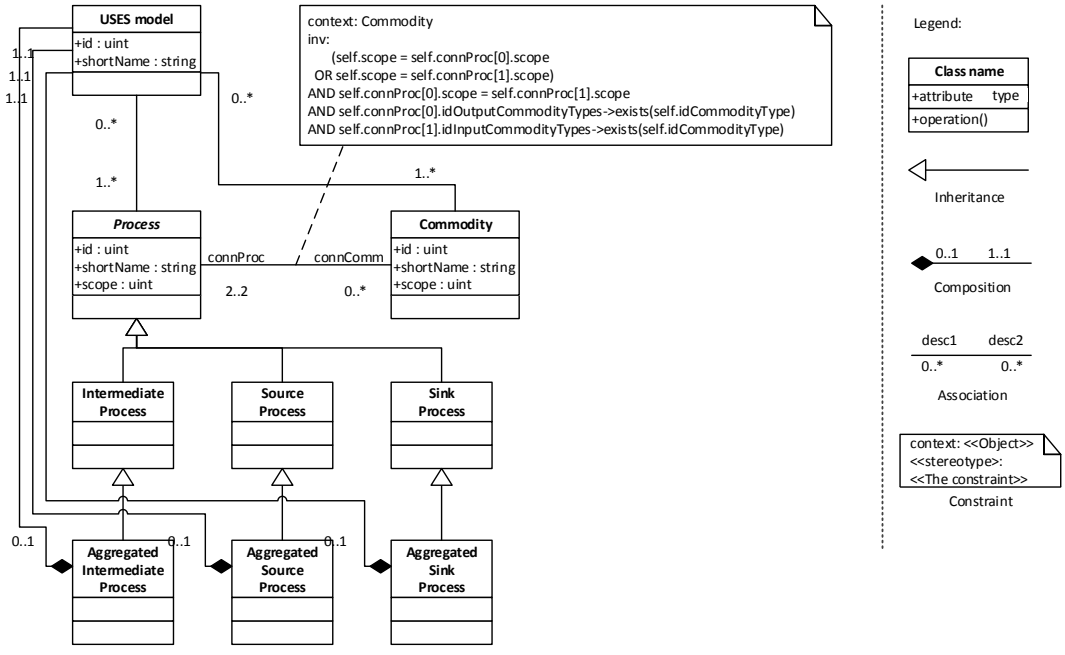

Figure 12: Meta model of USES in the class diagram language of the UML.

Since some of the syntactical rules of USES cannot be depicted using explicit diagrammatic language constructs of the UML class diagram language, constraints are required to annotate these additional rules at the respective places in the meta model. Both, the informal natural language as well as the formal *Object Constraint Language* (OCL) are allowed candidate languages for constraint specifications within class diagrams. Here, constraints are expressed in OCL to avoid ambiguities in interpretation [OCL].

The class diagram language permits constraints in form of pre- and post-conditions as well as invariants. The invariant stereotype is used to express that the constraints are true for the attached object during its complete lifetime. The only constraint applicable for USES is related to the rule that 1 commodity connects 2 processes. In this connection both processes have to have the same scope. Both, the source and the sink process of this connection have to include the same *idCommodityType* as the commodity in its *idOutputCommodityTypes* / *idInputCommodityTypes* attribute. Here, the UML implicit concept of surrogate keys to uniquely identify every object is used. This surrogate key is explicitly saved in the attribute called *id*.

4 Application

USES may not only be used for modeling small energy systems following the traditional centralized-producer-distributed-consumer architecture. Instead it is of most interest when smart grids with intermittent and decentralized energy sources and a multidisciplinary SoS approach are being investigated. In Figure 13 an example of a smart grid is shown separating the whole energy system in multiple logical and physical micro-grids. It can be seen that multiple physical micro-grids (PMG1 to PMG5) are connected with each other over physical and virtual energy connections featuring virtual power stations (VPS1 to VPS9). Two micro-grids (PMG4 and PMG5) are shown in more detail forming a hierarchy within the schematic illustration.

