Peer-reviewed version available at Energy Strategy Reviews 2018, 22, 16-25; doi:10.1016/j.esr.2018.07.001

The Open Energy Modelling Framework (oemof) - A new approach to facilitate open science in energy system modelling☆

S. Hilpert[a,b,*], C. Kaldemeyer[a,b,c], U. Krien[d], S. Günther[a,b], C. Wingenbach[a,b], G. Plessmann[d]

[a] *Center for Sustainable Energy Systems (ZNES), Flensburg*
[b] *Department of Energy and Environmental Management, Europa-Universität Flensburg, Auf dem Campus 1, 24943 Flensburg, Germany*
[c] *Department of Energy and Biotechnology, University of Applied Sciences Flensburg, Kanzleistraße 91-93, 24943 Flensburg, Germany*
[d] *Reiner Lemoine Institut gGmbH, Rudower Chaussee 12, 12489 Berlin, Germany*

Abstract

Energy system models have become indispensable to shape future energy systems by providing insights into different trajectories. However, sustainable systems with high shares of renewable energy are characterized by growing cross-sectoral interdependencies and decentralized structures. To capture important properties of increasingly complex energy systems, sophisticated and flexible modelling tools are needed. At the same time open science becomes increasingly important in energy system modelling. This paper presents the Open Energy Modelling Framework (oemof) as a novel approach in energy system modelling, representation and analysis. The framework forms a toolbox to construct comprehensive energy system models and has been published open source under a free license. With a collaborative development based on open processes the framework seeks for a maximum level of participation and transparency to facilitate open science principles in energy system modelling. Based on a generic graph based description of energy systems it is well suited to flexibly model complex cross-sectoral systems and incorporate various modelling approaches. This

☆This document is a collaborative effort.

*Corresponding author, Phone: +49 (0) 461 805 3067
Email address: simon.hilpert@uni-flensburg.de (S. Hilpert)

makes the framework a multi-purpose modelling environment for modelling and analyzing different systems ranging from an urban to a transnational scale.

Keywords: decision support, energy system modelling, optimization, collaborative development, open science

1. Introduction

The global transition process towards more sustainable and low-carbon energy systems requires the development of alternative future trajectories for a profound scientific discussion. Based on this, decision processes on different levels e.g. in transnational policy making or local energy planning can be supported. However, future energy systems imply a rising complexity in technical, economical and socioeconomic dimension due to increasingly cross-sectoral and decentralized structures [1]. Insights into such complex systems can be gained by applying computer based modelling approaches which create a quantitative basis for the abovementioned discussion and decision processes [2].

Depending on the specific investigation and research question, a variety of model types can be applied. Among others, such model types include power flow models for electricity transmission network operation and planning, economic dispatch models for general capacity planning and unit commitment models for power plant utilization [3, 4, 5, 6]. Applications range from large scale transnational investigations using purely economical top-down equilibrium models to detailed technical local infrastructure planning using bottom-up models based on technology-specific data. Moreover, many models can be adapted to integrate different sectors such as electricity, heat and mobility to investigate cross-sectoral interdependencies.

Energy system models and derived results have often been heavily discussed among different stakeholders and being criticized for not opening their internal logic and underlying assumptions [7, 8, 9]. As a result, during the last decade more scientists have opened their models and data [10, 11]. This process goes along with a general trend to open science in many other research fields. The

rationale for open science includes an improved efficiency, scrutiny and reproducibility of results, re-usability of scientific work and an increased transparency of all scientific processes [12]. As the European Commission has recently started to push open science in their research programs [13], the subject of openness has finally moved into the public spotlight.

This paper presents the Open Energy Modelling Framework (oemof) as a novel approach to foster open science in the field of energy modelling and analysis. At first, the idea of a single energy modelling framework is differentiated from other approaches to delineate the scientific contribution in Section 2. Secondly, the underlying concept with its mathematical representation as well as the framework architecture and its implementation are outlined in Section 3. Building on this, the general process of application development is described in Section 4 along with a selection of existing applications. Finally, the general approach and its scientific contribution are summarized in Section 5.

2. Scientific contribution

To delineate our approach, first a brief overview on relevant energy system modelling software is provided. Subsequently, the presented framework is compared to similar existing software and unique features are outlined. For extensive reviews on this topic it is referred to Hall and Buckley [14], Connolly et al. [15] and Pfenninger et al. [1].

2.1. Overview of modelling landscape

In the following, it is distinguished between the three terms *model*, *model generator* and *framework*. Models are concrete representations of real world systems (e.g. with a specific regional focus and temporal resolution). Such a representation may consist of multiple hard- or soft-linked sub-models to answer clearly defined research questions. Models can be build from model generators that allow to build models with a certain analytical and mathematical approach (e.g. by the use of predefined set of equations, represented technologies). Fi-

nally, a framework can be understood as a structured toolbox including sub-frameworks and model generators as well as specific models (e.g. wind feed-in models). In addition this kind of a collection has other requirements in terms of structures and processes that guide the development process.

With respect to open science principles, a rough division into a line of closed (*1st generation*) and open (*2nd generation*) models for energy system analysis can be derived.

The *1st generation* models and model generators have a long tradition and are predominant in the academic energy system modelling landscape. Among the most widely used proprietary model generators are TIMES/MARKAL [16]. Models of this family have been used to answer research questions in the field of energy planning which is indicated by the high number of references in academic literature [14]. Similarly, MESSAGE is a prominent model generator that has been used for the IIASA global energy scenarios [17]. Besides this, the Energy-PLAN simulation model has been applied in various research projects to analyse sector integrated electricity, heat and transport systems [18].

The Balmorel model [19] can be seen as one of the first *2nd generation* energy system models. It is has been designed for power and heat dispatch modelling with optional investment within the Baltic region and is written in GAMS. Another early project is the model generator OSeMOSYS [20] which is mainly used for long-term integrated assessment and energy planning. This project aims to facilitate modelling and education due to a free software approach and a fast learning curve of the software. Since then, various other projects with different purposes have been developed (e.g. urbs [21], PyPSA [22], calliope [23]). Their focus covers the full range from power flow simulation to long-term investment models. A list of open source models can be found on the website of the Openmod-Initiative [10]. While some of these projects are models for a specific region, others can be classified as model generators.

