A comparison of two ontologies for agent-based modelling of energy systems

James Keirstead*
C410 Roderic Hill Building
Dept of Chemical Engineering
Imperial College London
SW7 2AZ
London, United Kingdom
j.keirstead@imperial.ac.uk

Koen H. van Dam†
Delft University of Technology
Faculty of Technology, Policy and Management
P.O. Box 5015
2600 GA Delft
the Netherlands
k.h.vandam@tudelft.nl

ABSTRACT

Conceptualisations formalised in ontologies are useful to provide an interface between people (e.g. between modellers and stakeholders), people and computers (e.g. data entry), and to ensure interoperability between software elements (e.g. communication between agents). As such, ontologies are useful for modelling purposes: with a formal definition of concepts, no misunderstanding about the intended meaning is possible. In this paper two ontologies designed independently for modelling applications in energy systems are discussed. The first ontology is designed for socio-technical infrastructure systems and it has been applied in a wide range of domains, while the second was developed with a focus on urban energy systems in particular. The different motivations for the development of these ontologies are addressed, one comparable key class is examined and lessons learned from these developments are presented.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation, Formalisms and Methods—*ontologies*

General Terms

Standardization

Keywords

Energy systems, ontology, interoperability, agent-based modelling

1. INTRODUCTION

Energy systems are an integral part of modern societies, powering economic activity, transportation, building climate control, lighting and many other applications. Yet their pervasiveness means that they are also extremely complex and difficult to model. Disciplinary perspectives on energy systems range from macro-economic and policy issues (e.g. [13]) to the psychology of domestic consumption (e.g. [1]) or the engineering specification of technologies and networks (e.g. [5]).

As a result, energy systems models are typically developed for a single application with a specific question in mind. This enables the boundaries of the modelling exercise to be clearly defined, reducing uncertainties that might otherwise impede progress. However, limiting the scope of a model in this way creates two problems. First, the model may fail to capture important cross-disciplinary interactions within the energy system or, at the very least, fail to explicitly identify where such links have been omitted. Second, specific models cannot be easily transferred to other contexts (e.g. from electricity to gas), creating significant amounts of repeat work. Developing tools that balance the needs of context-specific analyses with interoperability is therefore a research priority.

Ontologies are formalised conceptualisations [6], i.e. common models of data and concepts in a field of practice. They are an increasingly popular way of improving the interoperability of software models. By clarifying the definitions of major concepts in a field, it is possible for multiple modellers and models to reuse common components and to have a mutual understanding of available data sets. These techniques and technologies therefore suggest themselves as valuable tools for the modelling of energy systems where the meanings of modelled objects may have multiple meanings, depending on the modelling discipline and technology.

The idea of using an ontology to describe energy systems is not a new one. For example, Borst et al. [2] tried to define an ontology for the physical behaviour of "connected" systems that "are able to exchange energy", following the ideas of General Systems Theory. Others study the use of ontologies in linking energy systems and the built-up area in cities, e.g. [16] who note that "without the shared perception it would not be possible to develop adequate design methodology or approaches for design support that are systematic, consistent, reusable and interoperable". We have, however, not found evidence for a widely-used ontology that can be applied to describe different energy transformation technologies at different scales (e.g. a domestic boiler and a nuclear power plant) and their use in infrastructure networks.

This paper examines two related but independently-developed ontologies that have been applied to agent-based modelling of energy systems. After first introducing the two approaches in Section 2, we consider the differences in the specification

*Dr. Keirstead is Research Fellow and Team Leader with the BP Urban Energy Systems project at Imperial College London.

†Dr. van Dam is researcher at the Energy and Industry group of the Faculty of Technology, Policy and Management of Delft University of Technology.

of a single representative class within each ontology in Section 3. The concluding discussion in Section 4 examines the degree of overlap between the specifications and highlights the potential for developing a common ontology for energy systems modelling.

2. TWO ENERGY SYSTEM ONTOLOGIES

This section provides an overview of two ontologies for modelling energy systems, both created using the Protégé Frames software [11]. The ontologies were developed independently with different motivations and, of course, final structures.

2.1 An ontology for socio-technical systems

This section presents an ontology for socio-technical systems developed in the Energy and Industry section at Delft University of Technology. In this paper it is called the STS ontology.