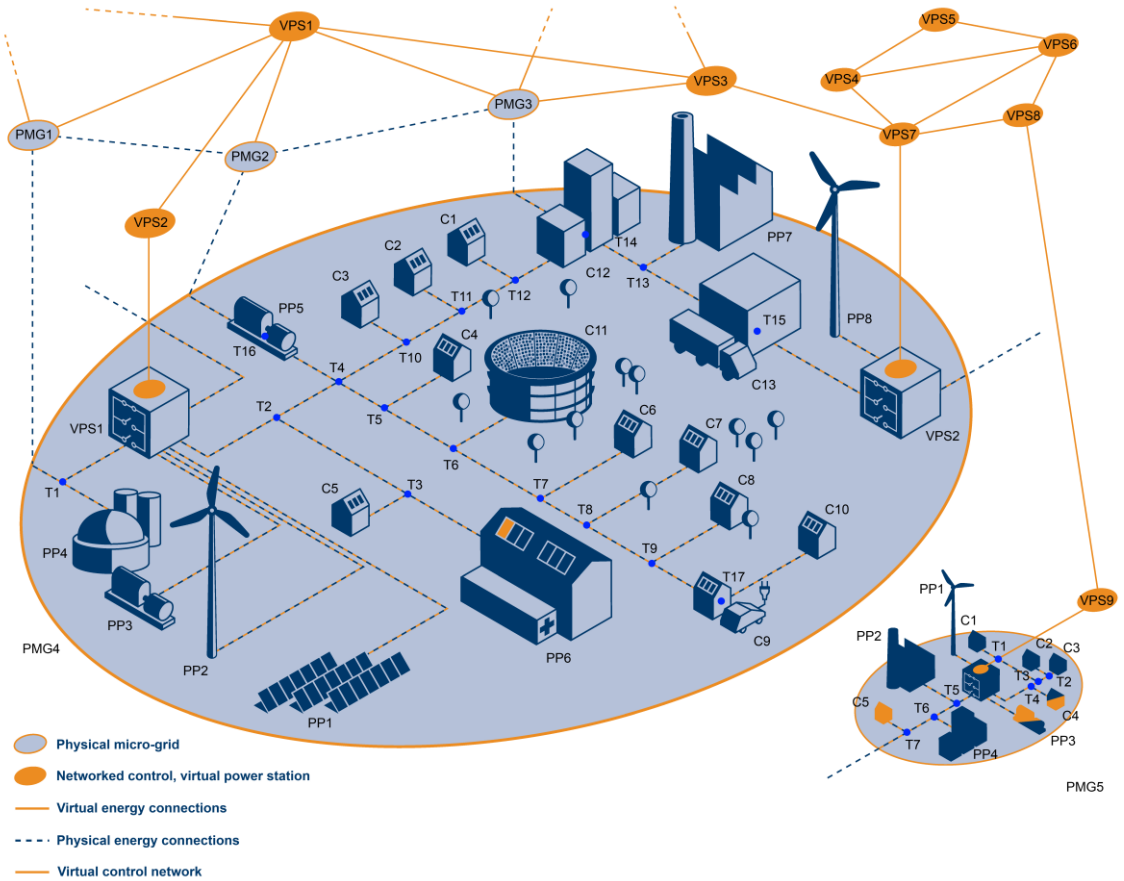

Figure 13: Schematic cut-out of a micro-grid of a distributed energy supply architecture. Source: Based on [GeBr12].

The real model of the smart grid illustrated in Figure 13 taking advantage of the graphical USES representation is shown in Figure 14. Represented are the five physical micro-grids (PMG1 to PMG5) as aggregated intermediate processes where each one could contain an own smart grid on a lower hierarchical level. To give a better overview the USES feature of hierarchical modeling was used for PMG4 and PMG5. Their internal topology will be explained in the following paragraph. PMG1 to PMG5 are connected with each other and with virtual power stations (VPS1 to VPS9) over commodities (physical and virtual energy connections). In favor of clarity the alternative bi-directional representation of commodities is being used.

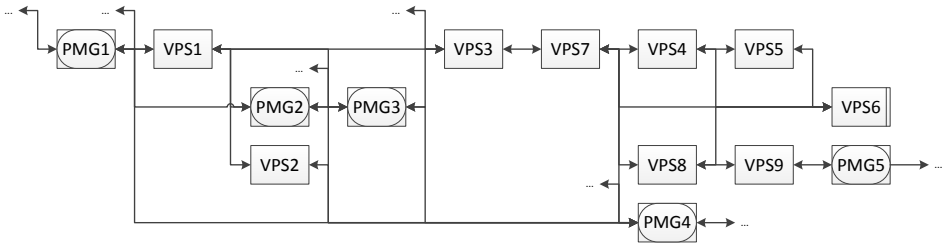

Figure 14: Model of a micro-grid taking advantage of the USES representation.