2.2. Comparison to other software

Since the landscape of modelling software is extensive, the framework is related to similar existing tools. For this, major categories with single characteristics are introduced. These encompass the general suitability for *Open Science*, the technical *Concept* and overall modelling *Functionality*.

A requirement to foster *Open Science* is the free availability of the software itself. Free available software is software that is available without additional cost. The usage of fee-based software creates entry barriers in terms of reproducibility since the experiment can only be repeated if a respective license is procured. Moreover, an open license allows to spread, understand and change the source code and thus enhances transparency since model assumptions and internal logic can be understood, changed and evaluated in terms of their influence on the results. Open licenses include all types of open source licenses. The issue of re-usability can also be addressed when software is published under an open license since other developers can re-use any part of the software. Finally, collaborative software development allows for continuous improvements, enables an easier detection of bugs and makes it possible to discuss new features in a transparent manner. Collaborative development in this context refers to joint work ties on the software's code basis without mandatory institutional ties. This includes a common road map, discussion of new features and changes and in general a high level of communication among the developers. A central characteristic of this definition is the transparency of all associated processes.

The category *Concept* is defined by technical and structural characteristics. Implementing the software in a high-level language lowers barriers of usage and contributing. High-level programming languages are characterized by a strong abstraction from computer's hardware, are easier to use and understand, may include elements of natural language and make software development simpler. Available libraries of a language allow to integrate various tasks in the modelling tool-chain. Moreover, different interfaces to other languages can be used to extend capacities. A generic data model enables a separation between the mere topological description and subsequent calculation e.g. within an optimization.

Generic data models are data structures rather designed to specifically suit the representation of data of a particular problem than being capable to store data of multiple different problems. In terms of energy system modelling, the graph-based representation of energy systems is one example that for example allows to represent electricity systems as well as heating systems. Providing the option to define the level of accuracy flexibly is an added value to energy system modelling toolboxes. This, for example, allows for user-defined precision in representation of dimension of time or in modeling an energy system components by extending the libraries scope by user-defined components. Another aspect of the concept is designing it for multiple purposes. This extends the core functionality by other useful tools. For example, an energy system modelling toolbox that includes tools for generating feed-in profiles of renewable energy sources.

Provided *Functionality* in this case is defined by concrete modelling capabilities for problem classes such as economic dispatch, investment planning, also across multiple periods (multi period investment planning), power flow calculation and unit commitment. Furthermore, the general capability to model sector coupling problems is a prerequisite to model multi-sectoral energy systems that include different sectors such as electricity, heat and transport.

To compare the framework to other tools, Table 1 lists a selection of popular *1st* and *2nd* generation of modelling frameworks and model generators. Though oemof shares certain characteristics with existing software, the collaborative development, the generic data model and multi model toolbox (framework) differentiates oemof's from existing approaches.

2.3. Unique framework features

A core feature of the framework is its *collaborative development* with the goal of *community building*. Many existing tools are not developed by one institution only. For example researchers are encouraged to develop and improve the source code of MARKAL and other tools of the Energy Technology Systems Analysis Program (ETSAP)[27]. However, the review processes and decisions are not transparent and valuable information maybe lost in case of rejection of input.

	Open Science			Concept				Functionality						
	Free of charge	Open license	Collaborative development	High-level language	Generic data model	Flexible level of accuracy	Multi purpose toolbox	Economic dispatch	Multi period investment planning	Investment planning	Power flow	Unit commitment	Sector coupling	Source
WASP IV	✓							✓	✓	✓			✓	[24]
EnergyPlan v12	✓							✓		✓			✓	[18]
MARKAL/TIMES								✓	✓	✓			✓	[16]
MESSAGE-III								✓	✓	✓			✓	[17]
oemof v0.2	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓	✓	[25]
urbs v0.7	✓	✓		✓				✓		✓		✓	✓	[21]
calliope v0.5.3	✓	✓		✓				✓		✓		✓	✓	[26]
PyPSA v0.12	✓	✓		✓		✓		✓		✓	✓	✓	✓	[22]
OSeMOSYS	✓	✓						✓	✓	✓			✓	[20]

Table 1: A comparison of features of selected software tools that are similar to oemof.

In contrast, oemof tries to have an open process about future improvements. To align with open science principles the idea is to enable full transparency of the development process and not only the final source code. For that reason, the project follows a strict free software development. In addition, processes are designed for community building, collaborative and transparent source code development.

Another unique feature is the *generic data model* which has emerged from the collaboration of various researchers with different scopes and backgrounds. This has led to the development of a framework with a common and generic basis (Section 3.1) consisting of layer structured set of tools and sub-frameworks). A

generic graph-based basis allows to differentiate between the topology of an energy system and its calculation based on a specific mathematical approach. In this sense, oemof may be interpreted as an approach to a domain specific language [28] on how to represent arbitrary energy systems as a graph. As a consequence, an energy system in oemof can represent an economic system on a high abstraction level as well as detailed a single power plant. This demonstrates the flexibility in terms of different levels of accuracy.

Generally, the framework serves as a *multi purpose toolbox* for energy system modelling and has been created to integrate a growing set of toolboxes in future. Open source model generators like *calliope* presented by Pfenninger and Keirstead [23] and the toolbox *OseMOSYS* presented by Howells et al. [20] are designed to build specific models of one model family or type by the use of predefined sets of equations (e.g. bottom up linear optimization based models). Furthermore, some of the existing projects like *PyPSA* [22] include several model generators for different purposes that may be combined. In contrast to other tools, oemof encompasses valuable model generator approaches to generate specific economic dispatch, investment and unit commitment models. Going one step beyond, it provides a structured set of tools to facilitate the modelling process. At the current state, this set includes an optimization library (model generator) as well as tools to simulate feed-in from renewable energy sources or local heat demand for a specific region.

Summarizing, the underlying concept, the software architecture, the free software philosophy and in particular the framing processes (e.g. open meetings, open code review, open web-conferences, open platforms and open pull-requests) enable collaborative development and participation. These combined features distinguish oemof from existing projects and constitute a basis for open science in the field of energy system modelling. Its academic value lies exactly in this difference in terms of open science. In the following the specific characteristics are described in detail.

