Wind Energ. Sci. Discuss., https://doi.org/10.5194/wes-2019-17
Manuscript under review for journal Wind Energ. Sci.
Discussion started: 2 May 2019

OWFgraph: A graph database for the offshore wind farm domain

Erik Quaeghebeur[*], Sebastian Sanchez Perez-Moreno[*], and Michiel B. Zaaijer[*]
[*]Wind Energy Section, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, the Netherlands
Correspondence: Erik Quaeghebeur (E.R.G.Quaeghebeur@tudelft.nl)

Abstract. The construction and management of an offshore wind farm involves many disciplines. It is hard for a single designer or developer to keep an overview of all the relevant concepts, models, and tools. Nevertheless, this is needed when performing integrated modeling or analysis. To help researchers keep this overview, we have created OWFgraph, a knowledge base for the offshore wind farm domain, implemented as a graph database. This paper presents the structure of this graph database—content stored in nodes and relationships between them—as a foundational ontology, which classifies the domain's concepts. This foundational ontology partitions the domain in two: a part describing physical aspects and a part describing mathematical and computational aspects. This paper also discusses a number of general difficult cases that exist when adding content to such a knowledge base. This paper furthemore discusses the potential applications of OWFgraph and illustrate its use for computation pathway discovery—the application that triggered its creation. It also contains a description of our practical experience with its design and use and our thoughts about the community use and management of this tool.

1 Introduction

An off-shore wind farm is a complex system, composed of many subsystems that interact with other subsystems and external systems. Its design involves multiple disciplines, such as installation, operation, maintenance, and decommissioning. The result is that keeping an overview of the offshore wind farm 'domain' is hard to do for a single person or a small team. However, at the current stage of development of the domain, where subsystems—such as turbines—have already seen a good number of optimized design iterations, the system interactions have become relatively more important. This prompts a more holistic, systems engineering approach to open up the possibility of further gains in, e.g., productivity, efficiency, and robustness (van Kuik et al., 2016, §7). Therefore, a tool that enables a developer, designer, or researcher in the field to keep the overview can provide great benefits.

This paper presents such a tool, OWFgraph. It is a graph database for the offshore wind farm domain. Graph databases are used as knowledge bases for various purposes in diverse domains, such as general human knowledge (Bollacker et al., 2008; Vrandečić and Krötzsch, 2014; Speer et al., 2017), biochemistry and biomedicine (Jupe et al., 2012; Franceschini et al., 2013; Himmelstein and Baranzini, 2015), and aerospace (Taymaz et al., 2013). The domain, both in its physical and mathematical aspects, is described using information-carrying nodes—concepts—connected by edges—relationships. Information can be added locally and through the graph structure be connected to the global whole. The database can be queried to return the information desired in the format required. The creation of this graph database was started in the context of projects dealing with offshore wind farms as a whole. Namely, (i) mapping uncertainty and its propagation through the offshore wind energy system and (ii) exploring the effect of model fidelity on wind farm simulation. However, its usefulness as a tool for multiple purposes soon became apparent, which resulted in its more informed and serious development.

Other tools that enable practitioners in the field to keep the overview may be conceived. They could be quite different from OWFgraph, consider for example a domain-specific collaborative wiki, which has important implications for the possible use cases it enables. So OWFgraph should effectively be seen as a proposal, the first of its kind in the wind energy domain. This paper gives a thorough description of OWFgraph, brings to the fore its qualities and limitations, and provides reflection. It does not provide an analysis of effectivity, as its use as of yet has been too limited for that. It does provide a very concrete starting point for further exploration, test cases, and discussion in the community.

An overview of the its contents and why that content is included provides a guideline through the paper: To make the content of the database accessible, it should be effectively structured. It starts by describing how the content is structured (Sect. 2). Once this structure is in place, content can be added and queried in an informed way. However, the agreed-upon structure does not prescribe how content must be added, which means that representation challenges arise for some parts of the domain. So next it gives a view of the database's current content and highlight the challenges encountered (Sect. 3). The database can be queried in different ways, so to also give an idea about the possible uses, the paper discusses a number of use cases: computational pathway discovery, tool interoperability, defining disciplines, and education (Sect. 4). The paper ends with our conclusions about OWFgraph and a vision for its future (Sect. 5).

This introduction closes with Fig. 1, which gives a first impression of OWFgraph. When using a graph database, it is natural to visualize its content by showing excerpts from the graph. (The whole graph is too large in terms of nodes and edges to be usefully shown in its entirety.) The amenability to such visualizations actually greatly contributes to the usability of the database. This paper makes extensive use of graph visualizations. An example is shown in Fig. 1. (The meaning of the different elements of this visualization is discussed later in the paper.) It shows the relationship between the power output of an offshore wind farm—OWF—and its constituent wind turbines—WT. The involvement of the electrical connection system—elec. connect. syst.—appears through the transmission cables and the cable endpoints through which the power flows.

2 Structure

This section discusses the structure of OWFgraph. It starts from the basics in Sect. 2.1, describing the general concept of a graph database and more specifically the features we use. Then Sect. 2.2 details the structure tailored specifically to our needs—the database schema and in particular our 'foundational ontology'—designed on top of the basic graph database features. Next, Sect. 2.3 discusses the use of the graph database labeling feature in ways that fall outside the structure prescribed in Sect. 2.2. Finally, Sect. 2.4 discusses the practical software implementation that we used.

Wind Energ. Sci. Discuss., https://doi.org/10.5194/wes-2019-17
Manuscript under review for journal Wind Energ. Sci.
Discussion started: 2 May 2019

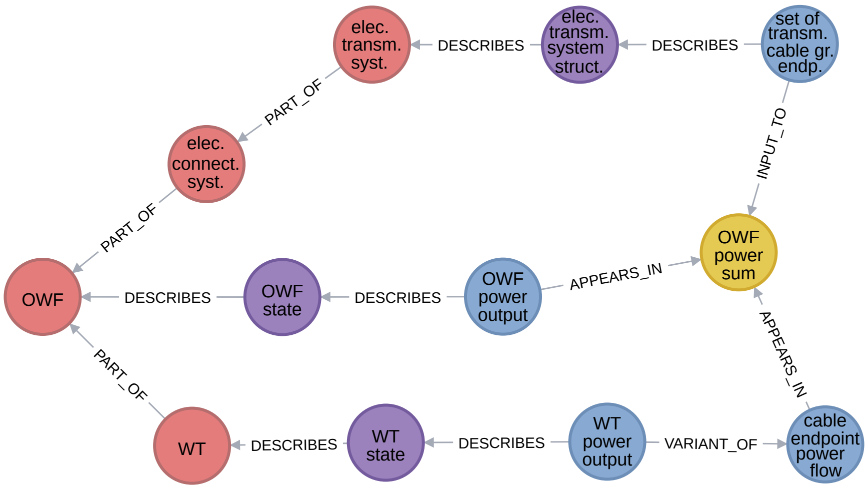


Figure 1. Visualization of an excerpt of the content in OWFgraph.

2.1 Graph database features

A graph database is in the widest sense the combination of graph database software and an amount of content stored and managed by this software. Content is discussed later. Here the focus lies on the conceptual aspects of the software that make it a graph database and not on the actual implementation as software.

A graph database is built on the concept of a graph, which has proven itself as a useful way of structuring information. A graph consists of *nodes*—also called vertices—connected by *edges*. In Fig. 1, discs represent nodes and arrows between nodes represent edges.

There are multiple variants of the mathematical graph concept that differ, for example, in whether some or all edges are directed or not, that is, whether they point from one node to the other. The graph variant used is a *directed graph*, in which all edges are directed. There is at most one edge between any pair of nodes and there are no loops—edges between a node and itself. Because of this, *ordered* pairs of start and end nodes *uniquely* define edges.

For building a useful graph database, one needs to be able to add content in addition to the graph structure. For both nodes and edges, an arbitrary number of *properties* can be added. These are key-value pairs that can be used to store content specifically associated to the node or edge. Moreover, a unique *relationship type* must be associated to every edge. Furthermore, zero or more *labels* can be added to nodes. A graph with such content-carrying features is called a *labeled property graph*. In Fig. 1, relationship types—in all-caps—are overlaid on the edges, node labels determine color-coding of nodes, and the value of the property ‘name’ is, to the degree possible given the available space, printed on the node discs.

The features discussed above provide enormous flexibility in creating the graph structure and adding content. In fact, without further restrictions, collaboration on adding material to the database and useful extraction of data is next to impossible. The next section therefore describes the rules put in place to make consistent use of OWFgraph possible.

Wind Energ. Sci. Discuss., https://doi.org/10.5194/wes-2019-17
Manuscript under review for journal Wind Energ. Sci.
Discussion started: 2 May 2019

2.2 Database schema & foundational ontology

2.2.1 Introduction and overview

An important task in the creation of a knowledge base is to define the way in which its content is to be structured. The graph database *labeled property graph* model described above provides features and some structure already. This section describes the further rules put in place to specify consistent addition of material. This enables discoverability of the database's contents by someone who is informed about both the labeled property graph model and these additional rules.

In classical—often relational—databases, the description of the database structure, including such rules—often called constraints—, is its *schema* (see, e.g., Silberschatz et al., 2011). In the database world, this is widely understood as a formal machine-readable specification that is automatically enforced. Graph database software is often mostly schema-less in this formal sense. Nevertheless, a specification of the structure in natural language that users need to adhere to can act as an informal schema.

OWFgraph's database schema consists of two conceptually separable parts: The first is a so-called *foundational ontology* for the domain; it defines finite sets of node categories (implemented as labels) and relationship types (see, e.g., Staab and Studer, 2009). The design of (foundational) ontologies has become a proper research topic in some fields, such as cyber security (Iannacone et al.). The second is an enumeration of the properties that can or should be attached to nodes and relationships. In practice, both parts cannot be completely separated, as the prescribed properties depend on the category and relationship type. The reference diagram in Fig. 2 provides a schematic overview of the categories and relationships defined by the foundational ontology. It also mentions the properties that can be attached to nodes and relationships.

Below, subsections discuss each of the concepts mentioned above and appearing in Fig. 2. Section 2.2.2 describes the categories of OWFgraph's foundational ontology, Sect. 2.2.3 describes its relationships, and Sect. 2.2.4 describes the properties. Sect. 2.2.5 closes with some reflections.

2.2.2 Categories

When describing a domain, such as offshore wind energy, its concepts can be categorized. Such a categorization is the basis for the foundational ontology. In the specification of the categories, expressivity must be balanced with simplicity. Expressivity pushes towards a larger number of categories. Simplicity favors a smaller set of categories that can be kept in mind by human users. The set of categories decided upon was determined by

1. the original use cases, which required a description of *models* for offshore wind energy subsystems and their interconnections through *variables*, and

2. the need to contextualize those models by relating them to the physical systems themselves.

This prompted a first division into two overarching classes, dubbed *the virtual world* and *the real world*.