2.1.1 Background and motivation

Challenges for the development of network infrastructure models – including electricity and gas grids, motorways or supply chains – arise when trying to incorporate both technical and social systems within one model. Existing tools deal with either the physical (e.g. models of industrial processes) or the social networks (e.g. economic market models), but these worlds have to be brought together in an integrated modelling approach for socio-technical systems [14].

To support the development of agent-based models of socio-technical systems, an ontology for this domain has been developed. The aim was to build an ontology not for one specific application domain (e.g. an electricity infrastructure), but to find commonalities between applications and therefore to develop a modelling framework that is able to deal with the reality of socio-technical network systems that are interconnected across sectors.

Modellers use the ontology to formalize domain knowledge, as language in the definition of behavioural rules and as communication protocols between agents. In this way, parts of the model can be re-used (e.g. re-using the model of a certain technology with a different agent, or re-using behavioural rules of one agent in another one, or even copying complete agents with their physical nodes into another model) and models of different infrastructures can be connected, even when they are developed by different modellers. In the framework proposed in [14], the STS ontology therefore acts as a cornerstone, providing both a user interface and a shared world model.

2.1.2 Structure

Since socio-technical systems, such as infrastructures, can be viewed as networks, the main concept in the ontology is that of a `Node`. Nodes are connected to one another by `Edge`s. The first distinction between `Node` classes is that of `SocialNode` and `PhysicalNode`, following the requirement that social and technical aspects of the system can be modelled independently of each other. In Figure 1 a small fraction of the STS ontology is presented. It shows, for example, that an `Agent` 'is a' `SocialNode` and that it 'has a' `Technology`. All nodes share slots (i.e. data fields) for describing, among others, `economicProperties` (i.e. properties related to the economics of the node) and `physicalProperties` (i.e. properties related to the physical aspects of a node).

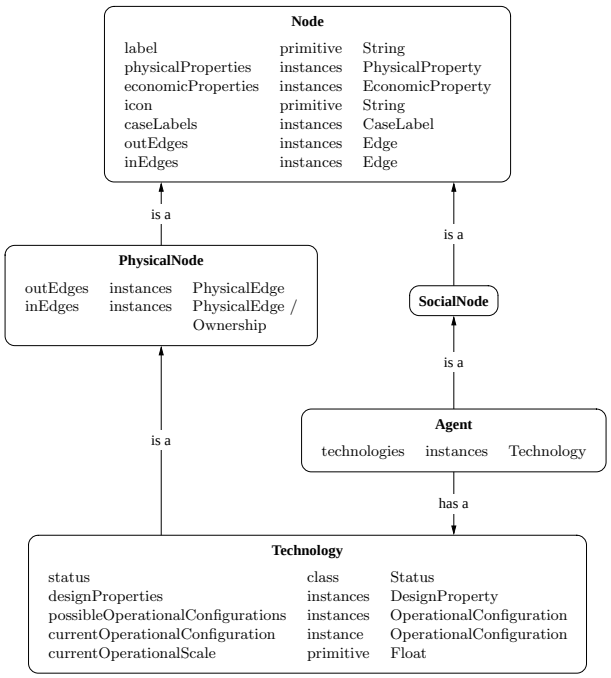


Figure 1: A fragment of the ontology for socio-technical systems, showing the relationship between different classes of `Nodes` (Social and Physical) and some of their slots. `Agents` and `Technologies` together form the socio-technical network.

The key classes that make up the socio-technical network are introduced below:

SocialNode A `SocialNode` is a `Node` capable of making decisions about `PhysicalNode`s. Subclass `Agent` represents an actor in the system. This can be a single person (e.g. an owner of a photovoltaic panel), a group of people (e.g. the operations department) or a whole organisation (e.g. the government). Moreover, `Agent` has one class-specific slot that distinguishes it from its super classes: a list of `Technologies` that the agent owns, controls, maintains, etc.

PhysicalNode A `PhysicalNode`, on the other hand, represents an element in the physical world, such as an engineered system. For `PhysicalNode`s, a process system perspective is followed: in the node a transformation takes place between inputs and outputs. A `PhysicalNode` can either be a small unit (e.g. a battery) or a very large system (e.g. a power plant). Subclass `Technology` will be discussed in more detail in Section 3.1.