Since the physical micro-grids PMG4 and PMG5 are illustrated in more detail in Figure 13 and only the aggregated counterparts were used in Figure 14 they are also fully modeled in USES in Figure 15 and 16. In here all of the power plants (PP1 to PP8 for PMG4 and PP1 to PP4 for PMG5), virtual power stations (VPS1 to VPS2 for PMG4 and VPS1 for PMG5), transformers (T1 to T17 for PMG4 and T1 to T7 for PMG5), and consumers (C1 to C13 for PMG4 and C1 to C5 for PMG5) are modeled and connected via physical energy connections. The connections are indeed uniquely labeled but for reasons of overview enhancement not displayed in either figure.

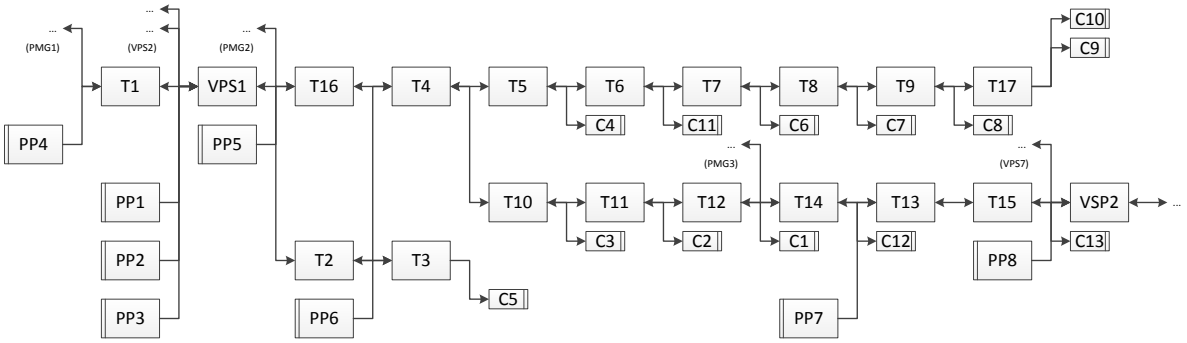

Figure 15: Sub-model of a micro-grid (PMG4) taking advantage of the USES representation.

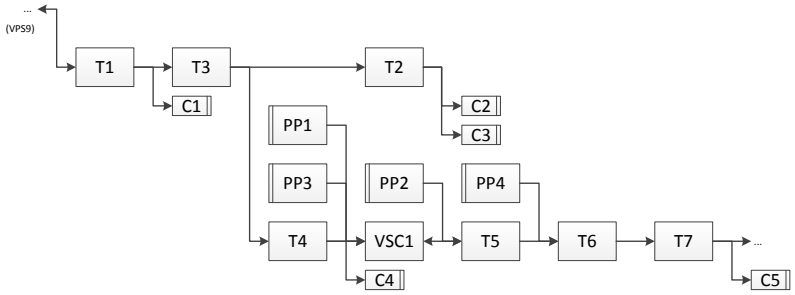

Figure 16: Sub-model of a micro-grid (PMG5) taking advantage of the USES representation.

5 Conclusions and Outlook

Since simulating complex energy systems require high computational cost efficient simulation models are needed. These models have to be based on languages whose syntax is free of shortcomings. Not only to let the user create consistent models of their energy system but also to allow software implementations to verify the topologic consistency of the model correctly. With the *Universal Scheme for modeling Energy Systems* (USES) a new language for modeling energy systems void any hitherto existing shortcomings was proposed in this paper. It provides a unitary modeling scheme and is a valuable methodology to create models of real and artificial energy systems. Because being minimal and clearly designed a conceptually complete, consistent, and precise formal syntax specification of USES is possible. In this paper the possibility was taken to show the straightforwardness of defining the syntax of USES informally and formally using graph theory and the UML class diagram language.