Each of these two classes consists of a number of mutually exclusive categories. For the virtual world, these are

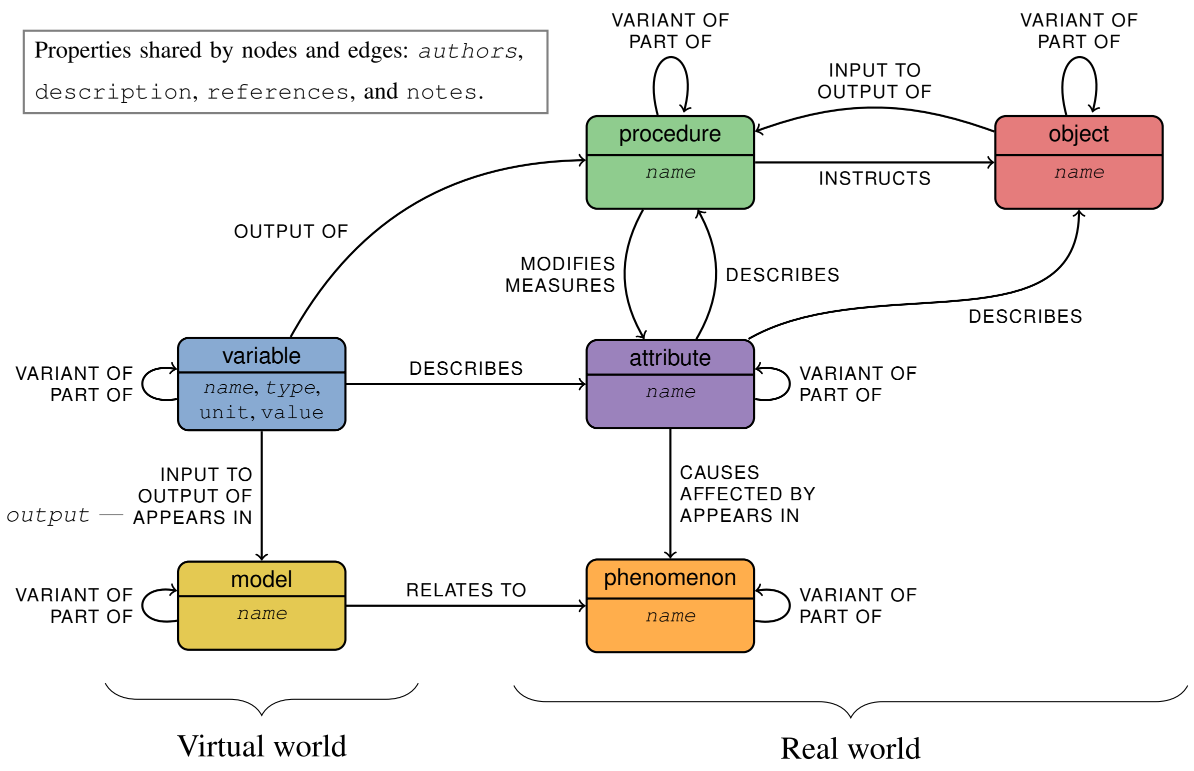


Figure 2. A schematic overview of the categories, relationships, and (*required*) properties defined by the database schema.

– models, encompassing both mathematical models as well as computational tools, and

– variables, covering the variables, parameters, and constants used as inputs and outputs for the models.

For the real world, these are

– objects, representing all physical things,

– procedures, describing concrete implementations of planned processes,

– attributes, specifying objects and procedures, and

– phenomena, categorizing physical processes.

Table 1 gives illustrative examples for each of the categories. In the graph database, labels are used to encode categories.

As already mentioned, in Fig. 1 colors correspond to labels, and therefore colors correspond to categories; from left to right we can see objects (red), attributes (purple), variables (blue), and models (yellow). Figure 2 shows the categories as the nodes of the diagram, with virtual world categories on the left and real world categories on the right. Figure 2 shares its color-coding with Fig. 1 and other graph extracts.

Even if the virtual world is the important part for the original use cases—but perhaps not for other potential uses of the database—, the real world creates a mind map that supports users. This holds both for adding content to the database and querying the database. For example, having a 'wind turbine rotor' object node as a starting point helps adding the variables and models that relate to it. The other way around, it also allows someone else to easily discover the variables and models in the database related to it.

2.2.3 Relationships

Next to restriction of nodes to certain categories, the foundational ontology also constrains the relationships between nodes, in a category-dependent manner. For example, from their description above, attributes clearly have a special relationship with members of other categories—objects and procedures. But this is just the tip of the iceberg. The next few paragraphs go over the restricted set of relationships that have been defined and how they may be combined with the different categories. All defined relationships are included in the diagram of Fig. 2. Their names are attached to the arrow connecting the categories between which the relationships are defined.

There are two intra-category relationships that can be used for all categories:

- *is a* PART OF, to decompose concepts into subconcepts, and
- *is a* VARIANT OF, to list *more specific* variants of a concept.

These two relationships predominantly tend to create tree-shaped subgraphs. Namely, nodes are generally only PART OF or a VARIANT OF a single parent. However, this is not a strict requirement and for some attributes it is even very sensible for them to be PART OF two parents. Figures 3 and 4 show excerpts of the subgraphs defined by the PART OF and VARIANT OF relationships. Figure 3 shows a tree-structured subgraph for object (on the left, red nodes) and an excerpt from the subgraph for attribute (on the right, purple nodes) in which a node with multiple parents appears. Figure 4 shows subgraphs for objects (top left, red nodes), attributes (top right, purple nodes), and models (bottom, yellow nodes).

Table 1. Example concepts for each of the categories defined.

Category	Examples
object	atmosphere, composite blade, generator, monopile, pitch controller, RNA (rotor-nacelle assembly), substation, tower
procedure	OWF (off-shore wind farm) decommissioning, OWF maintenance, RNA assembly, WT (wind turbine) maintenance
attribute	blade geometry, decommissioning cost, electrical cable state, grout structure, site location, turbulence, waves
phenomenon	(none yet)
model[a]	ECN Install, Katić mixed wake model, power coefficient, rotor swept area, wake speed deficit, WT available power
variable[b]	blade length, farm-average availability factor, LPC (electricity levelized production cost), OWF rated power

[a] Models are often named after the variable they are meant to compute. [b] Variables often have the same name as the attribute they describe.

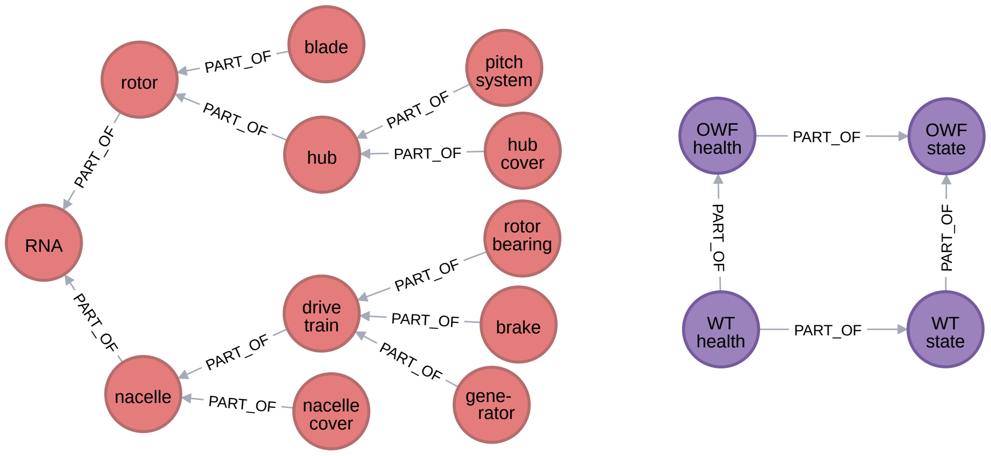


Figure 3. Excerpts of the subgraphs defined by the PART OF relationship.

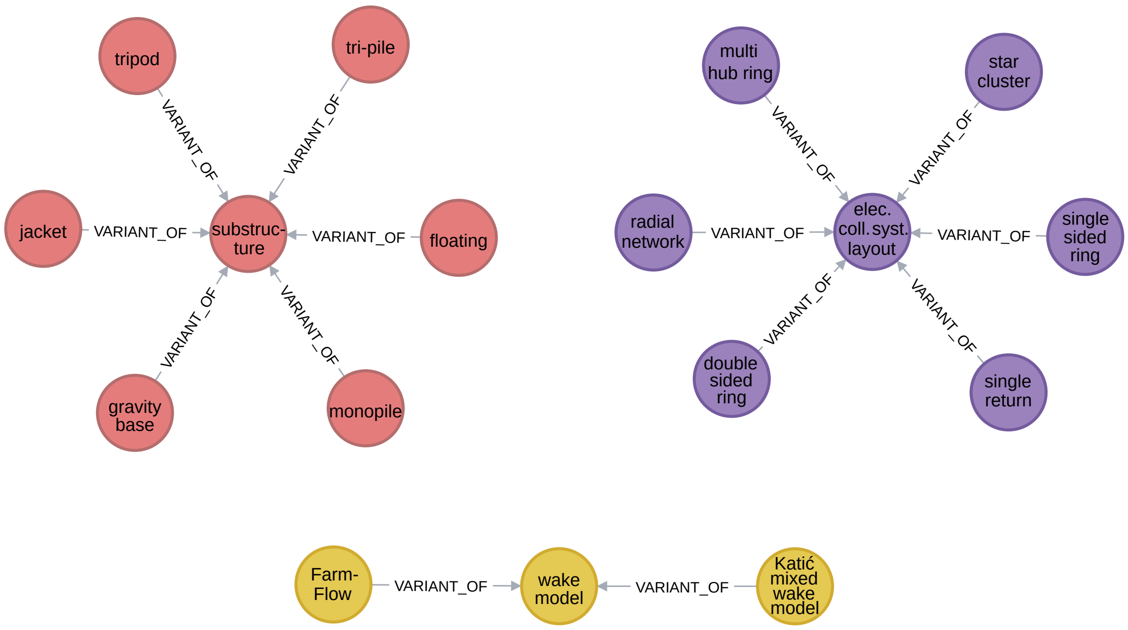


Figure 4. Excerpts of the subgraph defined by the VARIANT OF relationship.

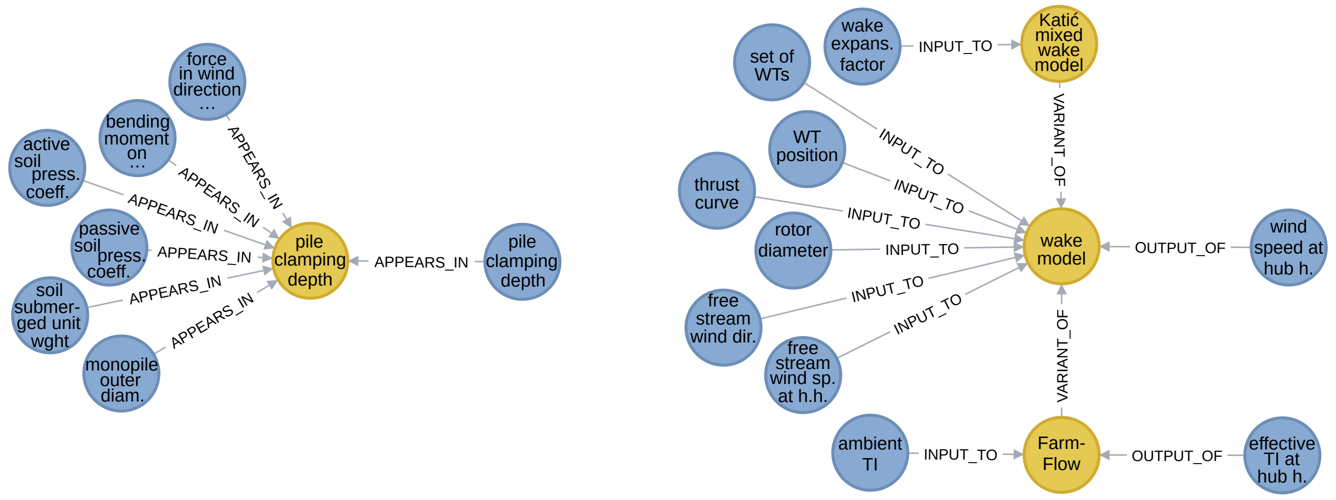


Figure 5. Excerpts of the subgraph of model/variable pairs.

All the other relationships are inter-category ones. They are also specific to the actual categories of the nodes they connect, although names are reused for similar relationships—cf. DESCRIBES in Fig. 2.

In the virtual world, the inter-category relationships between models and variables express which variables appear in which models and what their role is:

– a variable is an INPUT TO a model, and cannot be an output,

– a variable is an OUTPUT OF a model, and cannot be an input, and

– a variable APPEARS IN a model, to express that it can be both input and output, depending on the use case of the model.

Recalling that the model category comprises both mathematical models and computational tools, it should come as no surprise that APPEARS IN is mostly used for the former and OUTPUT OF and INPUT TO are mostly used for the latter. Figure 5 shows excerpts of the subgraph of model/variable pairs. Specifically, on the left-hand side, it shows the use of the APPEARS IN relationship with a mathematical model (an implicit one, for pile clamping depth). On the right-hand side, it shows the use of INPUT TO and OUTPUT OF relationships for computational tools (farm wake models, in this case).