3. Concept, architecture and implementation

For a more detailed description of the framework, its underlying concept, architecture and specific implementation are outlined. As a first step, a general mathematical representation of energy systems is proposed which serves as a base for higher level software architecture presented in the second place. Finally, the specific implementation in a higher level programming level is described and justified.

3.1. Underlying concept

The main feature of the framework is the separation of an energy system's topological description, i.e. is representation and its calculation. The representation may serve as a foundation to run graph-based algorithms (i.e. is the graph connected?) or to perform exploratory analyses. Subsequently parameters of the system (or sub-systems) can be computed based on concrete modelling approaches. Due to this property, oemof can incorporate other models and model generators with varying modelling approaches and different programming languages.

To achieve this, a generic concept which constitutes the foundation of all oemof libraries has been developed. In this concept an energy system is represented as a network consisting of nodes and flows connecting these. Nodes N are subdivided into buses B and components C. When representing an energy system, an additional constraint that buses are solely connected to components and vice versa is imposed. Components are meant to represent actual producers, consumers or processes of the energy system while buses are meant to represent how these components are tied together. Flows are used to represent the inputs and outputs of a component.

An energy system that is represented in such a way can be mathematically described using concepts from graph theory by looking at it as a bipartite graph G. The mathematical formulation of this graph in its general form is given in Equation 1. A more detailed description of this concept with its theoretical

foundation has been published by Wingenbach et al. [29]. The nodes of the graph are partitioned into buses B and components C while flows act as the directed edges E of the graph.

$$\begin{aligned} G &:= (N, E) \\ N &:= \{B, C\} \\ E &\subseteq B \times C \cup C \times B \\ C^+ &\subseteq C \\ C^- &\subseteq C \\ T &\subseteq C \end{aligned} \tag{1}$$

Moreover the components can be subdivided into sources C^+, sinks C^- and transformers T:

1. *Transformers* have inflows and outflows. For example a gas turbine consumes gas from a gas bus and feeds electrical energy into an electricity bus. The relation between inflow and outflow can be specified in the form of parameters, for example by specifying the transfer function or an efficiency factor.
2. *Sinks* only have inflows but no outflows. Sinks can represent consumers of which households would be an example.
3. *Sources* have outflows but no inflows. For example, wind energy or photovoltaic plants but also commodities can be modelled as sources.

A similar, purely mathematical formulation of multi-commodity network flow optimization models for dynamic energy management has been illustrated by Zeng and Manfren [30]. Furthermore, related structures of energy systems can also be found in different energy models [20, 31, 16, 32]. These publications demonstrate that using a graph is an intuitive way of representing an energy system. The main difference of our approach compared to existing ones is the identification (and its object oriented implementation) of a specific graph structure that may be used as a representation for all types of energy systems. Every

calculation based on a specific model will be derived from this representation. A graphical representation of how to describe an arbitrary energy system using this network structure is shown schematically in Figure 1.

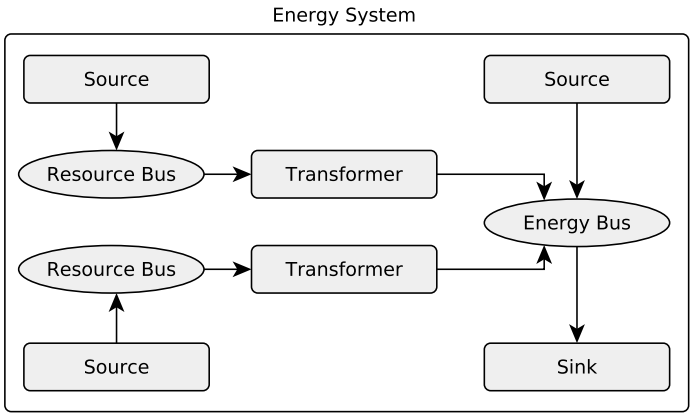

Figure 1: Schematic illustration of an energy system represented as an oemof network.

Based on the described concept, oemof provides basic components which can be used directly while also facilitating the development of more specific components built upon the basic ones. Being an openly available, uniform, sector-independent framework that provides a consistent object-oriented implementation of the generic network structure makes oemof a valuable contribution to the energy modelling and especially the energy framework family.

3.2. Project architecture

For the broad field of energy system modelling it is neither feasible nor useful to cater for it in a single library. Nevertheless, the oemof project tries to accommodate energy system modellers with a large set of functionalities they need. Hence, the project and its development process follow an architecture that groups the content of the framework into functional and organizational units. This architecture consists of the four layers depicted in Figure 2. These four layers are used to categorize the libraries associated with the oemof project according to their dependencies and commonalities.

The framework itself and its underlying concept are implemented using an object-oriented approach in the high-level programming language Python and they are published under the GNU GPL3 license. As Python libraries are called packages, a main component of the framework is the oemof package. This package covers the first layer completely and the second layer partially. The layers beyond the first differentiate how closely libraries are associated with the oemof package and its developers in terms of organizational ties as well as technical dependencies.

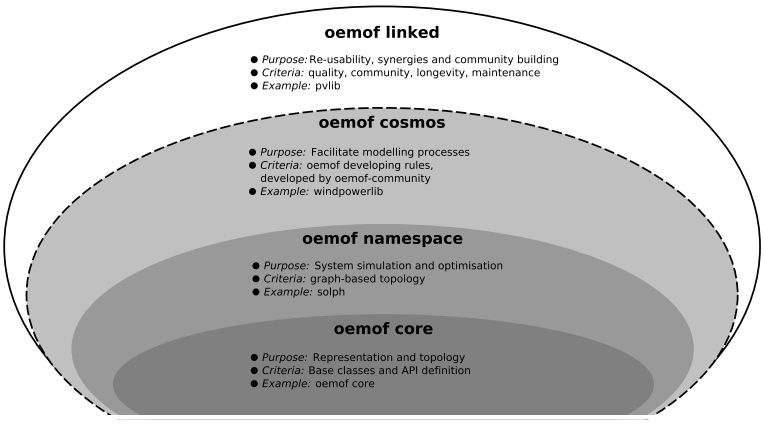

Figure 2: Layer structure of the oemof project architecture.