Edge `SocialEdges` (e.g. contracts) and `PhysicalEdges` (e.g. a pipeline) between the nodes can be created, in which money and information flows occur between the social elements in the model, and mass and energy flows between the physical nodes. Social and technical networks can be formed, which interrelated become a socio-technical network.

Additionally, the ontology contains a large set of properties (e.g. maintenance costs, volume, maximum capacity)

that can be used to define characteristics of the system and a collection of units makes it possible to define the meaning of their values. Furthermore, the concept of a `GoodName` describes the "products" that exist in the system (e.g. crude oil or electricity). By using these in trade contracts or the specification of the required inputs of a certain process, it is guaranteed that communication is successful and that the same meaning is given to a concept. Finally, properties can be "attached" to `GoodName`s in flows or configurations to express, for example, a certain volume of natural gas with a specific pressure and temperature or an amount of wood pellets with a certain energy content.

2.1.3 Application

The STS ontology has been used in over 25 agent-based models developed by different modellers. These models have been applied to analyse the effects of carbon policies on the electricity infrastructure [3], to explore scenarios for the evolution of industrial clusters [10], for decision support on the location of an intermodal freight hub [12] and abnormal situation management in an oil refinery supply chain [15], to name just a few examples in different infrastructure domains.

Because the models share an ontology, specific building blocks are shared between the models. Agents from one such model can be added to a different domain and they are able to communicate about the same concepts, such as trading good names in certain amounts, transport contracts, and physical connections. Agents can be suppliers in one model (e.g. electricity producers) while in another model this agent is on the demand side (e.g. for natural gas). Furthermore, sharing an ontology between different modellers has enabled cooperation and mutual understanding of the mechanisms in the models of different infrastructure domains.

As an illustration of this collaborative process, over 300 different instances of `Technology` are defined in the shared knowledge base, each providing a rich and detailed definition of the inputs and outputs of the transformation process. Examples range from an oil refinery to a wind turbine, and from a fuel cell to a storage tank for diesel. For a large number of technologies, detailed property data is also included, for example, on maintenance costs, maximum capacity and construction time. These instances may be shared and re-used between different case studies. The knowledge base is read by the agent-based model to instantiate the agents, which means that the model can be changed (e.g. by adding new agents) without adjusting the source code.

2.2 SynCity ontology for urban energy systems

SynCity (short for "Synthetic City") is a modelling system for urban energy systems developed at Imperial College London, a key element of which is an ontology of relevant concepts.

2.2.1 Background and motivation

The goal of SynCity is to provide a platform for the modelling of generic urban energy systems. This concept facilitates the analysis of both real cities and abstract "toy" models without being overly burdened by the requirements of site-specific input data sets.

The platform consists for three major modelling components. First, a mixed-integer linear programming optimization (MILP) model is used to determine the optimal (e.g. low cost or low energy) layout of housing and activities within a city. This information is then used by the second model, an agent-based model which simulates the demands for energy resources in time and space as urban citizens go about their daily activities. The final model is a MILP optimization model that determines the optimal (as above) combination of energy supply and conversion technologies to meet the specified pattern of demand. Users interact with the system by using a Java library to build, configure and run model simulations.

Within this context, an ontology was introduced as a solution to two problems. First, consistent class definitions are needed to ensure that the results of one model can be seamlessly transferred to the next. While this function could be addressed purely with the use of Java object classes, the second problem – the storage and management of system components – creates an additional impetus for the use of an ontology.

2.2.2 Structure

Unlike the STS ontology, which is based on the unifying principle of a socio-technical system, the SynCity ontology is designed primarily as a library of domain-specific components. It therefore consists of a number of object classes that describe the main elements of an urban energy system, as well as specific instances of these classes. The main classes within the ontology are:

Resources Energy resources, such as electricity or natural gas, are described by a series of physical, economic, and model attributes. These include mass and energy densities, unit prices, or maximum stock values.

Infrastructures Infrastructures describe the physical structure of the city including, for example, buildings and networks. Each subclass has specific attributes (e.g. the average energy consumption of a building type), but all infrastructures share common slots such as capital and operational costs.

Processes Processes are technologies that convert one set of resources into another set. There are multiple subclasses to describe simple conversion technologies as well as more complex transportation and storage processes.