Attributes form the main connection between the real world categories object and procedure and the virtual world category variable. A single relationship name, DESCRIBES, is used for all the relationships between these categories. Namely,

– a variable DESCRIBES an attribute,

– an attribute DESCRIBES an object or procedure.

Figure 6 shows excerpts of the subgraph defined by the DESCRIBES relationship. On the left-hand side, it shows how the offshore wind farm object is DESCRIBED. On the right-hand side, it shows how the offshore wind farm installation procedure is DESCRIBED.

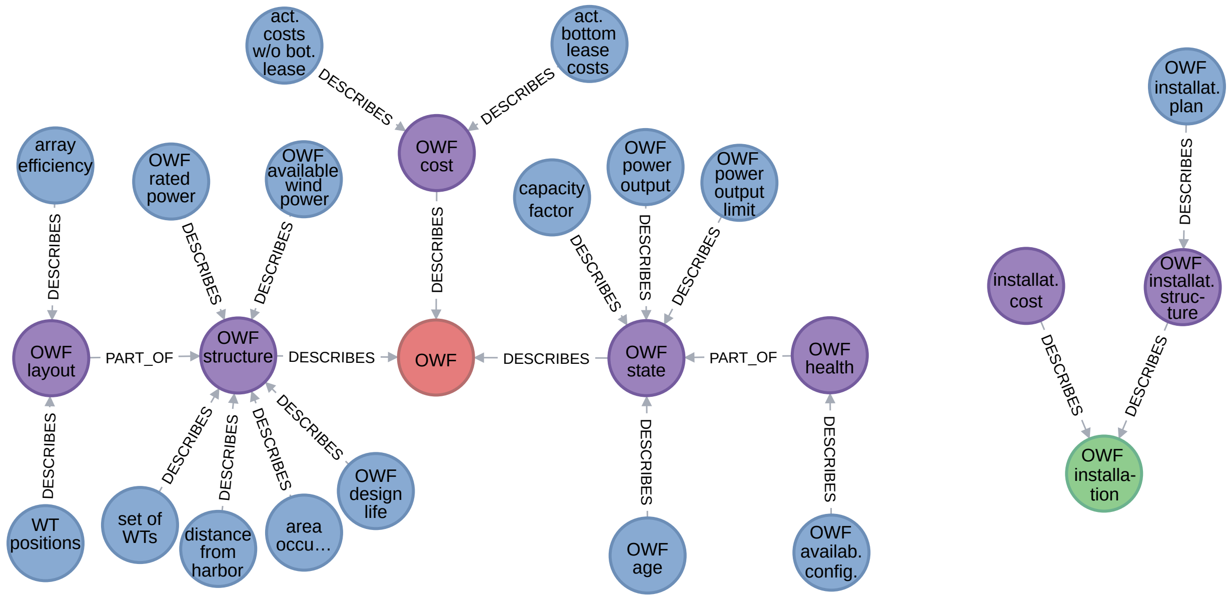


Figure 6. Excerpts of the subgraph defined by the DESCRIBES relationship.

Procedures effect change in the real world. To describe how this is done, a number of Procedures-specific relationships have been defined. Shared for all procedures is that

– a procedure INSTRUCTS an object (a crew, for example).

This object performs tasks prescribed by the procedure. Depending on the kind of task, specific relationships are used.

– For changes to which objects are present in the wind energy system, we have
  – objects as INPUT TO a procedure, and
  – objects as OUTPUT OF a procedure.
– For changes to attributes, we have that
  – a procedure MODIFIES an attribute.
– For measuring attributes, we have that
  – a procedure MEASURES attributes, and
  – variables as OUTPUT OF a procedure.

  This relationship between variables and procedures represents the only actual relation between the real world and the virtual world. It represents that measurements and observations in the real world produce data that are used as input to the models.

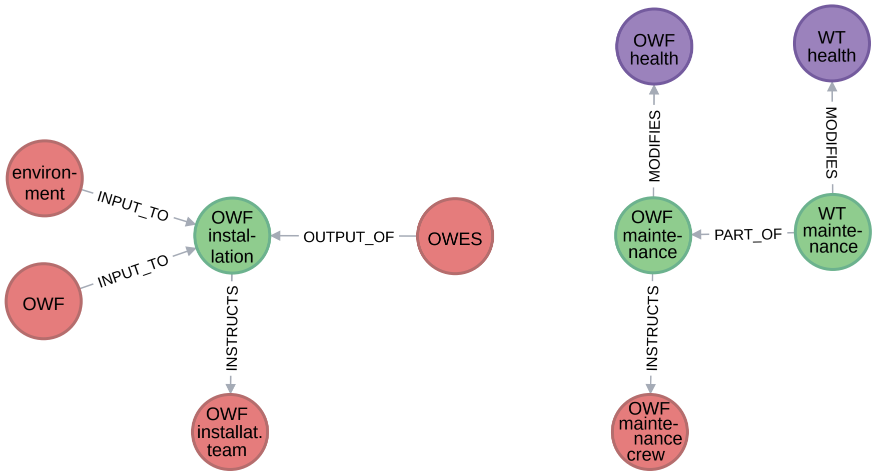


Figure 7. Excerpts of the subgraph defined by the INSTRUCTS relationship, and its context.

Figure 7 shows excerpts of the subgraph defined by the INSTRUCTS relationship, and its context. On the left-hand side, it shows a procedure where an object is created, namely the 'Offshore Wind Energy System'. On the right-hand side, it shows procedures where attributes are modified.

One could argue that it is not the procedure that MODIFIES or MEASURES, but the object that has been INSTRUCTED. However, having procedure as the nexus for all relationships involved provides for more easily recognizable and less ambiguous connection patterns.

The last set of relationships are connected to phenomena. As can be seen in Fig. 2, phenomena and attributes mirror models and variables in terms of relationships. Namely,

- an attribute CAUSES a phenomenon, but is not itself affected by it,
- an attribute is an OUTPUT OF a phenomenon, but not a cause, and
- an attribute APPEARS IN a phenomenon, to express that it can be both a cause and effect.

A direct connection with the virtual world can also be made:

- a model RELATES TO a phenomenon.

In this section, it has become clear that in the foundational ontology, the categories and relationships are not defined independently, but inextricably linked.

2.2.4 Properties

Section 2.2.2 provided categories for the concepts of the offshore wind energy domain. It did not touch upon the practical concern of how concept names or descriptions are attached to the nodes that represent them. This is done by adding *properties* to the nodes (cf. Sect. 2.1).

To facilitate manageability and discoverability, only properties from a predefined set are allowed. Moreover, some properties must be present on all relationships, all node categories, or specific node categories.

The properties allowed on all nodes and relationships are:

– `authors` (list of strings), listing the email addresses of the people that have added or modified that element in the database, to provide a pointer on whom to contact for clarification,

– `description` (string), containing a description of the concept or a clarification of the relationship in one or a few sentences,

– `references` (list of strings), containing one or more references to the literature where more information about the concept or relationship can be found, and

– `notes` (list of strings), providing information about recognized issues with how the element fits into the database or is described, meant as a pointer to correct this issue.

Of these, `authors` is required for all nodes and relationships, so that 'ownership' of all database content is made explicit. Also, a `description` is—in principle—required for all nodes. `Descriptions` of models may include mathematical expressions, as long as the variable symbols used are also explained therein.

Next comes a property that must be present on all nodes:

– `name` (string), containing a description of the concept in one or a few words.

To be unambiguous, the `name` must be unique for a given category. If this `name` is judged sufficiently informative for some concept, its `description` may be omitted. In `name`, uniformly used abbreviations for words common in the database are encouraged to improve their usefulness when visualizing the graph; for example, 'WT' for 'wind turbine'.

Furthermore, there are three properties specifically for variables:

– `type` (string), giving information about the set of values the variable belongs to, with typical values being 'bool', 'integer', 'rational', 'real', and 'string'.

– `unit` (string), listing the SI unit of the variable, which must be omitted for dimensionless quantities, and

– `value` (number or string), which holds a value if the variable represents a constant and must be omitted otherwise.

The last property defined is specifically meant for the APPEARS IN relationship between variables and models:

Table 2. Example values for each of the properties defined.

Property	Examples (separated by commas; lists are delimited by square brackets)
`authors`	[E.R.G.Quaeghebeur@tudelft.nl, S.SanchezPerezMoreno@tudelft.nl], [S.SanchezPerezMoreno@tudelft.nl]
`description`	'Vessel, labor and equipment costs to pull export cables through J-Tubes and connect to transformers'
`references`	[G. F. Moore (ed.) "Electric Cables Handbook" 3rd ed., Blackwell Science Ltd., 1997; Equation (8.1) on page 124.]
`notes`	[Add description., Perhaps this node does not belong here?], [This is assumed to be a constant in ECN Install.]
`name`[a]	'storm', 'switchgear cost', 'project development', 'downtime during preparation of repair of a WT failure'
`type`	'string', 'bool', 'integer', 'rational', 'real', 'real, between 0 and 1', 'positive real'
`unit`	W, °C, Hz, %, J, N, h, year, V, Pa, A, H, $^\circ\,'\,''$, $\mathrm{N\cdot m}$, $\mathrm{m/s}$, $\mathrm{kg/m^3}$, $\mathrm{rad/s}$, $\mathrm{F/m}$, Ω/m, $1/\mathrm{K}$, $\mathrm{N/m^3}$, $\mathrm{K\cdot m/W}$
`value`	86400, 2.3, -1.5, 'around 2', 'approximately 0.4', 'Zaaijer suggests $2.05\cdot10^6$ EUR(2003)'
`output`	true, false

[a] All the examples of Table 1 are (edited) `name` values.

– `output`, holding a Boolean value to indicate whether or not the connected variable is the usual output of the connected model.

Whereas the INPUT TO and OUTPUT OF relationships between variables and models unambiguously provide the role of the variables, APPEARS IN does not. The `output` property makes it possible to nevertheless encode the variable's typical role, which is essential to get a view of actual practice in the domain. It is therefore required to be present.

Table 2 gives illustrative examples for all of the defined properties.

The last remaining schema rule is a naming convention for attributes that DESCRIBE: they are either *state*, *structure*, or *cost* attributes. This means their `name` must be the `name` of the concept they DESCRIBE followed by one of those terms. This convention is used to force the many potential attributes into three sub-categories, as this gives a better overview. An example of how this convention is implemented is given in Fig. 8.

2.2.5 Reflection on the schema and foundational ontology

The foundational ontology and other aspects of the schema were in large part designed in a few brainstorming sessions. Afterwards, informed by experience adding material to the database, they were further tweaked. Even if our experience shows that the current schema in general and the foundational ontology in particular provide a useful basis for a knowledge base, it is not set in stone and may evolve further. This paper describes the first public version—'version 1.0.0'.

While the schema design is informed by practical experience, that was not the driver. Examples of this are the elements in the foundational ontology that do not yet appear in the actual database: the phenomenon category and MEASURES relationship. In fact, the foundational ontology should drive the way content is added to the database. It should lead to the database being accessible to a relatively wide audience within the offshore wind energy community. Therefore, it should be small and appre-

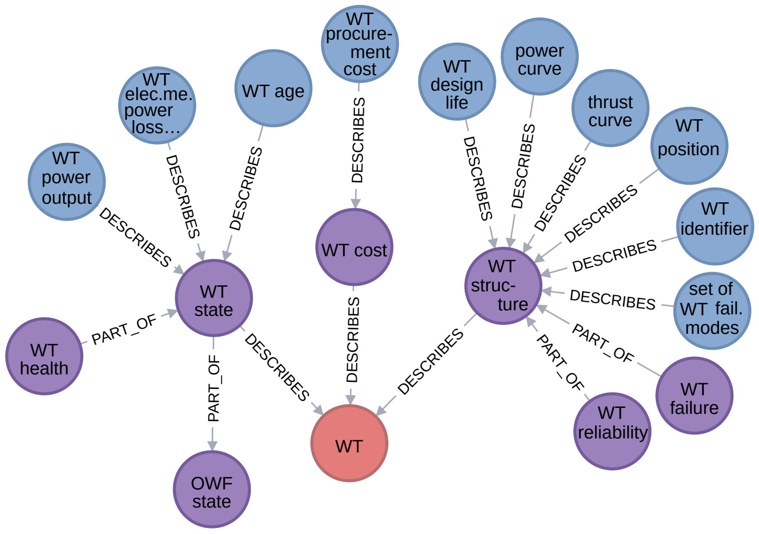


Figure 8. An example implementation of the top-level attribute state/structure/cost `name` convention.

hensible without constant reference to an extensive specification; a diagram such as Fig. 2 should be sufficient for day-to-day use.