The graph-theoretic approach considers USES as a network of vertices and edges. This is advantageous because all concepts and algorithms that apply for a graph like e.g. adding, removing, grouping, or routing can also be applied on USES. The formal specification based on graph theory can therefore be used to implement functional behavior within USES although the functional model was not explicitly explained here. By creating a meta model the data model on which USES is based on is formally expressed. Taking advantage of the model-driven design approach, described in [La11], which is gaining popularity in software engineering, the verification of the models themselves against its specification is essential. This increases both, confidence in and quality of the model and its target application. In [CCR08 and KuGo12] a verification approach for UML class diagrams using OCL constraints is presented that can be applied on USES. Because the UML class diagram language was used the meta model can also be easily translated into code [PSB11]. By combining both approaches a high-quality information system for energy system models based on USES can be developed.

Because of its advantages USES is an ideal candidate becoming the standard for modeling energy systems and exchanging these models along with their respective data. This could pave the way for a software system implementing USES with which the topologic consistency of the created models can be verified correctly. Such a system is currently being developed. It is an information management and data storage system for multiple energy system simulation and optimization models and therefore it provides the quantitative data out of which the data models to each real model is formed. Both, topologic and additional quantitative data of energy system models are being stored in its database which scheme relates to the power and the features USES as a modeling language and the information system offers. The former discussed ways of drawing USES diagrams enriched with additional information to become a Sankey-, GIS-, or a Sankey-GIS-USES diagram will also be implemented.

List of References

[An12] Andy, K.: Data Visualization. A successful design process. Packt Publishing, 2012.

[BHZ04] Bahn, O.; Haurie, A.; Zachary, D.S.: Mathematical Modeling and Simulation Methods in Energy Systems. In: Mathematical Models, from Encyclopedia of Life Support Systems (EOLSS). Oxford, UK. 2004.

[BSS98] Blesl, M.; Schweiker, A.; Schlenzig, C.: Erweiterung der Analysemöglichkeiten von NetWork – Der Netzwerkeditor. In: Technical Report, Vol. 51, Institute for Energy Economics and the Rational Use of Energy at University Stuttgart, Germany, 1998.

[CCR08] Cabot, J.; Clarisó, R.; Riera, D.: Verification of UML/OCL Class Diagrams using Constraint Programming. In: Proceedings of IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW), pp. 73-80, 2008.

[De09] DeMers, M.: Fundamentals of Geographical Information Systems. Wiley, 2009.

[Fr05] Franzén, T.: Godel's Theorem. An Incomplete Guide to Its use and Abuse. Peters, 2005.

[GeBr12] Geisberger, E.; Broy, M.: agendaCPS. Integrierte Forschungsagenda Cyber-Physical Systems. Acatech Study, 2012.

[Ha95] Harary, F.: Graph Theory. Perseus, Cambridge, MA, USA, 1995.

[HaRu00] Harel, D.; Rumpe, B.: Modeling Languages - Syntax, Semantics and All That Stuff. Part I - The Basic Stuff. In: Technical Report, Jerusalem, Israel, 2000

[Hi07] Hipel, K.; Jamshidi, M.; Tien, J.; White III, C.: The Future of Systems, Man, and Cybernetics. Application Domains and Research Methods. In: IEEE Transactions on Systems, Man, and Cybernetics – Part C. Applications and Reviews, Vol. 37, No. 5, 2007.

[KuGo12] Kuhlmann, M.; Gogolla, M.: From UML and OCL to Relational Logic and Back. In: Proceedings of the 15th International Conference of Model Driven Engineering Languages and Systems (MODELS), pp. 415-431, 2012.

[La11] Lahman, H.: Model-Based Development - Applications. Pearson, 2011.

[Ne10] Newman, M.; Networks - An Introduction. Oxford University Press. Oxford, NY, USA. 2010.

[OCL] OMG Object Constraint Language, Version 2.3.1. Last access on 19.08.2013. http://www.omg.org/spec/OCL/

[OMG] OMG Unified Modeling Language, Version 2.5 FTF Beta 1. Last access on 19.08.2013. http://www.omg.org/spec/UML/2.5/Beta1/PDF/

[PSB11] Parada, A.; Siegert, E.; Brisolara, L.: Generating Java code from UML Class and Sequence Diagrams. In: Proceedings of the 2011 Brazilian Symposium on Computing System Engineering (SBESC), pp. 99-101, 2011.

[Se12] Sebesta, R.: Concepts of Programming Languages. Addison-Wesley, 2012.

[YaIs02] Yasuda, K.; Ishii, T.: Decentralized Autonomous Control of Super Distributed Energy Systems. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 6, 2002.