1. At the *core* layer a generic graph structure is implemented via core classes. These classes are used to instantiate the objects comprising an energy system graph and define how an energy system is described. In addition the basic application programming interface (API) is defined through which the the core objects and their properties representing the graph are accessed. This also means that the API defines the format of input and output data for core objects. The graph structure, in fact the entire *core* layer, is kept free from energy system specific logic in order to accommodate a broad spectrum of modelling approaches. Additionally, it allows to decouple the energy system's representation from how it is modelled. The intended use of the core objects in layers above the *core* layer is communicated via carefully chosen naming

and explicit documentation.

2. The *namespace* layer contains associated libraries that share the same basic system formulations, i.e. libraries modelling energy systems as graphs described in terms of objects from the *core* layer. These libraries publish their functionalities under the oemof namespace and may be part of the oemof package or they may be standalone packages. They depend on the basic API by either directly using *core* classes or adding functionality on top of them via inheritance. That way different modelling approaches can be used on energy systems described in a uniform way, namely as energy system graphs consisting of instances of *core* classes or of classes inherited from them. Possible modelling approaches can model energy systems with respect to cost, power-flow or any other kind of simulation or optimization goal. Currently the *oemof.solph* can be used to generate linear (mixed-integer) optimization problems from an energy system representation based on core objects. However, the graph structure is capable of accommodating other concepts such evolutionary optimization approaches or agent based modelling.
3. The *oemof* layer contains libraries from the field of energy system modelling that are associated with oemof in an organisational way without sharing the basic API. These libraries, while still part of the oemof project, are not developed as part of the same package and may thus be used, reviewed and developed by third party modellers and experts as well. However, as they are developed as part of the framework, they follow the common development rules (Section 3.4). As most modellers are not primarily programmers, sharing the same development, structure and documentation rules can help to learn the ropes of libraries. One example of such a library is the windpowerlib [33], a library generating feed-in time series of wind energy turbines from meteorological data.
4. As Pfenninger [11] mentioned that open source does not necessarily lead to cooperation, the fourth layer contains libraries that could be part of the layers below but already exist as a community project and have been proven to be capable of being integrated in oemof models due to their general struc-

ture. These libraries are written in Python but do not necessarily share the same rules. However, in order to be considered associated, they should meet general standards for quality, code development, longevity, maintenance and community structure. One example of such a library is the pvlib [34], which is a library developed independently from oemof and is going to be integrated into the framework via adapters in feedinlib. The process of developing these adapters already lead to code contributions towards pvlib, instead of the creation of a parallel, competing solution.

3.3. Implementation

The network concept has been implemented at the *core* layer in form of a class hierarchy which is sketched in Figure 3. The root elements of this class hierarchy are *Node*, *Edge* and *EnergySystem*. *Node* is the abstract base class for *Bus* and *Component*, which are used to represent nodes in the bipartite graph representing the energy system. Furthermore, components are subdivided into *Source*, *Sink* and *Transformer* classes depending on how they are connected to *Bus* objects. Objects of the class *Edge* represent the directed edge between two nodes, i.e. the connection between a *Bus* and a *Component* object. Moreover, the class *EnergySystem* serves as a container for nodes and may hold additional information about the energy system.

All basic energy system components such as energy demands, (renewable) energy sources and transformers between different energy buses can be modelled by means of these basic classes. Additional components that introduce new features can be added via inheritance. If sub-classing is not suitable, new classes can be created and used together with the core classes. As an example, the *solph* library introduces a storage class with different individual parameters.

The initially outlined main framework feature of separating the description of the energy system from its computation is reflected by the introduction of the *Edge* class which is separate from the *Node* class hierarchy. Objects of this class hold information about the flow between two nodes such as maximum available transfer capacities of power line flows or whether the amount of a

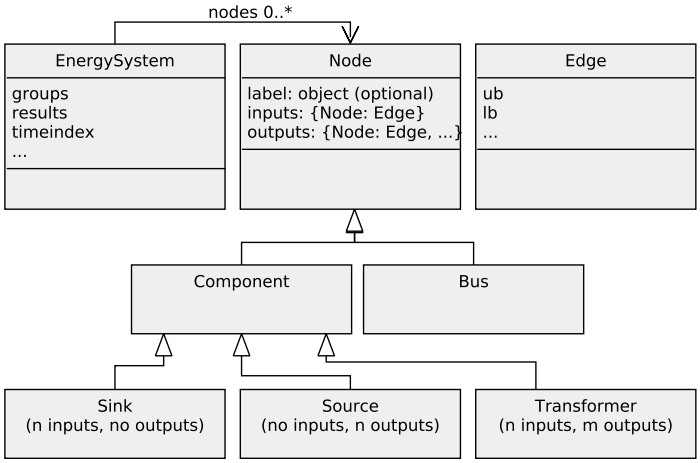

Figure 3: UML class diagram of oemof core classes.

certain flow is fixed and if so, its value. As an example for the provided generic flexibility, objects of this class are used in the *solph* library to build inter-temporal constraints for different kinds of energy system optimization problems such as combined heat and power modelling or unit commitment.

The *EnergySystem* class serves as a container for the above mentioned elements and provides the possibility to add extra information such as grouping structures or optimization parameters. Additionally, it provides interfaces to save and restore the energy systems instance and to process results. This allows for an intuitive handling of energy systems by treating them as an own entity.

Overall, an implementation within the high level programming language Python has the advantage of a rich set of external libraries usable for scientific computing. Oemof itself makes heavy use of external modules for optimization problem (pyomo [35]) and data handling (pandas [36]).

3.4. Documentation, collaboration and testing

'A critical part of any piece of software is the documentation' has already been stated by Greenhall and Christie [37]. This is of particular importance for open source projects with many users and a changing developer base. With

the objective of a profound documentation in all stages and formulation of general nomenclature, a documentation approach on four different levels similar to Howells et al. [20] is followed:

1. *Comments inside the code* are used to explain non-intuitive lines of code to new developers and interested users at the lowest level.
2. *Docstrings* located inside the source code describe the API, i.e. how to use the various classes, methods, and functions.
3. *Higher level descriptions* provide the user with additional information about the possible interactions between different libraries or application-specific usage information. These manuals are located inside the repository and are therefore shipped with the source code.
4. *Examples* provide an additional source of documentation that is particularly useful to new users.