The downside of using a small, apprehensible foundational ontology is that it may be rather crude. For example, the subcategories of attribute do not really partition the set of all possible attributes. This can be illustrated using Fig. 8: ‘wind turbine reliability’ is not completely structural—as implied by the current connections—, but has state aspects as well. Furthermore, the foundational ontology does not provide guidance on many practical domain representation issues one encounters while adding content to the database. Such issues will be discussed in Sect. 3. To underline the fact that our foundational ontology is indeed small, MarineTLO—a foundational ontology for the marine species domain—has 55 categories and 37 relationships (Tzitzikas et al., 2013), although that it must be said that such numbers cannot be comapared directly.

Even while keeping its content the same, the schema could have had a different structure. Namely, in a graph database it is always possible to replace a node property by a node containing the property value and a relationship expressing the property type. For example, the `unit` property of variables could be replaced by a unit category and a HAS UNIT relationship. The same remark can be made about labels, for example, using discipline nodes (cf. Sect. 2.3) and relationships instead of labels. Such a design would add more structure to the database contents, at the expense of increasing the number of categories, making the foundational ontology more complex.

2.3 Labels other than categories

Section 2.1 mentioned that an arbitrary number of labels can be added to nodes. The description of the foundational ontology in Sect. 2.2.2 already mentioned one application of labels: categories. Of these, one and only one must be added to each node. Labels are a convenient tool for other purposes as well. They appear in the database to indicate

- the *discipline* a concept belongs to (e.g., maintenance, electricity, mechanics),
- a related set of nodes whose addition to the database is being worked on, and
- the fact that a model or variable is 'internal' to some larger, overarching model.

Apart from not creating 'too many' labels, there is no guideline yet on what is and what is not a valid use case for non-category labels. Of the ones mentioned above, the labels indicating disciplines are meant to evolve into a standard set of disciplines which can be used as a categorization orthogonal to the one of the foundational ontology—more on this in Sects. 3.3.6 and 4.3. Whenever content is added to the database, labeling the nodes that are part of work-in-progress helps avoid edit conflicts and supports tracking them over multiple editing sessions. A reference to a common source (e.g., NREL cost breakdown) can for example be used for the label. When such work-in-progress content has been sufficiently integrated into the database, such a temporary label must be removed. Finally, the 'internal' label provides functionality that supports properly representing large, modular models—more on this in Sect. 3.3.5—, but it is not yet clear whether it is a sufficiently effective approach.

2.4 Practical implementation

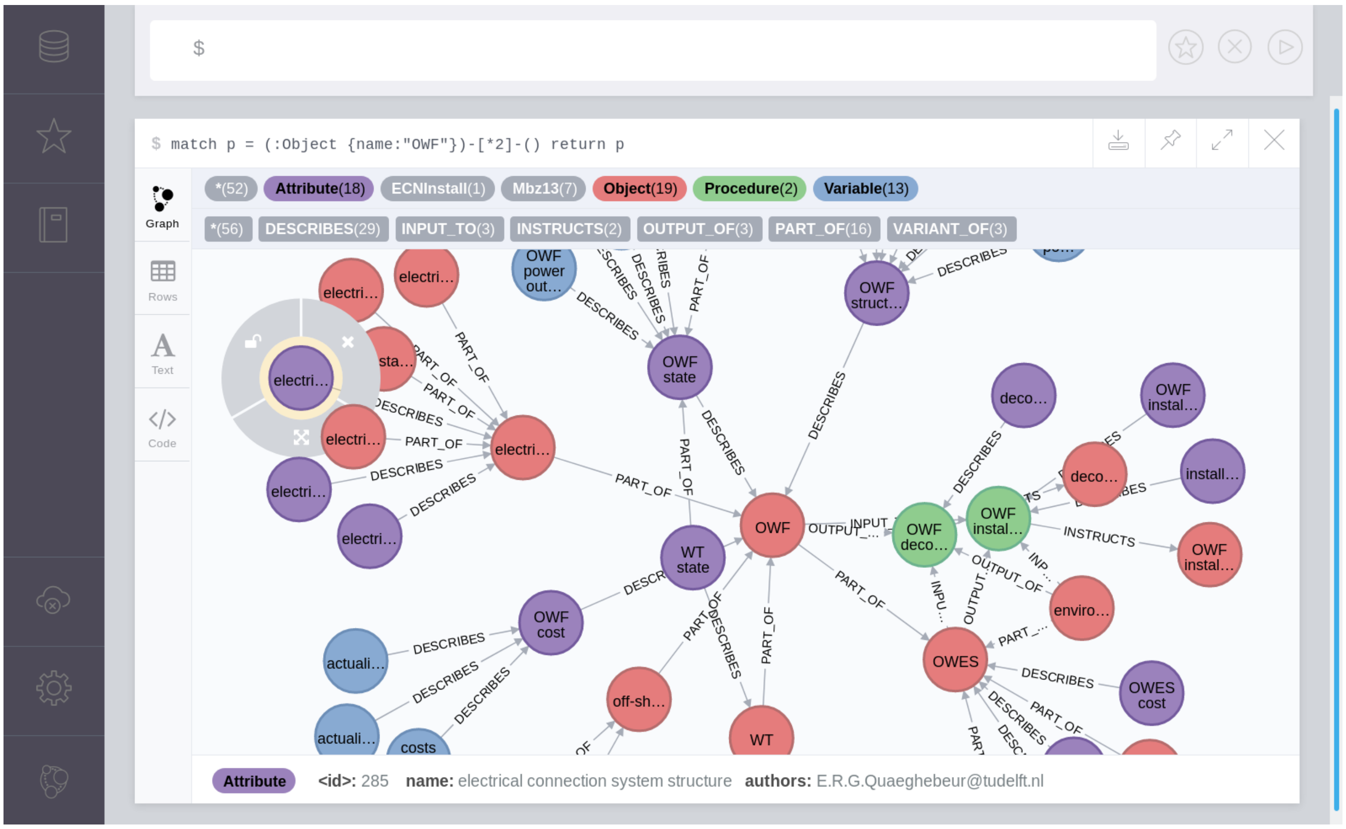
We have chosen to use the *Neo4j* graph database server software. It implements the labeled property graph model detailed in Sect. 2.1. It is cross-platform Java software available under both free and commercial licenses. Interaction with the server can be done through a provided internet browser interface, application programming interfaces to various popular programming languages, or via HTTP requests containing JSON-formatted messages (ECMA International, 2017). (HTTP, the HyperText Transfer Protocol, is the ubiquitous communication protocol of the World Wide Web and JSON, the Javascript Object Notation, is a widely supported text-based data format.) Commands for the server are formulated in *Cypher*, a graph database-specific query language (Neo4j, Inc., 2019), comparable to what SQL (Structured Query Language) is for relational databases (Silberschatz et al., 2011).

The main reasons for choosing this particular graph database software are

- the fact that it has a free license (AGPL version 3), taking away any cost considerations at this stage of development (FSF, Inc., 2007),
- its interactive web interface, which greatly facilitates interaction with the database, and
- its server nature, so that multiple users can access the same database concurrently and from different locations.

Furthermore, it has good documentation and the fact that it is one of the market leaders makes it straightforward to find support. We give a screenshot of the web interface showing interactive functionality—graph element selection—in Fig. 9. It shows the command line box for entering Cypher queries (top), a menu bar for accessing database-related information (left), and a query result window containing an interactive graph visualization. All graph visualizations in this paper have been obtained as exports from this interface.

It does have some limitations:


```
$ match p = (:Object {name:"OWF"})-[*2]-() return p
```

Attribute <id>: 285 name: electrical connection system structure authors: E.R.G.Quaeghebeur@tudelft.nl

Figure 9. A screenshot of the Neo4j web interface.

– The web interface provides no interactive editing of graph elements or properties. This must all be done using Cypher queries. However, there exist separate tools that make this possible; these have not been tested.

– There is only a very limited support for specifying a formal schema. Namely, only per-label property existence and uniqueness constraints can be added. Thus, most of the schema we specified must be respected through author discipline and periodic efforts by the maintainer to correct deviations from the schema.

The database structure and content does not depend on the currently chosen implementation. The structure is compatible with other databases and database types. The content can be exported to various machine-readable and human readable formats, such as JSON and GraphML (Brandes et al., 2002). The advantage of using graph database software, however, is that it includes functionality specifically tailored for working with graphs, such as shortest path algorithms. The advantage of using a database server is that effective collaboration on database content is possible.

3 Content

Section 2 gave a description of the structure of OWFgraph. Database content did make an appearance in the examples and illustrations, but was not the focus.

This section shifts focus to the content. This is not done by actually listing the contents, as that cannot be done in a manageable way; the actual database implementation is the most convenient approach to discover, browse, and otherwise query the database (access can be obtained by contacting the authors). It is done by giving statistics, in Sect. 3.1, listing the main sources for content and giving an overview of the content addition process, in Sect. 3.2, and focusing on specific cases where domain representation was not initially straightforward, in Sect. 3.3.

3.1 Database statistics

Currently, the database contains 1219 nodes and 1718 edges. Table 3 shows a breakdown over node categories and edge relationship types. The average in-degree and out-degree for the different categories are also included. (The in-degree of a node is the number of incoming edges and the out-degree of a node is the number of outgoing edges.)

Some facts that are immediately apparent from the table: Most database content is currently concentrated in the virtual world of models and variables. Also, the PART OF and VARIANT OF relationships form a substantial part of all connections made for all categories except for procedures. Furthermore, models have an average in-degree above four, meaning that on average more than four variables are connected to each model.

3.2 Sources for content

Initially, content was added reflecting the personal knowledge of the contributors, supported by standard references such as the books by Manwell et al. (2009), Burton et al. (2001), and the collection edited by Twidell and Gaudiosi (2009). This started by developing the subgraph for object PART OF object, which describes the breakdown of the physical wind energy system. This created the context for adding and connecting some models. It included adding a few procedures to see how they could be integrated. Throughout this initial stage, the foundational ontology changed based on the experience and insight gained.

The initial stage gave way to trying out various approaches for obtaining and adding further content. One was gathering information about models used in the field by interviewing domain experts. Specifically, the wind energy group of ECN (Energy research Centre of the Netherlands, now part of TNO) contributed in this way. This led to general and also ECN-specific additions such as their FarmFlow wake modeling tool, shown in Fig. 10.

In the first of two trials for adding large bodies of content, the models described in the PhD thesis of Zaaijer (2013) were imported. This was done in a semi-automated fashion: To start, models from the thesis and the variables involved were manually copied to a structured spreadsheet description. This description was then imported to the database using a script, adding about 150 models, 330 variables, and 600 relationships. Finally, connections between imported nodes and those previously present were manually added to integrate the material. This integration is still an ongoing process and is not trivial; it effectively prompted the discussion on model grouping in Sect. 3.3.5.

Table 3. Database content node and edge numbers breakdown.

Category	**Number**	**Avg. Degree**		**Source**	**Relationship**	**Target**	**Number**
		in	**out**				
model	205	4.2	0.22	model	PART OF	model	37
				model	VARIANT OF	model	6
				model	RELATES TO	phenomenon	0
variable	812	0.56	1.8	variable	PART OF	variable	410
				variable	VARIANT OF	variable	44
				variable	DESCRIBES	attribute	198
				variable	APPEARS IN	model	685
				variable	INPUT TO	model	89
				variable	OUTPUT OF	model	37
				variable	OUTPUT OF	procedure	0
object	93	1.7	1.1	object	PART OF	object	62
				object	VARIANT OF	object	29
				object	INPUT TO	procedure	5
				object	OUTPUT OF	procedure	4
procedure	6	2.7	1.2	procedure	PART OF	procedure	1
				procedure	VARIANT OF	procedure	0
				procedure	INSTRUCTS	object	4
				procedure	MODIFIES	attribute	2
				procedure	MEASURES	attribute	0
attribute	103	2.3	1.0	attribute	PART OF	attribute	25
				attribute	VARIANT OF	attribute	7
				attribute	DESCRIBES	object	67
				attribute	DESCRIBES	procedure	6
				attribute	CAUSES	phenomenon	0
				attribute	AFFECTED BY	phenomenon	0
				attribute	APPEARS IN	phenomenon	0
phenomenon	0			phenomenon	PART OF	phenomenon	0
				phenomenon	VARIANT OF	phenomenon	0
Total	1219	1.4	1.4		**Total**		1718

Wind Energ. Sci. Discuss., https://doi.org/10.5194/wes-2019-17
Manuscript under review for journal Wind Energ. Sci.
Discussion started: 2 May 2019
© Author(s) 2019. CC BY 4.0 License.