The ontology also features a number of minor related classes. For example, a `Property` is a generic attribute of an object as described by a value and a measurement unit. In turn, the `Unit` class and its instances were designed with the JScience library [4] in mind to facilitate easy unit conversion within SynCity.

2.2.3 Application

Relative to the capabilities of Protégé and other ontology applications, the SynCity ontology is essentially a glorified database. It does not take advantage of more advanced ontology features so, for example, there are currently no automated checks to ensure that certain attributes are specified in compatible units (e.g. all mass densities should be compatible with units of kilograms per cubic metre). Rather, the most valuable feature of the ontology is to provide an easy interface for new users to inspect the relationships between different object classes and to add their own object instances as needed for specific case studies. It also provides

the developers of the SynCity Java library with a canonical definition of how each class should behave.

The ontology has been used in the analysis of a proposed UK eco-town development [9], as well as to assess the viability of urban bio-energy supply strategies [8].

2.3 Comparison

Table 1 provides a basic comparison of the structure and application of the two ontologies. However the figures require some interpretation and so we explore the differences in more detail below.

2.3.1 Background and motivation

Each ontology was clearly designed with different goals in mind. The STS ontology has a wide scope and is intended to bring together the social and physical aspects of infrastructure systems. In contrast, the SynCity ontology was primarily designed to alleviate the difficulties of software integration, i.e. the use of different modelling technologies for the specific problem of urban energy systems analysis.

However there are common features between the ontologies. First, both ontologies were motivated by a desire to provide an explicit statement of domain knowledge. For expert users, i.e. programmers building models using the ontology components, this unambiguous statement of domain concepts is valuable in guiding the design of software behaviour, for example in an object-oriented paradigm. For non-expert users, the conceptual reference function helps to clarify the links between concepts within the resulting models. Furthermore, both groups were able to use the ontology as an effective method of data management. New object instances can be created, building up a valuable library of model components for re-use in multiple applications thanks to the consistent underlying structure of the data.

2.3.2 Structure

As highlighted above, the metrics presented in Table 1 provide only a cursory summary of the structure of each ontology. We now examine the differences in number of classes, slots and instances in more detail.

For the classes, the main difference between the STS and SynCity ontology can be accounted for by a tree of "labels" that is included in the STS ontology. This consists of no less than 140 different classes and makes up for almost half the total number of classes in the ontology. The hierarchical label structure can be used to "tag" anything from an `Agent` to specific descriptions of the inputs of a `Technology`. An example usage is tagging a CO_2 outflow of a powerplant as emissions, the input of the coal as a fossil fuel while it can be co-fired with different types of inputs labelled as biomass. When this label structure is omitted, the number of classes in both ontologies is more comparable.

The properties of each class are described by a number of "slots" (i.e. data fields) and there is a strong correlation between the number of classes and slots, as a higher number of classes often means that new slots have to be added to identify the difference with the other classes. However, further analysis is complicated by the use of slots in both ontologies to describe not only the core features of a domain concept, but also to conveniently store model-specific parameters.

A good example is the SynCity ontology's `Resource` class. Overall the class has twenty slots but of these, arguably only four are fundamental attributes of a `Resource` (namely its name, measurement units, energy and mass densities). The remaining slots describe model parameters related to a `Resource`, such as optimization limits on maximum and minimum rates of import or colours to be used for output graphics. From a user's perspective, it is convenient that these attributes are part of the `Resource` description (and hence on the same input form when creating a new instance). However, from a knowledge description viewpoint, these properties should be separate from the underlying core concept. Otherwise, users building a different type of model may be unable to reuse these core components.

Finally, the STS ontology has a much larger number of instances not only because it is used in more and in different case studies, but also because of the structure of the ontology. Each possible mode of operation of a `Technology` is an instance of a class, which requires instances for each `ComponentTuple` to connect a value with a unit. The SynCity structure has a sparser structure, consisting mainly of relatively discrete object instances.

2.3.3 Application

Both ontologies have been applied to a number of case studies. For the STS ontology, the scope has been broadest incorporating a range of systems including, but not limited to, energy systems. In contrast, the SynCity ontology has been limited to a select number of urban energy analyses. This again reflects both the design scope of the respective ontologies and also the number of researchers working within each project.