Keeping such detailed documentation consistent and up to date across continuous releases comes at the expense of a high maintenance effort. Nevertheless, it is of special importance, as oemof provides most of the mathematics that an oemof based model is built upon. Therefore, the oemof documentation is the central point to find information on the formulas used within an oemof based model. As such, being up to date, consistent and well tracked with respect to changes over time is essential if external users want to understand the internal logic of a model, especially with respect to being able to reproduce scientific results. The upside is that documentation adhering to these principles acts as a citable source of information, reducing the amount of redundant information needed to be published and digested in order to understand a model. This in turn increases transparency and comparability when it comes to energy system models and their results.

As oemof is an open source community project, a common platform for collaboration is needed. Similar to Greenhall and Christie [37] as well as other open source energy modelling projects, oemof uses GitHub for collaboration, code hosting and bug-tracking which allows for an easy copying and forking

of the project. To lower entry barriers for new developers, hierarchies for all processes are kept as low as possible. We found that this can create a sense of belonging for collaborating developers which increases participation. GitHub is based on the version control system *git* and code can be developed in parallel on different branches. In order to ensure an effective branching strategy and release management, a well established git workflow model [38] is set as standard for all developers. Contributions to the code base are managed in form of pull requests which allow for an openly discussed review process of potential changes. Moreover, code changes are checked for conflicts before being merged back into the development branch by the respective developer in charge of the affected library.

In order to test oemof's functionality in case of changes to multiple parts of the code base, *unit tests* are applied. During the testing process, all integrated application examples are run and the created results are checked against stored historical results. Only if all examples run without errors a pull request is merged back into the development branch. This procedure ensures the functionality even if major changes to the code base are applied from one release version to another.

4. Usage: Applications

The framework is not designed to constitute a standalone executable. Instead, the oemof libraries are meant to be used in combination build energy system models. In the following we will refer to such models as oemof-*applications*.

4.1. Application development

Applications can be developed by the use of one or more framework libraries depending on the scope and purpose. Figure 4 illustrates a exemplary process of building an application. Modelling can thus range from a few plain steps in a standalone Python executable to complex procedures bundled in a new Python library based on oemof. Due to the modular concept, specific functionalities

of oemof libraries can be substituted easily depending on modelling task. This provides a high degree of freedom for developers, which is particularly relevant in scientific working environments with spatially distributed contributors and fast evolving research questions.

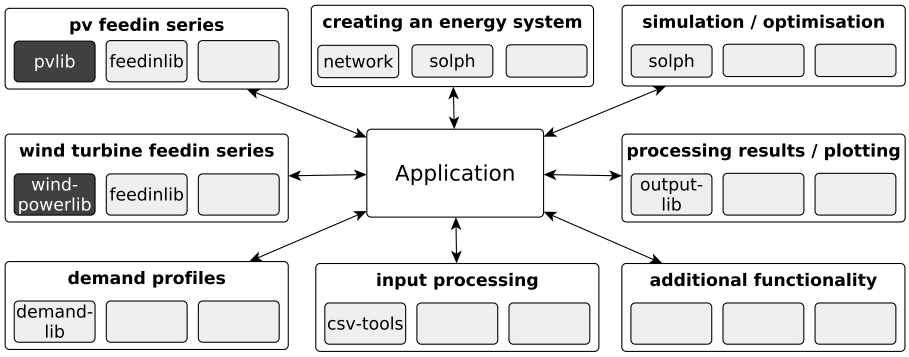

Figure 4: Building an application based on libraries of the oemof cosmos and external libraries (dark grey).

Depending on the problem, input data can be created by means of libraries such as the *feedinlib* or *demandlib* library. Moreover, a standardized result processing library (*outputlib*) provides all optimization results in convenient data structures that are ready for exports to different formats, detailed analyses and plotting. Although this feature might appear trivial, it is one major advantage over other heterogeneous optimization tool-chains that require a switching between tools e.g. GAMS for the modelling and a spreadsheet based solution for result processing.

However, regarding the modelling workflow for the oemof namespace layer, all applications have some major steps in common and throughout include all required data pre- and post-processing. First of all, an empty energy system object is created. This object acts as a container for the nodes and carries information such as the time resolution. The energy system object may also hold different scenarios representing different developments of the system. Additionally, methods to handle nodes are provided. The next step is the instantiation of

nodes and flows of the modelled energy system which are added to the existing energy system instance (population of energy system). Subsequently, the results of the energy system can be computed by simulating or optimizing the system. Finally, results can be processed with the output library of oemof. The *oemof-outputlib* makes it easy to get different views of the results and plots based on a uniform output data format.

4.2. Exemplary application workflow: System optimization

One common use case for a modelling process that utilizes different toolboxes is the optimization of energy systems. In this process the *solph* library can be used in combination with existing input and output data libraries. In a first step, feed-in data for renewable power plants and electricity demand profiles can be generated within the *feedinlib* and *demandlib* libraries. Subsequently, the data are used as exogenous parameters within the *solph* library before the optimization results are processed within *outputlib*.

The *solph* library enables to create (mixed-integer) linear models. As a common requirement, an energy system graph has to be created with *core* level objects, respective *solph* subclasses from the *namespace* layer or a mix of both. The energy system serves as a container that holds all nodes and general information such as the temporal resolution of the optimization problem. Since an oemof optimization model inherits from a model of the *pyomo* package, the full functionality of this package can be leveraged. Depending on the experience and modelling task three different ways exist to create an optimization problem based on an oemof energy system instance.

1. In the most common and easiest use case, the energy system describes graph with flows on its edges by combining basic components and buses. The optimization model for this use case is automatically created by a logic that transfers the graph (connections between buses and components and their attributes) into respective constraints e.g. commodity balance equations or inequalities for lower and upper flow bounds. When using this way of modelling, all models are derived by the object parametrisation

and no mathematical definitions like sets, variables or parameters have to be implemented.

2. In the second use case, basic energy systems can be adapted by defining additional constraints on top of the aforementioned graph logic. Since this logic is throughout consistent, entry barriers for new users are lowered . As one example an annual limit on a commodity flow can be implemented easily by a definition (in)equations applied to a set of flows.
3. In the third use case, custom components can be added to a model. This is possible by subclassing from core components or creating own components from scratch. As mentioned before, the full functionality of the *pyomo* package can be utilized up to model complex internal relations of a components with numerous constraints, specific sets and different variable domains. Such a component needs to provide input/output slots that may be connected with flows of graph.