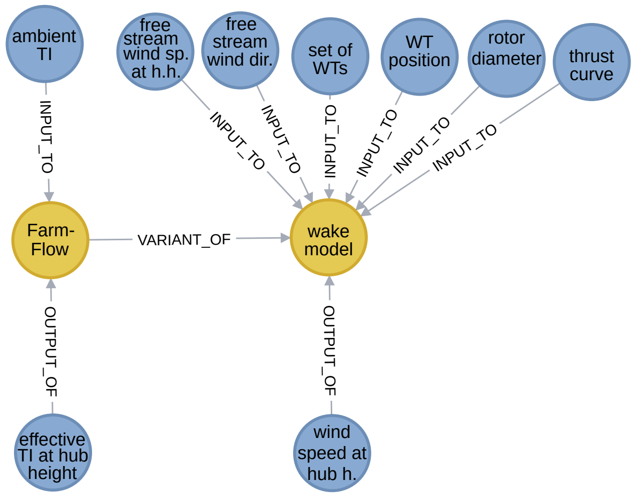


Figure 10. The model node for the FarmFlow wake tool and its surroundings in the graph.

The second large import was of the NREL cost breakdown for offshore wind farms (Moné et al., 2015, App. F). This time, the tabular structure could be readily copied to a spreadsheet and from there automatically imported using a script. Almost 350 cost variables were added in this trial, all part of a tree-shaped cost breakdown. This imported material has not yet been connected to the pre-existing content.

The experience with the two import trials showed that it is straightforward to add content, but that this is not the case for integrating the imported material. Integration usually requires adding new nodes that allow creating relationships with the existing and imported material and merging of imported and existing nodes to remove duplication. This is complicated by the lack of naming convention in the literature and often missing explicit definitions of variables such as cost components. Planning integration before importing can greatly reduce the effort required, as much of the needed work can be anticipated and taken into account in the spreadsheet structure and import script.

3.3 Non-straightforward domain representation cases

Our experience in adding content to the database has shown that adding a set of nodes with their properties and interrelations is straightforward and can be done relatively quickly. However, as mentioned in the previous section, it is the integration with the other database content that is time-consuming. Also, the content sources, such as a description of a set of models underlying a software tool (e.g., Zaaijer, 2013), provide a conceptually different view of the domain than the one presented in a knowledge base. For example, the choice of variables may be geared towards computation and therefore—from the knowledge base perspective—issues, such as duplication of variables, may be present. So, choices about how to deal with such issues need to be made, each of which needs to have a general applicability throughout the database.

This section therefore investigates the most important cases encountered where non-straightforward representation choices had to be made. Each of the choices made in these cases effectively corresponds to a set of guidelines that, while not part of

Wind Energ. Sci. Discuss., https://doi.org/10.5194/wes-2019-17
Manuscript under review for journal Wind Energ. Sci.
Discussion started: 2 May 2019
© Author(s) 2019. CC BY 4.0 License.

the schema or foundational ontology, should be followed in similar content representation situations elsewhere in the database. The reason is that they form consistency criteria that database users can rely on. For some of these cases, the issue at hand is still not resolved entirely satisfactorily.

3.3.1 Content harmonization

The content in the database is added by multiple people with different use cases for the database and originates from various sources. Even if the database schema and foundational ontology is adhered to, if no effort is made to integrate these additions, the database would become a disconnected bunch of graphs. This would defeat its purpose of being a *coherent* description of the offshore wind farm domain.

Therefore, content must be harmonized to achieve this goal. Concretely, there are the following guidelines:

– A concept may only be represented by a single node

– Tool-specific models and variables must be used sparingly and always described using PART OF or VARIANT OF relationships with generic models and variables.

Even while not being essential in their representation, the provenance of nodes should still be indicated in the `description` and `references` properties.

Examples from the database can illustrate both of the above guidelines.

Figure 11 gives an example of concept duplication that should be eliminated. At the top in the figure is an excerpt from the NREL cost breakdown (Moné et al., 2015)—top-level cost nodes for the offshore wind farm—and at the bottom an excerpt of the cost model of Zaaijer (2013)—top-level cost nodes for the offshore wind energy system—, both discussed in Sect. 3.2. The capital expenditure concept is duplicated, but not necessarily its PART OF children, so that some care needs to be taken merging these two subgraphs—this still remains to be done. (Note that the PART OF relationship here has been used to indicate decomposition into terms of a sum. Such usage is ambiguous and should therefore—despite its convenience—be eliminated adding intermediate sum models.)

Figure 12 shows how tool-specific nodes can be handled. The example of the ECN Install tool is used. Here, the tool-specific input variables are described as either a VARIANT OF a generic variable or decomposed into generic variables using the PART OF relationship.

3.3.2 Concept unicity & object multiplicity

The fact that each concept may only be represented by a single node has further implications. Namely, many objects in offshore wind farms come in multiple instantiations, such as wind turbines and electrical cables. The fact that there are multiple of them plays a role in many models, such as those for calculating wake deficits and electrical losses. An approach must be used that acknowledges the INPUTS of those models and respects the unicity of concepts.

The chosen approach is to introduce ‘set of’ variables to represent these—unique—sets of objects. Such set variables can be found in many of the graph excerpts used as illustration above: the ‘set of wind turbines’ in Figs. 5 (right), 6 (left), 10,

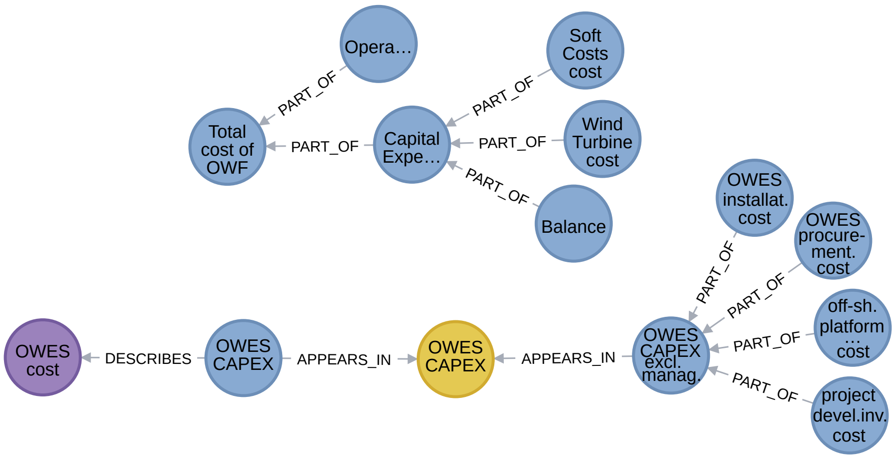


Figure 11. An example of concept duplication to be eliminated.

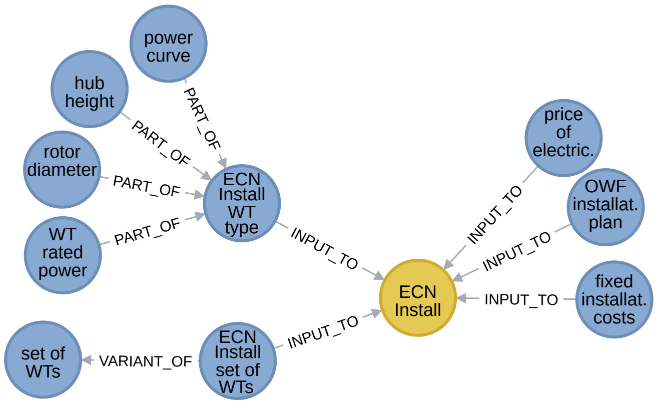


Figure 12. An example of limiting tool-specific variable nodes.

and 12; the 'set of transmission cable grid endpoints' in Fig. 1; the 'set of wind turbine failures' in Fig 8. To refer to a specific instance of an object of which multiple are present in an offshore wind farm, an 'identifier' variable is used, for example the 'wind turbine identifier' visible in Fig 8. However, no use of such a variable as input to or output of a model is yet made and they may still prove superfluous.

It is informative to take a closer look at the role 'set of transmission cable grid endpoints' plays in Fig. 1. It is used to calculate the offshore wind farm power output. Namely, it provides the information necessary to sum the right cable endpoint power flows. The reason for using cable endpoints and not cables in such a context is that the power flow in a cable is not constant due to losses, which we also need to be able to express.

3.3.3 Non-scalar variables

Scalar variables are trivial to represent, but this is not generally the case for non-scalar variables. Many models—certainly tools—deal with non-scalar variables of various types: vectors, matrices, lists of instructions, data sets, etc. Some of these non-scalar variables also are closely related to models and confusion may arise about how to include them in the database. Furthermore, when and how is a variable PART OF another variable?

First of all, anything that is considered a variable of some form by a model for a part of the offshore wind farm domain, is a valid variable to be included in the database. This of course includes constants, parameters, but also files with input and output in some tool-specific format. Some tools can be extended with scripts included in input files; such things are borderline cases. A useful discriminator here is that anything that adds substantial *functionality* to a tool cannot be a variable, but should perhaps better be dealt with using model groups (cf. Sect. 3.3.5).

Next, how should variables that essentially contain all the information necessary to define some model be dealt with? Despite the apparent redundancy, the idea is to include both and, if needed, make their relationships explicit. This is best clarified and explained using an example. The graph on the left in Fig. 13 presents a few models and variables related to the power curve of a wind turbine. The power curve itself (in the middle) is represented as a variable—think about a list of wind speed and power output pairs. Two related models are shown: The one at the bottom transforms wind speed values into power outputs in conformance to the information specified in the power curve variable. The one at the top represents an algorithm that takes wind speed and power output pairs and produces a power curve variable. (Neither of these models is specified in a detailed way, for example whether and how the conformity model interpolates; such detail is not needed in general, but could be added if some database use case requires it.)

Finally, a variable can be represented as a PART OF another variable if it is a *distinguishable* part thereof. Components of a vector and lines in a list or file are prime examples. Figure 13 illustrates this with excerpts from the graph: from left-to-right, it shows how cut-in and cut-out speeds are features of the power curve, how a power curve is part of the ECN Install wind turbine description input, how an (ECN Install) offshore wind farm installation plan consists of distinct installation steps, how scale and shape parameters are features of a Weibull distribution, and how the free stream wind velocity has speed and direction components.

3.3.4 Variable connections of variant models

Multiple VARIANTS OF a model can be added to the database. Even more so, it is actually a project goal to have such variants in the database, to support analyses where performances of alternative models are compared. However, such variants will share many variables. The question then becomes how to connect those variables to all the variants.

In this context it is useful to recall from Sect. 2.2.3 that a concept that is a VARIANT OF another concept should be more specific. If two variants cannot be ordered in this way, they should be represented as VARIANTS OF a more generic common abstraction. Therefore, a set of variant nodes is (partially) ordered from most generic to most specific.