An informal review of these applications suggests that, practically speaking, the only major structural difference between the two models is the inclusion of social nodes within the STS ontology and the explicit definition of different types of connections. Even though concepts may be labelled differently in each ontology and other minor discrepancies exist that prevent a seamless merging of the two tools, the analysis suggests that a) the SynCity ontology could possibly be applied to other problem domains or, at the very least, b) significant common ground between the ontologies exists in the definition of the technical system and some type of unified ontology could be developed.

2.3.4 Conclusions

Overall then, the differences between the ontologies can be explained by three factors: the STS ontology covers a wider application domain than the urban energy system-specific SynCity ontology; the STS ontology has a larger user base and has been in use longer than SynCity; and there are some structural design differences. These issues demonstrate that the design of an ontology is as much art as science. However, by analysing such experiences, good practice guidelines can be established to ensure maximum compatibility and scalability of the resulting ontologies.

3. COMPARING A COMMON CLASS

We now turn our attention to a single object class in order to explore the differences between the two ontologies in more detail and to assess the feasibility of creating a common energy systems ontology.

A component of any energy system is a *process*, i.e. a device which converts one type of energy resource into another resource or a service. A simple example is a gas-fired domestic boiler. The boiler acts as a conversion process, taking in

Table 1: Basic comparison of the STS and SynCity ontologies.

	Socio-technical Systems	SynCity
Start date	January 2006	March 2008
Number of users	24	5
Number of classes	315	70
Number of slots	203	77
Number of instances	5532	551
Motivation	To facilitate re-use, sharing, interoperability and interconnectivity in agent-based models as well as data entry in a shared knowledge base	To link diverse modelling technologies (e.g. agent-based and mathematical programming models), to facilitate data entry and software design
Application	Socio-technical networks in general	Urban energy systems only
Implementation	Protégé Frames	Protégé Frames

natural gas and providing hot water and waste heat as outputs. However, there is significant scope for variation in the ontology specification: are capital costs annualized before entering the data, or do users provide equipment lifetimes and discount rates? How are the capacity limits and performance characteristics of each technology represented? Are there any simplifying assumptions made in the specification and if so, where and why?

After discussing each ontology's representation of the process class, a gas-fired domestic boiler will be conceptualised as an instance in the STS and SynCity ontologies.

3.1 The STS Ontology: `Technology`

In the STS ontology a `Technology` is a subclass of `PhysicalNode` and it represents a physical unit in which a transformation takes place.

3.1.1 Class context

Technologies are owned or controlled by an `Agent`, who makes decisions about how to use the technology. The properties defined for a technology constrain the choices the agents can make. Furthermore, agents may decide to buy additional technologies, or dismantle ones that are not profitable anymore, for example.

A `Technology` follows the input-output paradigm and is a black box itself. It can use different recipes, that each may have different input-output pairs, to reflect different modes of operation. Properties define, for example, the capacity of the technology to produce a certain product, the maintenance and operational costs attached to its operation and so on. Since the `Technology` is not an active unit itself, it has to be operated by an `Agent` who signs contracts for the transaction of mass or energy with other agents. The agent also "runs" the technology when it is producing, and has to choose which mode of operation and throughput to select.

3.1.2 Class description

Table 2 shows an overview of the core slots defined for the `Technology` class. The `possibleOperationalConfigurations` slot is list of possible configurations for the operation of this `Technology`, formalised as an `OperationalConfiguration`, a concept which is explained below. The current choice for configuration for the operation is set as the `currentOperationalConfiguration` and the `currentOperationalScale` is the scale of the operation (i.e. the throughput at which the unit currently operates). The relative numbers from the `OperationalConfiguration` are multiplied by this float. Finally, the `status` indicates if this particular system is under construction, under maintenance or operational.

Table 2: Slots of the STS class, `Technology`. Cardinality values: RS = required single, RM = required multiple.

Slot	Type	Cardinality
`possibleOpConfigs`	`OperationalConfig`	RS
`currentOpConfig`	`OperationalConfig`	RM
`currentOpScale`	`Float`	RS
`designProperties`	`DesignProperty`	RM
`physicalProperties`	`PhysicalProperty`	RM
`economicProperties`	`EconomicProperty`	RM
`status`	`Status`	RS

Table 3: Slots of the STS class, `ComponentTuple`.