All use cases can either be applied separately or combined within one model. The model type itself e.g. an economic dispatch, investment or unit commitment model is determined by its parametrisation. This allows for maximum flexibility as one can quickly change the model e.g. from economic dispatch to investment by exchanging single components e.g. a storage with fixed capacity (parameter) by one with variable cost-determined capacity (decision variable). In a similar way, complete sub-graphs can be exchanged quickly by connecting or disconnecting them from a main problem. One example for this flexibility would be a detailed mixed-integer model of a municipal district heating model (unit commitment) that ca quickly be connected to an integrated power market model (economic dispatch).

4.3. Existing applications

The framework has already been used to build comprehensive applications for different research projects (s. [39, 40, 41, 42, 43, 44]). In addition, oemof is also used actively in teaching by some institutions in order to gain insights

into complex energy systems. An example for such a system modelled as an oemof application is illustrated in Figure 5. In the following, selected oemof applications are described to illustrate the broad range of applications. These distinguish themself for example by considered technologies, modelled demand sectors, regional representation, the time horizon of the analysis, the modelling methodology to represent technological characteristics and perhaps a market representation.

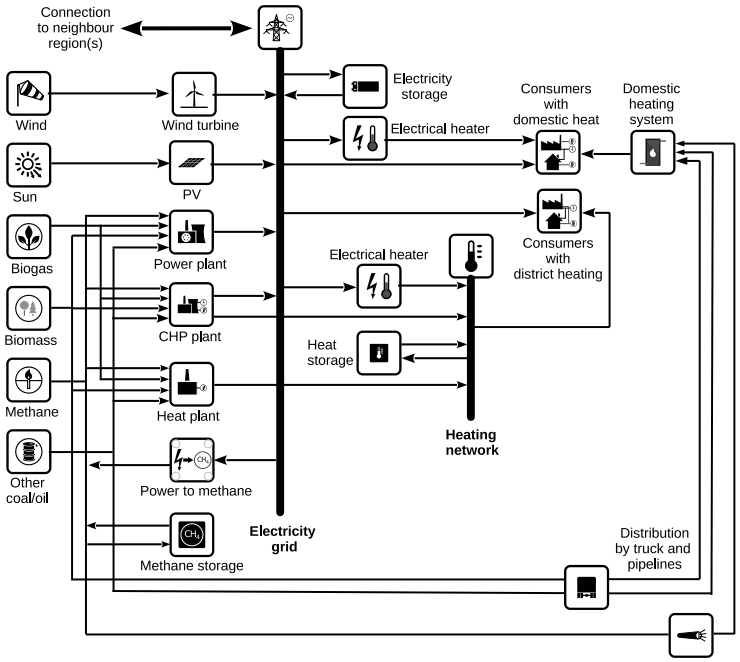

Figure 5: Representation of a complex energy system within an oemof application.

The renewable energy pathways simulation system (*renpassG!S*) [45] is a bottom-up fundamental Western European electricity market model. Future scenarios of the power plant dispatch and price formation in Germany and its interconnected neighbouring countries can be modelled based on operational and marginal costs and the assumption if an inelastic electricity demand. Based on *renpassG!S* a spin-off model is created which is adapted to the requirements of the Middle East and North Africa (MENA) region. In this application the

solph library was used with a restriction to solely linear equations. Both applications have in common to use a standardized interface to csv-files for the *solph* library that was created to simplify the usability for users with no programming experience.

The *openMod.sh* model is a flexible software tool that is strongly based on oemof's underlying concept [42]. This model has been applied in participative workshops for the development of regional climate protection scenarios. The combination of a graphical browser-based user interface combined with an open source modelling approach enhances the modeller-decision maker interface. The extension of the underlying concept to a database concept shows that this concept may not only be used for the computation of systems but also for its representation in a relational database. This shows that due to the open license and the high level language oemof applications can be setup on public servers with low effort and without legal barriers.

An example for the flexible extension of oemof on the application level is demonstrated within the Heating System Optimization Tool (*HESYSOPT*) [46]. In this application, detailed heating system components are modelled with mixed integer linear programming techniques that are based on oemof.solph functionalities. Using the underlying *pyomo* library *solph* provides an interface to add new components within the application. After a review such components can be integrated into *solph* to be available to the entire community.

As a fourth example, *reegis*^{hp} [47] models heat and power systems on a local scale. The objective is to evaluate district heating and combined heat and power technologies in energy systems with a high share of renewable resources from an environmental and economical perspective. The local system is connected in terms of electricity to a national model based on the idea of the model *renpassG!S* [45] which is extended to include the heating sector. This application is using the oemof's *windpowerlib*, *feedinlib* and *demandlib* to provide the input data for the model. Furthermore, the *solph* library is used to create a large scale linear model and a detailed mixed integer linear problem. This example demonstrates how models of different scale may be combined in one application.

These applications illustrate the flexibility of oemof and the extent of the potential user group not only with regard to the content, but also concerning the level of involvement. It is possible to build a full-scale energy system model adapted to the user's needs by just employing existing functionalities. Moreover, different models can be combined and adapted at low effort to create tools for specific purposes. Overall, this enables users to answer challenging research questions within a single framework.

5. Conclusion

The paper presents the Open Energy Modelling Framework (oemof) as a contribution to the scientific modelling community. With a collaborative and open development process it is designed for transparency and participation. Complementary to its technical features, the project constitutes a novelty in energy system modelling and aims to facilitate open science in this research field.

One central feature of oemof is the generic graph based foundation which has been implemented using an object-oriented approach in the high level programming language Python. The cross-institutional collaborative development of the framework has triggered a process towards this common and generic structure. This concept highlights the distinction between the description of an energy system with its components and subsequent computations based on combining the description with a specific mathematical approach. Moreover, it lays the foundation for a universal representation of multi-sectoral energy systems at different scales. Another important feature is its strict open source and non-proprietary philosophy. This philosophy, the underlying concept and the extensive documentation allows new developers to adapt or extend the framework easily and leverage features of other scientific Python libraries. With these properties, the project is geared towards new developers and users and thereby a continuous future development process.