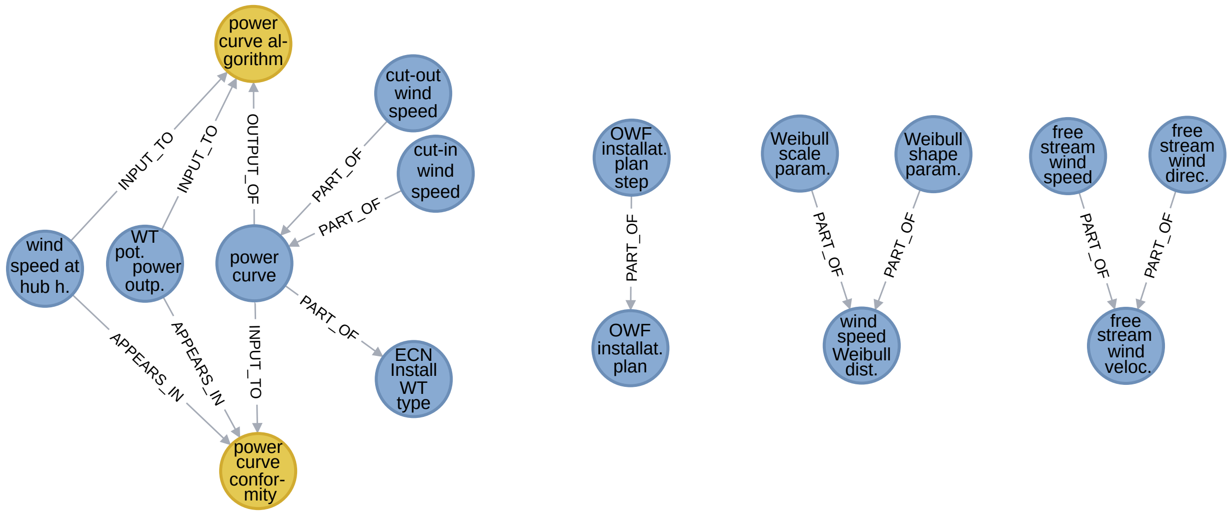


Figure 13. Examples of non-scalar variables and their surroundings in the graph.

In the context of models this ordering can be exploited to limit the number of connections to shared variables and make part of the specificity of each variant immediately apparent. Namely, *variables must be connected to the most generic variant in which they appear*. Then, to know all the variables involved in a specific variant, we can look at all the variables connected to that variant and any of its more generic 'ancestors'. Also, those variables connected to this specific variant are then immediately known to be specific to it and all its 'descendants' in the order.

To illustrate, Fig. 14 shows an example of an ordered set of models and their variables. Namely it shows a generic farm wake model (on the left) with its typical inputs and its output, the disturbed wind speed at hub height. It has two more specific, concrete variants, the classical farm wake model by Katić et al. (1987) and FarmFlow (Brand and Wagenaar, 2010), ECN's farm wake model. Both have an extra input variable, a wake expansion factor and the ambient turbulence intensity, respectively. The latter also has an extra output, the effective turbulence intensity at hub height. Furthermore, the Katić model also has a variant with wind speed rotor plane averaging.

The variable connection rule must actually be made more precise. Above, it was assumed that the relationship type of the connection is the same between a model and its variant. In case it is not, the connection must not be omitted. For example, a variable that APPEARS IN a model may be an OUTPUT OF a VARIANT OF that model. So the variable connection rule must be applied to variables *and* relationship types, not just to variables by themselves.

3.3.5 Groups of models

The detail with which models and tools should be represented in the database has not yet been specified. In principle any level of detail can be used. It corresponds to the size of the set of interconnected models and variables that are used for it in the database. At one extreme, a single model with all the 'externally' relevant variables corresponds to a high-level—rough—representation.

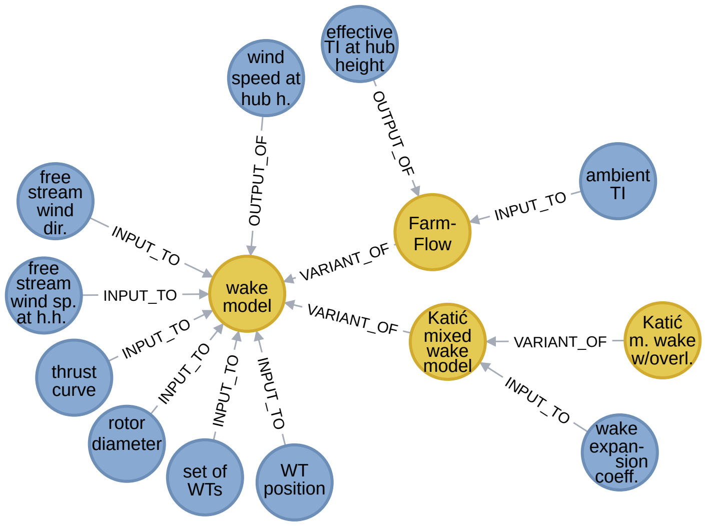


Figure 14. Example of an ordered set of variant models and their variables.

At the other extreme, a large set of models and variables, most of which are 'internal' in the sense that they are only connected to other models and variables within the set and not with externally relevant concepts, constitute a low-level—detailed—representation.

It can be useful to also have a high-level representation available for models that have been represented in high detail, for example to reduce complexity in analyses that do not require such detail. While part of such complexity may be hidden using specifically-tailored database queries, this cannot replace actually embedding domain knowledge. A natural way to implement the representation of models at different levels of detail would be to use a nested graph, namely, to embed a subgraph in a node—the overview model. However, support for nested graph functionality is not (commonly) available in graph database software, so an alternative is needed.

The approach implemented is the following:

1. Create the overview model node.
2. Add all models and variables of the low-level description, their interconnections, and connections to outside variables.
3. Connect all the low-level description models as PART OF the overview model and label them internal.
4. Label as internal all the low-level description variables that are OUTPUT OF some internal model. (The OUTPUT OF here should be understood to also include APPEARS IN with `output` set to true; the idea is to identify all variables that are produced internally.)
5. Connect all non-internal variables connected to internal models now also to the overview model. Care must be taken to use the right relationship type, as it may change from the one used for the low-level connection it mimics. Namely—going into the details—,

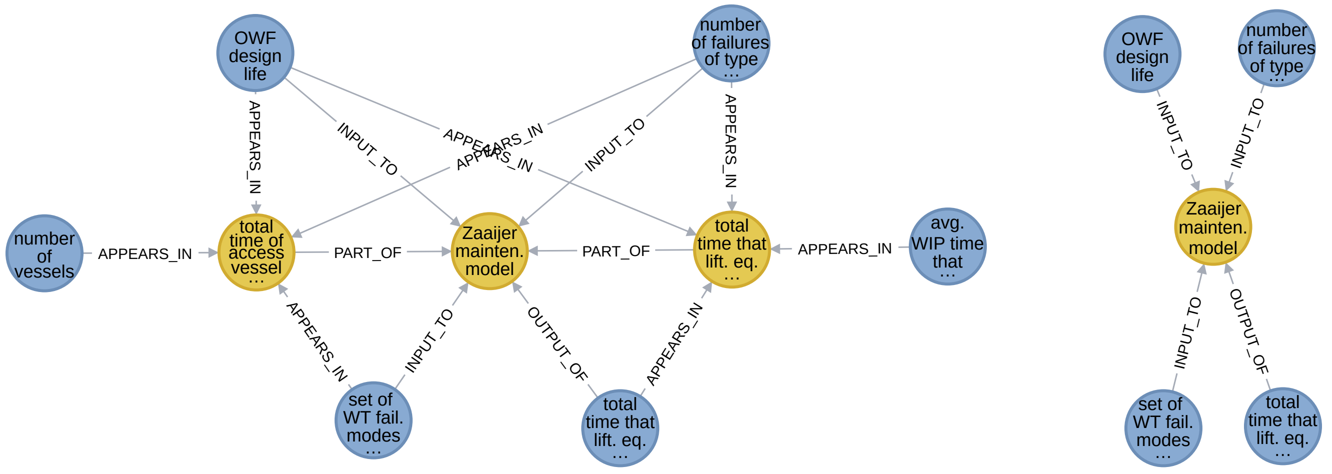


Figure 15. Excerpt showing an overview model and part of its low-level description (left). The same, except with internal nodes removed (right).

– when OUTPUT OF is used in the low-level description, use OUTPUT OF, and

– when INPUT TO is used in the low-level description, use INPUT TO,

– when APPEARS IN is used in the low-level description, use

  – OUTPUT OF when the low-level relationship's `output` property is 'true', and

  – INPUT TO when the low-level relationship's `output` property is 'true'.

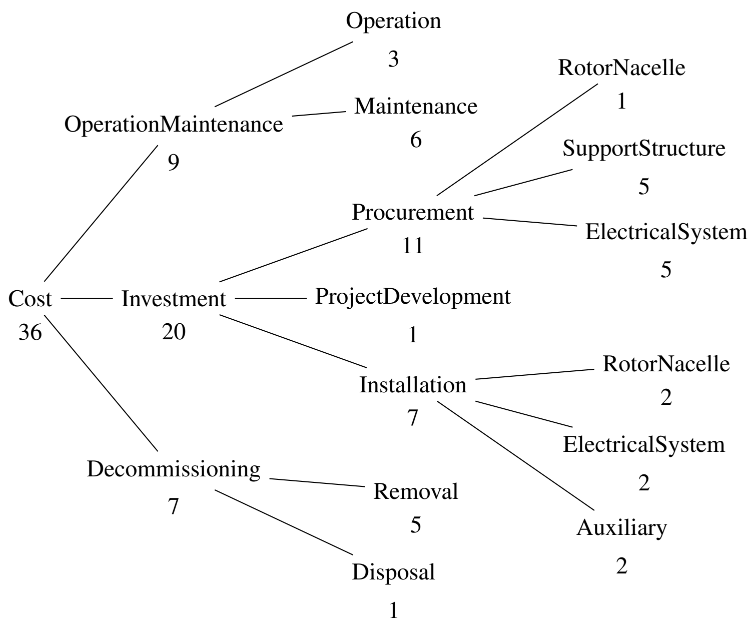
The reason for this is that internal variables are unavailable when dealing with the overview model and the input-output flexibility of an APPEARS IN relationship requires their availability. The overview model effectively behaves like a tool.

This approach makes it easy to filter out the internal nodes using the internal label and overview model. But at the same time, the low-level description is easily accessible. However, when adding connections from internal variables to external models, its internal label must be removed and appropriate connections to the overview model must be added.

The above approach was applied to the maintenance model of Zaaijer (2013, pp. 239-243), which consists of 37 models, 36 variables, and 230 relationships. Of these variables, 34 are not internal. This is too big to show in its entirety, but Fig. 15 on the left gives an illustrative excerpt. In the middle is the overview model, with an internal model both to the left and to the right. On the extreme left and right are two internal variables and above and below four external ones. For these external variables, notice the difference in relationship type for connections with the internal model and with the overview model. Figure 15 on the right shows the same model, but now with internal nodes removed.

Table 4. Discipline and cost category labels used. (The prefix 'Cost' has been removed from lower-level cost categories for brevity.)

Discipline	Number
Electricity	24
Mechanics	15
Hydrology	6
Geophysics	1
Rocks	1
SiteConditions	10
ArrayEfficiency	1
AerodynamicLoading	2
HydrodynamicLoading	5
GravityLoading	1
Maintenance	49



3.3.6 Assigning disciplines

The offshore wind farm domain is multidisciplinary. This means that the activities involved are diverse with respect to the background of the people performing them. When using the database, it can be useful to focus on just one or a few of these disciplines. Therefore, as mentioned in Sect. 2.3, *discipline labels* have been added. These were applied to the models imported from the thesis of Zaaijer (2013)—as discussed in Sect. 3.2—and correspond to the titles of the sections used there to list these models. These discipline names are listed in Table 4, where also the numbers of models that have been labeled as belonging to the discipline are listed; the disciplines that are effectively cost-categories are presented hierarchically, mirroring how they have been used.

Having such a set of disciplines is useful to focus on parts of the database. Namely, one can filter out nodes belonging to or not belonging to some disciplines. To that end, however, disciplines must be assigned to all nodes—or perhaps all models, which is currently not the case—and they should correspond to subsets of the domain that are meaningful to the users of the database. Furthermore, they should provide added value and not duplicate information already readily available. The set of disciplines used in this case was chosen in an ad hoc fashion and can be improved upon.

Actually, creating a proper set of disciplines is a task of its own, where not only a set of discipline names must be decided on, but also relationships between disciplines, and the assignment of disciplines to nodes. Our limited experience indicates that qua structure a partially ordered set of disciplines works well. For example, as can be gathered from the cost category decomposition on the left in Table 4, some nodes are effectively labeled with the 'Procurement' subcategory (of 'Investment') and also with ‘SupportStructure’. OWFgraph can actually support this task of creating a set of useful disciplines; Sect. 4.3 discusses how.