Slot	Type	Cardinality
`goodName`	`GoodName`	RS
`relativeAmount`	`Float`	RM
`unit`	`Unit`	RS
`physicalProperties`	`PhysicalProperty`	RM
`functionLabels`	`FunctionLabel`	RM

An `OperationalConfiguration` is the specification of the connection between input and output of a technical system, precisely defining which `GoodNames` are at the input (`OperationalInputs`) and how they are transformed to the output (`OperationalOutputs`). For each of these inputs or outputs, the following slots can be defined in the form of a `ComponentTuple`, as shown in Table 3:

goodName The name of the product (from class `GoodName`) used at the input or output.

relativeAmount The amount of this product needed in relation to the other inputs, or the amount of this product produced in relation to the other outputs.

unit The unit in which the `relativeAmount` is expressed.

physicalProperties Other properties of the input, such as required pressure or temperature.

functionLabels Labels to, for example, indicate if the input is a scarce resource or a ubiquity, or if the output

Table 4: Slots of the SynCity abstract class, `Process`.

Slot	Type	Cardinality
`name`	String	RS
`var_name`	String	RS
`has_capital_cost`	`Property`	RS
`has_discount_rate`	`Property`	RS
`has_load_period`	`Property`	RS
`has_operating_intervals`	`ProcessInterval`	RM

Table 5: Slots of the SynCity class, `ProcessInterval`.

Slot	Type	Cardinality
`name`	String	RS
`has_unit`	`Unit`	RS
`has_operating_cost`	`Property`	RS
`has_min_capacity`	`Property`	RS
`has_max_capacity`	`Property`	RS
`has_inputs`	`ResourceFlow`	RM
`has_outputs`	`ResourceFlow`	RM

is a primary product or waste.

3.2 The SynCity Ontology: `Process`

The SynCity ontology uses the class `Process` to describe the conversion of resources.

3.2.1 Class context

The SynCity notion of a `Process` is also that of a black box which takes one set of resources as an input and gives another set as output. However within the optimisation models used by SynCity, there is a great detail of variety in how this basic concept is implemented and as a result, most of the detail is left to subclasses. For this discussion, we will focus only a `ConversionProcess`, that is a static process which converts resources. Other types implemented within the ontology include `StorageProcess`es (which specific different performance characteristics depending on whether a resource is being added to, held in, or removed from storage) and `TransportProcesses` (which includes both passenger and freight transport on a specified network).

3.2.2 Class description

At the highest level, the ontology has an abstract class called `Process`. Its slots are described in Table 4. Most of these slots are filled by instances of other classes, specifically instances of the `Property` class and its subclass. As mentioned above, each `Property` slot includes a name, value, unit, and data source (optional).

The other instance slot, `has_operating_intervals`, describes the performance characteristics of the technology. Designing the most generic specification of a process's performance is extremely difficult. Certain processes may have linear performance characteristics and therefore ratios of inputs and outputs can simply be scaled. Other processes however may be highly non-linear (for example, wind turbines where output rises with the cube of the wind speed). Since the data specified here is used with mixed-integer linear programming models, the `ProcessInterval` class describes the performance of a technology assuming a linear performance curve between specified minimum and maximum limits. However the `Process` class allows multiple `has_process_intervals` instances and so users can create piecewise linear approximations of non-linear performance curves.

Table 5 summarizes the specification of the `ProcessInterval` class. It shows that operating costs are assumed to apply only to a given process interval. The `ResourceFlow` class is a sub-class of `Property` and describes a rate-based flow of a resource, for example cubic metres per hour of gas or kilowatts of electricity.

There are several additional slots to handle the unique characteristics of the `ConversionProcess` class. Since this class represents stationary processes, such as generating equipment, slots are provided to describe the footprint of the technology (e.g. in square metres, so that the optimization models can ensure that a specific technology can be assigned on a sufficiently large location) and the maximum number of technologies allowed within any given cell. While both of these parameters were introduced in response to our specific modelling application, the footprint characteristic could be seen as an immutable property of a `ConversionProcess` with meaning in other applications.