The framework has been successfully applied within different research projects

across several institutions. Among others, existing oemof applications include electricity market models, detailed technical unit commitment models for district heating systems and sector coupled regional energy system models. Essentially, energy systems ranging from distributed or urban ones up to a national scale may be modelled, making the framework a multi-purpose modelling environment for strategic energy analysis and planning.

Although there exists a learning curve for new users to build an oemof based application, we think there are profound arguments to choose oemof. Firstly, the flexibility in terms of application development allows adjustments along with changing research objectives and may thus avoid lock-in effects. This seems to be particularly relevant for project based research. Secondly, the community character of the oemof-project is another important factor. The possibility for active participation, which includes decision processes, allows users to be part of a community. We argue that this can create a sense of belonging which is a strong argument for choosing oemof in addition to the technical features of the software.

Acknowledgements

The group of authors of this paper is not identical with the oemof developer group. Since oemof is a collaborative project, all developers can be found on the contributors page of each repository on GitHub. The project was initially created by the following institutions:

- Europa-Universität Flensburg (EUF)
- Flensburg University of Applied Sciences (HFL)
- Otto von Guericke University Magdeburg (OVGU)
- Reiner Lemoine Institut Berlin (RLI)

Special thanks and acknowledgement is owed to all people who have been supporting the project since its starting phase. In particular we want to thank Birgit Schachler (RLI), Caroline Möller(RLI), Martin Söthe (ZNES), Wolf-Dieter Bunke (ZNES) and Steffen Peleikis for their individual contributions throughout different stages.

References

[1] S. Pfenninger, A. Hawkes, J. Keirstead, Energy systems modeling for twenty-first century energy challenges, Renewable and Sustainable Energy Reviews 33 (2014) 74–86.

[2] S. Quoilin, H. Gonzalez, Zucker, Modelling Future EU Power Systems Under High Shares of Renewables: The Dispa-SET 2.1 open-source model, JRC TECHNICAL REPORTS, 2017. doi:dx.doi.org/10.2760/25400.

[3] D. R. Biggar, M. Reza Hesamzadeh, The Economics of Electricity Markets, John Wiley & Sons, 2014.

[4] C. Harris, Electricity Markets: Pricing, Structures and Economics, John Wiley & Sons. Second Edition, 2006.

[5] A. J. Wood, B. F. Wollenberg, G. Sheble, Power Generation, Operation and Control, John Wiley & Sons. Third Edition, 2013.

[6] D. Kirschen, G. Strbac, Fundamentals of Power System Economics, John Wiley & Sons, 2004.

[7] S. Pfenninger, J. DeCarolis, L. Hirth, S. Quoilin, I. Staffell, The importance of open data and software: Is energy research lagging behind?, Energy Policy 101 (2017) 211 – 215.

[8] S. Pfenninger, Energy scientists must show their workings, Nature 542 (2017) 393.

[9] J. F. DeCarolis, K. Hunter, S. Sreepathi, The case for repeatable analysis with energy economy optimization models, Energy Economics 34 (2012) 1845–1853.

[10] Openmod-Initiative, openmod - open energy modelling initiative, http://www.openmod-initiative.org/, 2017. (accessed 26.04.2017).

[11] S. Pfenninger, L. Hirth, I. Schlecht, E. Schmid, F. Wiese, T. Brown, C. Davis, M. Gidden, H. Heinrichs, C. Heuberger, S. Hilpert, U. Krien, C. Matke, A. Nebel, R. Morrison, B. Müller, G. Pleßmann, M. Reeg, J. C. Richstein, A. Shivakumar, I. Staffell, T. Tröndle, C. Wingenbach, Opening the black box of energy modelling: Strategies and lessons learned, Energy Strategy Reviews 19 (2018) 63 – 71.

[12] OECD, Making open science a reality, Science, Technology and Industry Policy Papers, No. 25 (2015).

[13] European Commission, Horizon 2020 - the eu framework programme for research and innovation, https://ec.europa.eu/programmes/horizon2020/, 2017. (accessed 27.04.2017).

[14] L. M. Hall, A. R. Buckley, A review of energy systems models in the UK: Prevalent usage and categorisation, Applied Energy 169 (2016) 607–628.

[15] D. Connolly, H. Lund, B. Mathiesen, M. Leahy, A review of computer tools for analysing the integration of renewable energy into various energy systems, Applied Energy 87 (2010) 1059 – 1082.

[16] R. Loulou, G. Goldstein, K. Noble, Documentation for the MARKAL Family of Models, Technical Report, Energy Technology Systems Analysis Programme, 2004.

[17] L. Schrattenholzer, The Energy Supply Model MESSAGE, IIASA Research Report, IIASA, Laxenburg, Austria, 1981. URL: `http://pure.iiasa.ac.at/1542/`.

[18] Aalborg University, Energy Plan - Advanced energy system analysis computer model (Version 12.5), http://www.energyplan.eu/, 2016. (accessed 05.05.2017).

[19] H. F. Ravn, The Balmorel Model: Theoretical Backround, http://balmorel.com/images/downloads/The-Balmorel-Model-Theoretical-Background.pdf, 2001. (accessed 04.05.2017).

[20] M. Howells, H. Rogner, N. Strachan, C. Heaps, H. Huntington, S. Kypreos, A. Hughes, S. Silveira, J. DeCarolis, M. Bazillian, A. Roehrl, OSeMOSYS: The Open Source Energy Modeling System: An introduction to its ethos, structure and development, Energy Policy 39 (2011) 5850–5870.

[21] J. Dorfner, K. Schönleber, M. Dorfner, S. Herzog, tum-ens/urbs: v0.7, 2017. URL: https://doi.org/10.5281/zenodo.242029. doi:10.5281/zenodo.242029.

[22] T. Brown, J. Hörsch, D. Schlachtberger, PyPSA: Python for Power System Analysis, Journal of Open Research Software 6 (2018).

[23] S. Pfenninger, J. Keirstead, Renewables, nuclear, or fossil fuels? Scenarios for Great Britain's power system considering costs, emissions and energy security , Applied Energy 152 (2015) 83–93.