4 Use cases

The Introduction (Sect. 1) pointed out that a tool that enables people in the field of wind energy to keep the overview would be beneficial and it stated that OWFgraph has been developed to be such a tool. Sections 2 and 3 discussed the structure and content of OWFgraph. So now it should be clear why the database was created, what is in it, how it is stored there, and therefore also what can be extracted.

This section answers the question of how such a database can be used.

It presents a number of use cases; we are sure there are others. First is the one that actually triggered OWFgraphs development: the discovery of computation pathways—discussed in Sect. 4.1. Three others presented themselves during OWFgraph’s development: enabling tool interoperability—discussed in Sect. 4.2—, defining a coherent set of disciplines for the offshore wind energy domain—discussed in Sect. 4.3—, and finally education—discussed in Sect. 4.4. Each use case is delineated, the practical application of the database is detailed, and potential challenges are highlighted.

4.1 Computation pathway discovery

In an analysis or design of an offshore wind farm, a large number of models play a role. Furthermore, multiple sets of models can be used to achieve the same goals; call them computation pathways. The models in these pathways can vary in their fidelity and computational complexity. So the same analysis or design may be performed with varying computational time and result quality characteristics. Therefore it is useful to be able to discover such pathways.

Consider the example given in Fig. 16. It shows three pathways for calculating the power output potential of a wind turbine (on the right) starting from the wind speed at its hub (on the left). The topmost pathway uses the turbine’s power curve to directly calculate the output. The other pathways first pass via the power available in the wind at the rotor. Then the middle one directly computes the output from this using a power conversion factor. In the bottom pathway one more intermediate step is taken via the power extracted by the rotor, so separately considering the transformation from wind power to mechanical power and mechanical power to electrical power.

The subgraph of this example is the result of a multi-step process. The generic version of this process is the computation pathway discovery activity. The process consists of performing on-line queries on the database and iteratively modifying these queries until the sought-for outputs—subgraph visualizations, tabular listings, etc.—are obtained. This process is facilitated by the interactive functionality of the database sofware’s web interface, that is, the possibility of exploring the neighborhood of query result nodes in a point-and-click fashion. Next to exporting the result of this process, one can also save key queries in the process for later reuse, for example after new content has been added to the database.

The qualitatively distinct steps in the process are:

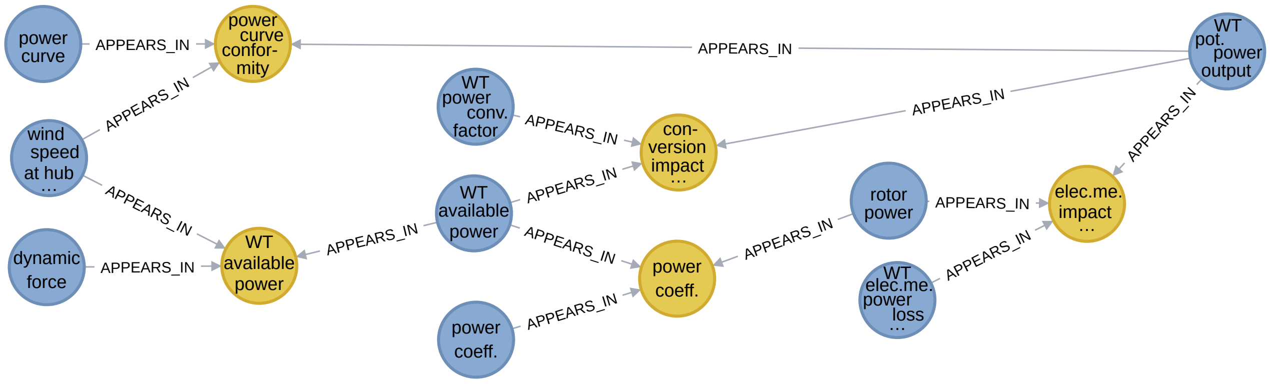


Figure 16. A simple example showing different pathways for computing the potential power output of a wind turbine (some edges omitted for clarity).

1. Decide which variables are the focus of the investigation. These are typically inputs—what one has available for computing with—and outputs—what one wishes to compute—, but can also involve intermediate variables. In the example, this would be 'wind speed' and 'power output'.

2. Perform a keyword-based query for each of these variables of interest, to see which concrete variables there are in the database. The query may need to be refined to get a more relevant set of matches. Based on the results, concrete variables are selected that will function as endpoints—or junctions—in the pathways. In the example, the initial keywords could be 'wind speed' and 'power', with, for example, the latter refined to 'WT power'. The selected variables are 'wind speed at hub height' and 'WT potential power output'.

3. Perform a query to search for a shortest path between endpoints traversing any junction nodes. In the example, the shortest path will be the top one, passing through the 'power curve conformity' model; it has a length of two.

4. Query for paths of increasing length, starting from the shortest path length. To keep the output manageable, queries looking for paths such as this one may often need to be refined to exclude or only include certain node categories or nodes from being present in the result. An upper limit on the length is decided on a case-by-case basis, typically informed by the results already observed. For the example, the node categories were restricted to variables and models, which is typical if the focus is really on computation. Furthermore, some nodes were excluded—such as the 'set of WTs' variable—that created paths that did not correspond to actual computational pathways. So in this part of the process, the middle length-four and bottom length-six pathways were discovered.

5. Finally, discover which other nodes are relevant using the interactive interface. This exploration of the neighborhood of pathways can be as extensive or concise as desired. In the example, only variables needed to complete the set of inputs of the individual models in the pathways such as 'power coefficient' were added in this way.

Not all calculation pathway analyses fit into the mold sketched above. For these, the process will remain generally similar, but should be adapted. An example is the situation where one considers only an output variable and is interested in sets of inputs that allow it to be computed. In this case, instead of starting from a shortest path between nodes and increasing the path length, one can start by considering incoming paths of length one and increase their length, again excluding unwanted paths by refining the query.

The calculation pathway discovery process is still a quite manual one and still requires some domain knowledge. But, it is greatly facilitated by the graph database functionality. Namely, compare it to what would need to be done without the database: a combination of a literature study, interaction with experts, and ad hoc graph-drawing to get a comparable result. Of course the quality of the result depends entirely on the amount of content in the database and the care with which it has been represented; this holds for this and all other use cases. However, the fact that the database can grow over time in a collaborative effort, is available to the entire community, and has multiple use cases, makes the investment worthwhile.

The discovery of computational pathways is particularly useful for informing the possibilities for system scope and model fidelity while building MDAO workflows.

Multidisciplinary Design Analysis and Optimization (MDAO) is a systems engineering technique that exploits the interactions between technical disciplines and the design of different sub-components, to solve trade-offs in benefit of the performance of the entire system. MDAO workflow is the term given to the coupling of computational models driven by algorithms that solve a use case, optimization, for example. The complexity of MDAO workflows can be categorized along three axes: system scope, model fidelity and MDAO architecture. System scope encompasses the choice of disciplines and sub-components that are modeled in the MDAO workflow, model fidelity stands for the level of sophistication of the models coupled in the MDAO workflow, and MDAO architecture is the logical flow of data between modules and between the algorithms and the modules.

Depending on the use case to solve, minimization of the levelized cost of energy (LCOE) of a wind farm with respect to the position of the wind turbines (layout), for example, one would typically be interested first in finding which disciplines or sub-components are affected by the wind farm layout, and which of these impact the LCOE. Of the entire list of variables, a wind farm designer can then shortlist those necessary to evaluate the performance of a particular sub-component, and select a subset of the variables deemed most impactful. Then, finding the paths between those variables and the LCOE variable informs which models (disciplines or sub-components) can make the link.

Concerning model fidelity, if there are two or more paths (two or more set of models) that link certain input and output variables, OWFgraph can provide what are the required inputs and the available outputs of each of these models. If the user has a list of variables at his disposal, a query of OWFgraph can inform what data is missing and which variables will not be needed.

Additionally, OWFgraph could be parsed by MDAO workflow-manipulating software, that automate the creation of workflows given a use case (cf. van Gent et al., 2017). Hence, if one can formalize the description of a use case using such software, by including what is the objective function, constraints and design variables, OWFgraph can provide the knowledge of which models can be coupled, and which would require more input data from the user.

4.2 Tool interoperability

There is a multitude of tools in use in the offshore wind farm domain, but no interoperability conventions yet. Some provide unique functionality and for some alternatives exist. They all work with the same general domain and therefore the many variables reappear as inputs or outputs. In principle they can be chained together in a workflow, or one tool can be replaced by another. However, this is hampered by the lack of interoperability conventions.

The advent of more integrated system-level design and analysis has brought to the fore the advantages of such conventions. Concretely, the current IEA Wind Task 37 'Wind Energy Systems Engineering: Integrated RD&D' (the *Task*)—which the authors are involved in—has as one of its objectives to "provide framework guidelines that will enable more seamless integration of analysis tools and reference models between organizations" (also see Sanchez Perez-Moreno et al., 2016). Specifically, in its 'Work Package 1' (the *Work Package*), the goal is to develop a common ontology ("hierarchical framework of characteristics") and a common data exchange format. The experience of the people working on this ontology and the data exchange formats is that it is a daunting task to be sufficiently comprehensive.

OWFgraph can facilitate efforts such as the one of the Work Package. First of all, it can provide the common ontology; namely, its subgraph of variables is effectively such an on ontology, structured through its connection with the 'real world' of objects, procedures, and their attributes. The tools considered by the Task are just model instances; using PART OF and VARIANT OF relationships they can be described to the level of detail required. The tool-specific inputs and outputs can be added as variables that are VARIANTS OF the tool-neutral 'common ontology' variables; this makes it possible to automate creation of common data exchange format descriptions, for example using the JSON Schema language (Andrews et al., 2018). These VARIANT OF relationships can carry—in their properties—the precise information necessary to transform 'common ontology' variable values to tool-specific variable values. Doing that opens the possibility of even automating data format translation.

The development of common ontology variables relates closely to the issue of integration of new tools in OWFgraph, discussed in Sect. 3.3.1. Duplication of concepts and name conflicts are solved by a naming convention for tool-neutral variables.

4.3 Defining disciplines

As stated before, the offshore wind farm domain is multidisciplinary. Section 2.3 mentioned that labels can be used to indicate the disciplines a node belongs to and that these can be used to filter the database content. Section 3.3.6 made this more concrete by showing what is currently present in OWFgraph in this regard. This section moves to the problem of defining a useful set of disciplines. Previous work in this area has been done by Sempreviva et al. (2017).

What constitutes a useful set of disciplines? We use the following criteria:

1. Individual disciplines correspond to what is typically understood to be a discipline, namely, a specific branch of knowledge, learning and practice. In other words, each discipline represents a subdomain that can stand on its own as an object of study.

2. The set of disciplines covers the whole domain.

3. Each discipline covers a non-trivial part of the domain—that is, disciplines that cover almost all domain concepts or just a few domain concepts provide no added value and would crowd the set of disciplines.

4. The difference between any pair of disciplines is also nontrivial in the sense that a sufficient number of concepts should belong to one, but not to the other. (This still allows *subdisciplines*.)

What comes to the fore in this set of criteria is a reliance on having an overview of the domain. This is what OWFgraph provides. How can it be used for the definition of a useful set of disciplines? The first criterion suggests that an initial proposal set of disciplines is formulated by domain experts; taking the taxonomy of Sempreviva et al. (2017), for example. These discipline terms are then applied to concepts in the database, either to all or to a specific subset, such as all models. This makes it possible to check the further criteria:

– In case some concepts are not covered by any discipline term, a discipline needs to be added or broadened in scope.

– In case a discipline covers almost all or just a few concepts, it can be removed, or, in the latter case, it can be decided that concepts need to be added to the database to improve its coverage of the domain.

– In case a pair of disciplines both cover essentially the same concepts, then they either need to be merged, or concepts need to be added to the database to differentiate the disciplines

– In case the majority of concepts in a discipline are connected to concepts in another discipline, it should be investigated whether the disciplines should be merged.