[24] INTERNATIONAL ATOMIC ENERGY AGENCY (Ed.), Wien Automatic System Planning (WASP) Package - A Computer Code for Power Generating System Expansion PlanningVersion WASP-IV, number 16 in Computer Manual Series, INTERNATIONAL ATOMIC ENERGY AGENCY, Vienna, 2001. URL: http://www-pub.iaea.org/MTCD/Publications/PDF/CMS-16.pdf.

[25] oemof Developer Group, Open Energy Modelling Framework (oemof) - A modular open source framework to model energy supply systems. Version v0.2, 2018. URL: http://doi.org/10.5281/zenodo.1146183. doi:10.5281/zenodo.1146183.

[26] S. Pfenninger, B. Pickering, calliope-project/calliope: Release v0.5.3, 2017. URL: https://doi.org/10.5281/zenodo.846864. doi:10.5281/zenodo.846864.

[27] ETSAP, Letter of Agreement, http://iea-etsap.org/tools/TIMES-LoA.pdf, 2017. (accessed 09.11.2017).

[28] G. Booch, A. W. Brown, S. Iyengar, J. Rumbaugh, B. Selic, An MDA Manifesto, 2004. URL: http://www.bptrends.com/publicationfiles/05%2D04%20COL%20IBM%20Manifesto%20%2D%20Frankel%20%2D3%2Epdf.

[29] C. Wingenbach, S. Hilpert, S. Günther, The core concept of the Open Energy Modelling Framework (oemof), in: Environmental Informatics – Current trends and future perspectives based on 30 years of history, 2016.

[30] D. Zeng, M. Manfren, Multi-commodity network flow models for dynamic energy management – Mathematical formulation, Energy Procedia 14 (2012) 1380–1385.

[31] L. G. Fishbone, H. Abilock, MARKAl, a linear-programming model for energy system analysis: technical description of the BNL version, International Journal of Energy Research 5 (1981) 353 – 375.

[32] J. Richter, DIMENSION - A Dispatch and Investment Model for European Electricity Markets, http://www.ewi.uni-koeln.de/fileadmin/user_up,load/Publikationen/Working_Paper/EWI_WP_11-03_DIMENSION.pdf, 2011. EWI Working Papers 2011-3.

[33] S. Haas, B. Schachler, U. Krien, S. Bosch, wind-python/windpowerlib: The new restructuring release, 2017. URL: https://doi.org/10.5281/zenodo.824268. doi:10.5281/zenodo.824268.

[34] W. Holmgren, Calama-Consulting, T. Lorenzo, U. Krien, bmu, DaCoEx, mayudong, M. Mikofski, E. Miller, Heliolytics, KonstantinTr, jforbess, M. A. Anoma, pyElena21, V. Beutner, MLEEFS, J. Dollinger, C. Hansen, mangecoeur, jkfm, V. Guo, M. Campanelli, L. T, J. Gaffiot, J. Oos, G. Peronato, B. Schachler, A. Kafkes, pvlib/pvlib-python: 0.5.1, 2017. URL: https://doi.org/10.5281/zenodo.1016425. doi:10.5281/zenodo.1016425.

[35] J.-P. W. Willian Hart, Carl Laird, D. L. Woodruff, Pyomo - Optimization Modeling in Python, Springer, 2012.

[36] W. McKinney, Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O'Reilly, 2012.

[37] A. Greenhall, R. Christie, Minpower: A power systems optimization toolkit, in: 2012 IEEE Power and Energy Society General Meeting, 2012, pp. 1–6. doi:10.1109/PESGM.2012.6344667.

[38] VanDriessen, A successful git branching model, http://nvie.com/posts/a-successful-git-branching-model, 2010. (accessed 31.10.2016).

[39] C. Möller, K. Kuhnke, M. Reckzugel, H.-J. Pfisterer, S. Rosenberger, Energy storage potential in the Northern German region Osnabrück-Steinfurt, IEEE, 2016, pp. 1–7. doi:10.1109/IESC.2016.7569497.

[40] U. P. Müller, I. Cussmann, C. Wingenbach, J. Wendiggensen, AC Power Flow Simulations within an Open Data Model of a High Voltage Grid, Springer International Publishing, Cham, 2017, pp. 181–193. doi:10.1007/978-3-319-44711-7_15.

[41] M. Degel, M. Christ, J. Grünert, L. Becker, C. Wingenbach, M. Soethe, W.-D. Bunke, K. Mester, F. Wiese, VerNetzen: Sozial-ökologische und technisch-ökonomische Modellierung von Entwicklungspfaden der Energiewende. Projektabschlussbericht, IZT Berlin, Europa-Uniersität Flensburg, Deutsche Umwelthilfe e.V., 2016.

[42] C. Wingenbach, S. Hilpert, S. Günther, openMod.SH - Ein regionales Strom-Wärme-Modell für Schleswig-Holstein basierend auf open source und open data, in: 14th Symposium on Energyinnovation, Technical University Graz, 2016.

[43] C. Kaldemeyer, C. Boysen, I. Tuschy, Compressed air energy storage in the german energy system – status quo and perspectives, Energy Procedia 99 (2016) 298 – 313. 10th International Renewable Energy Storage Conference, IRES 2016, 15-17 March 2016, Düsseldorf, Germany.

[44] O. Arnhold, M. Fleck, K. Goldammer, F. Grüger, O. Hoch, B. Schachler, Transformation of the German energy and transport sector - a national analysis, in: 2. Internationale ATZ-Fachtagung: Netzintegration der Elektromobilität 2017, Berlin, 2017. (Paper accepted).

[45] ZNES, renpass-gis (Renewable ENergy PAthway Simulation System capable of working with GIS data), https://github.com/znes/renpass_gis, 2016. (accessed 31.10.2016).

[46] S. Hilpert, HESYSOPT - An open source tool to support district heating system flexibilisation., in: V. Wohlgemuth, F. Fuchs-Kittowski, J. Wittmann (Eds.), Environmental Informatics – Current trends and future perspectives based on 30 years of history, Shaker, 2016, pp. 361–366.

[47] U. Krien, reegis - An oemof application to model local heat and power systems, https://github.com/rl-institut/reegis_hp, 2016. (accessed 31.10.2017).