– In a discipline, if there are hardly any connections between one set of concepts and another, it may need to be split.

The second and third point make it clear that adding disciplines to OWFgraph can cause underrepresented domain areas to become apparent. For the last two points, clustering algorithms for graphs may be used.

When one arrives at a set of disciplines built from the above steps, each discipline effectively corresponds to a nontrivial and unique set of concepts. It may reflect biases present in the database—because some part of the domain is underdescribed, for example—, but these can be remedied by expanding the database. Biases and other defects in a set of disciplines defined solely on the basis of expert opinion are hard to even concretely discuss; the resulting arbitrary nature of the set will negatively impact its acceptance. A set of disciplines defined using OWFgraph can be viewed as a high-level summary of information present in the database, so of offshore wind farm domain concepts. The fact that the disciplines are backed in this way by concrete sets of concepts removes the arbitrariness mentioned earlier, improving its chances of acceptance.

4.4 Education

In education, OWFgraph can be used for domain discovery, that is, to learn about the different parts of the offshore wind farm domain and how they are related. For example, it is possible to discover

– the different objects that together form the physical offshore wind energy system,

– the different ways in which specific variables play a role, and

– the alternatives that exist for certain models.

One can imagine students doing a project in a certain subfield checking the database to see if they have not missed anything relevant—e.g., concrete variables, models, objects, and procedures—in their subfield or in related fields. This educational use case is not only applicable to students in the strict sense, but to anyone needing to familiarize themselves with a part of the domain.

In practice, users will start with a simple query to obtain a single or a few nodes. Then they can use the interface (cf. Fig. 9) to explore the connections of the nodes obtained. This exploration can be repeated to explore an arbitrarily large part of the database. The interface makes it possible to remove nodes from view, so the graph excerpt shown can be kept to a manageable size. For each of the nodes, their properties, especially the `description` and `references`, provide information that goes beyond what can be learned from the connections and `name` alone.

There is already enough content for pilots of this use case, but for actual use the content still needs to grow and mature.

There is a potential big advantage to this use case. During their explorations, the students will encounter issues, such as errors and omissions. If appropriate procedures and facilities are put in place—a bug tracker—, they can signal and help fix them. So this use case can actually help improve the quality of the database content.

5 Conclusion & Vision

This paper started by stating that a tool that enables researchers in the offshore wind energy field to keep the overview can provide great benefits. This paper describes OWFgraph, a knowledge base for the offshore wind farm domain, a tool made just for that purpose. Its conception as a graph database allows us to make use of functionality in existing software that facilitates access to the content (Sect. 2.4). The design of its structure balances the requirements of expressivity and simplicity, to enable a broad group of practitioners to both make use and contribute (Sect. 2.2). Although the database content itself was not a focus of this paper, it does discuss and provide solutions for non-straightforward domain representation cases (Sect. 3.3) to provide an accurate view of the difficulties that arise when adding content to the database. The paper describes a number of different use cases to make clear not only *what* OWFgraph is, but also *how* it can be used.

So, this work shows that it is possible to build a tool that helps people keep the overview of the offshore wind farm domain. The actual benefits of this tool are greatly dependent on the amount and quality of the database content. Our experience in adding data, as reported mainly in Sect. 3, makes it clear that adding content is a time-consuming but mostly one-time activity. Our current judgment is that the time investment—also for management—will only really pay off if a larger group of people contribute and make use of the database. Each contributor can focus on their areas of expertise, making content addition more time-efficient. Each successful use case increases the value of the database, and therefore decrease its relative time investment cost.

Therefore, we see the future of OWFgraph as a shared resource built by an international on-line community. Such a community can increase the accessibility of the database by creating effective documentation, providing instruction, and facilitating collaboration between people with a specific use case and people with experience in using the database. Such collaborations are important, because there is an appreciable learning curve, which will never completely disappear, even if more beginner-friendly software tools for accessing the database become available. A successful community building and maintaining a shared knowledge base can effectively avoid duplication of structured domain description efforts of various research and working groups that would otherwise create single-use spreadsheets.

Data availability. A JSON export of OWFgraph is made available as extra material for this paper.

Author contributions. All authors intensively discussed the structure of data stored in the database, which led to the creation of the current foundational ontology and most of the choices made about the representation of domain concepts. Erik Quaeghebeur initiated the concept of using a graph database to describe the offshore wind farm domain, created and manages the concrete implementation, and was the main author of this paper. Sebastian Sanchez and Erik Quaeghebeur added almost all content, in comparable amounts. Michiel Zaaijer had an engaged supervisory role.

Competing interests. We declare that no competing interests are present.

Acknowledgements. We had fruitful discussions with Amir Birjandi (currently at Green Kappa Consulting and Technology Inc.) concerning OWFgraph, while he was on a research visit to our research group in 2016. He also contributed material to the database at that time.

We visited the wind energy group of ECN (Energy research Centre of the Netherlands, now part of TNO) and had multiple enlightening interviews and discussions with its members. We thank them for their kind and useful collaboration.

References

Andrews, H. et al.: JSON Schema, https://json-schema.org/, 2018.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, p. 1247–1250, ACM, https://doi.org/10.1145/1376616.1376746, 2008.

Brand, A. J. and Wagenaar, J. W.: A quasi-steady wind farm flow model, in: Proceedings of the European Wind Energy Conference (EWEC 2010), 2010.

Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., and Marshall, M. S.: GraphML Progress Report: Structural Layer Proposal, in: Graph Drawing, edited by Mutzel, P., Jünger, M., and Leipert, S., vol. 2265 of *LNCS*, p. 501–512, Springer-Verlag, Berlin, Heidelberg, https://doi.org/10.1007/3-540-45848-4_59, http://graphml.graphdrawing.org, 2002.

Burton, T., Sharpe, D., Jenkins, N., and Bossanyi, E.: Wind energy handbook, John Wiley & Sons, Ltd., 2001.

ECMA International: Standard ECMA-404: The JSON Data Interchange Syntax, https://www.ecma-international.org/publications/standards/Ecma-404.htm, 2017.

Franceschini, A., Szklarczyk, D., Frankild, S., Kuhn, M., Simonovic, M., Roth, A., Lin, J., Minguez, P., Bork, P., von Mering, C., and Jensen, L. J.: STRING v9.1: protein-protein interaction networks, with increased coverage and integration, Nucleic Acids Research, 41, D808–D815, https://doi.org/10.1093/nar/gks1094, 2013.

FSF, Inc.: GNU Affero General Public License, https://www.gnu.org/licenses/agpl-3.0.en.html, 2007.

Himmelstein, D. S. and Baranzini, S. E.: Heterogeneous Network Edge Prediction: A Data Integration Approach to Prioritize Disease-Associated Genes, PLOS Computational Biology, 11, 1–27, https://doi.org/10.1371/journal.pcbi.1004259, 2015.

Iannacone, M., Bohn, S., Nakamura, G., Gerth, J., Huffer, K., Bridges, R., Ferragut, E., and Goodall, J.: Developing an ontology for cyber security knowledge graphs, in: Proceedings of the 10th Annual Cyber and Information Security Research Conference, p. 12, ACM, New York, NY, USA, https://doi.org/10.1145/2746266.2746278.

Jupe, S., Akkerman, J. W., Soranzo, N., and Ouwehand, W. H.: Reactome – a curated knowledgebase of biological pathways: megakaryocytes and platelets, Journal of Thrombosis and Haemostasis, 10, 2399–2402, https://doi.org/10.1111/j.1538-7836.2012.04930.x, 2012.

Katić, I., Højstrup, J., and Jensen, N. O.: A Simple Model for Cluster Efficiency, in: EWEC ’86, edited by Palz, W. and Sesto, E., vol. 1, p. 407–410, A. Raguzzi, 1987.

Manwell, J. F., McGowan, J. G., and Rogers, A. L.: Wind energy explained, John Wiley & Sons, Ltd., 2nd edn., 2009.

Moné, C., Smith, A., Maples, B., and Hand, M.: 2013 cost of wind energy review, Tech. Rep. NREL/TP-5000-63267, National Renewable Energy Laboratory, http://www.nrel.gov/docs/fy15osti/63267.pdf, 2015.

Neo4j, Inc.: https://neo4j.com/docs/cypher-manual/, 2019.

Sanchez Perez-Moreno, S., Zaaijer, M. B., Bottasso, C. L., Dykes, K., Merz, K. O., Réthoré, P.-E., and Zahle, F.: Roadmap to the multi-disciplinary design analysis and optimisation of wind energy systems, in: The Science of Making Torque from Wind (TORQUE 2016), vol. 753 of *Journal of Physics: Conference Series*, p. 062011, EAWE, IOP Publishing, https://doi.org/10.1088/1742-6596/753/6/062011, 2016.

Sempreviva, A. M., Vesth, A., Bak, C., Verelst, D. R., Giebel, G., Danielsen, H. K., Mikkelsen, L. P., Andersson, M., Vasiljevic, N., Barth, S., Sanz Rodrigo, J., Gancarski, P., Reigstad, T. I., Bolstad, H. C., Wagenaar, J. W., and Hermans, K. W.: Taxonomy and metadata for wind energy research & development, https://doi.org/10.5281/zenodo.1199489, 2017.

Wind Energ. Sci. Discuss., https://doi.org/10.5194/wes-2019-17
Manuscript under review for journal Wind Energ. Sci.
Discussion started: 2 May 2019

Silberschatz, A., Korth, H. F., and Sudarshan, S.: Database system concepts, McGraw-Hill, New York, 6 edn., 2011.

Speer, R., Chin, J., and Havasi, C.: ConceptNet 5.5: An open multilingual graph of general knowledge, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI, https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/viewPaper/14972, 2017.

Staab, S. and Studer, R., eds.: Handbook on ontologies, International Handbooks on Information Systems, Springer, Berlin, Heidelberg, https://doi.org/10.1007/978-3-540-92673-3, 2009.

Taymaz, H. M., Özyer, T., and Cangelir, C.: A knowledge-based semantic tool for standard part management in aerospace industry, in: 2013 IEEE 14th International Conference on Information Reuse Integration (IRI), p. 634–642, https://doi.org/10.1109/IRI.2013.6642528, 2013.

Twidell, J. and Gaudiosi, G.: Offshore wind energy, Multi-Science Publishing Co. Ltd., 2009.

Tzitzikas, Y., Allocca, C., Bekiari, C., Marketakis, Y., Fafalios, P., Doerr, M., Minadakis, N., Patkos, T., and Candela, L.: Integrating heterogeneous and distributed information about marine species through a top level ontology, in: Metadata and Semantics Research, edited by Garoufallou, E. and Greenberg, J., p. 289–301, Springer International Publishing, https://doi.org/10.1007/978-3-319-03437-9_29, 2013.

van Gent, I., Rocca, G. L., and Veldhuis, L. L.: Composing MDAO symphonies: graph-based generation and manipulation of large multidisciplinary systems, in: Proceedings of the 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, https://doi.org/10.2514/6.2017-3663, 2017.

van Kuik, G. A. M., Peinke, J., Nijssen, R., Lekou, D., Mann, J., Sørensen, J. N., Ferreira, C., van Wingerden, J. W., Schlipf, D., Gebraad, P., Polinder, H., Abrahamsen, A., van Bussel, G. J. W., Sørensen, J. D., Tavner, P., Bottasso, C. L., Muskulus, M., Matha, D., Lindeboom, H. J., Degraer, S., Kramer, O., Lehnhoff, S., Sonnenschein, M., Sørensen, P. E., Künneke, R. W., Morthorst, P. E., and Skytte, K.: Long-term research challenges in wind energy a research agenda by the European Academy of Wind Energy, Wind Energy Science, 1, 1–39, https://doi.org/10.5194/wes-1-1-2016, 2016.

Vrandečić, D. and Krötzsch, M.: Wikidata: A Free Collaborative Knowledge Base, Communications of the ACM, 57, 78–85, https://doi.org/10.1145/2629489, 2014.

Zaaijer, M. B.: Great expectations for offshore wind turbines, Ph.D. thesis, TU Delft, https://doi.org/10.4233/uuid:fd689ba2-3c5f-4e7c-9ccd-55ddbf1679bd, 2013.