3.3 Building a gas boiler in both ontologies

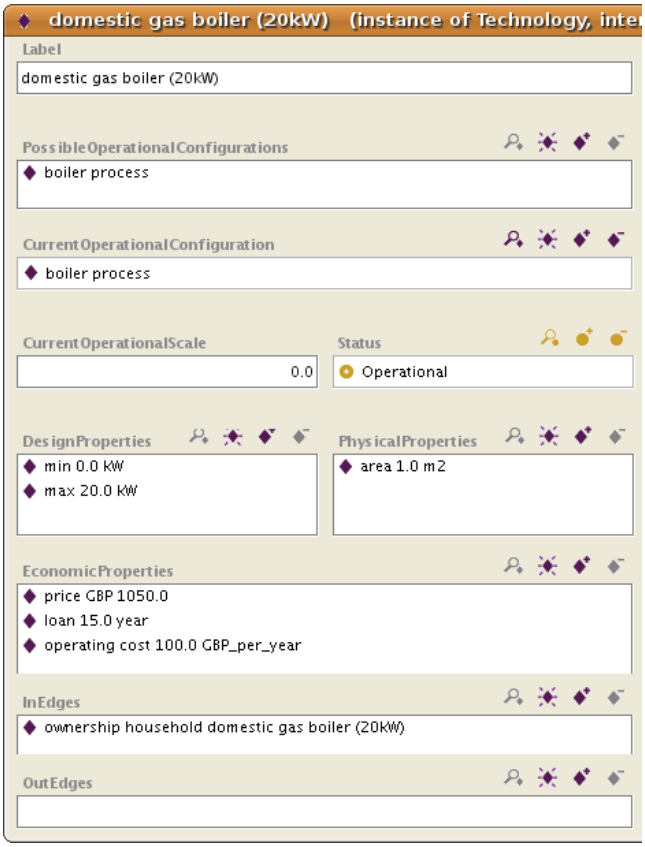
To facilitate a detailed comparison of each ontology, the specification of a typical gas-fired domestic boiler is outlined here. For this application no classes or slots have been added or modified in either of the ontologies.

Figures 2 and 3 show the Protégé forms used to describe the boiler in the STS and SynCity ontologies, respectively. Figures 2a and 3a illustrate the high-level `Technology` and `ConversionProcess` instance with common fields for properties such as name, capital cost, and so on, although there are slight differences. For example, the concept of a Footprint in SynCity is modelled in the STS ontology as an Area, although both are subclasses of their respective `PhysicalProperty` classes. Note that many of the fields in both ontologies contain explicit unit declarations.

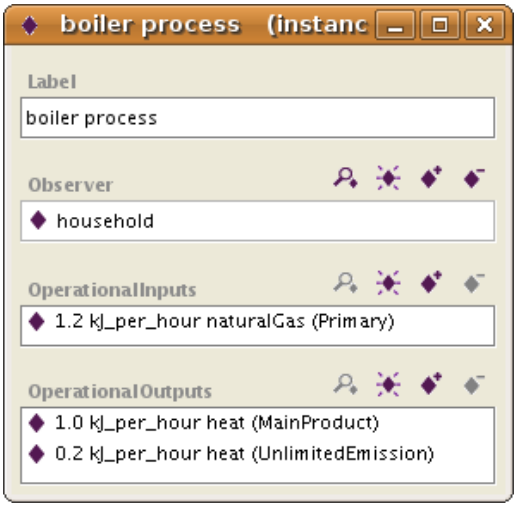
Details of the boiler's operational performance are shown in Figures 2b and 3b. In both cases, there is only one operational configuration (or interval) as it is assumed that the boiler's performance (i.e. the ratio of its inputs to outputs and its costs) is constant over the full 0–20 kW capacity range and it cannot use alternative modes with, for example, different fuels.

The general inputs/outputs structure of each ontology is quite similar, although there are discrepancies in some fields. A notable example is that the STS ontology places the operational limits on the technology as a whole, whereas in the SynCity ontology, this is a feature of the specific operational configuration. More generally, the `has_process_intervals` slot has no equivalent in the STS ontology (which could be an obstacle in defining more realistic curves between inputs and outputs), while the SynCity ontology does not explicitly contain the current status, operational mode and throughput (which limits expressive power on systems with alternative inputs or those that are under construction, for example). It must be stressed that such concepts could easily be introduced when needed.

The figures highlight some of the issues mentioned above. For example, §2.3.2 noted that it can sometimes be difficult to separate intrinsic properties of an ontology class from properties that may be required for specific modelling appli-

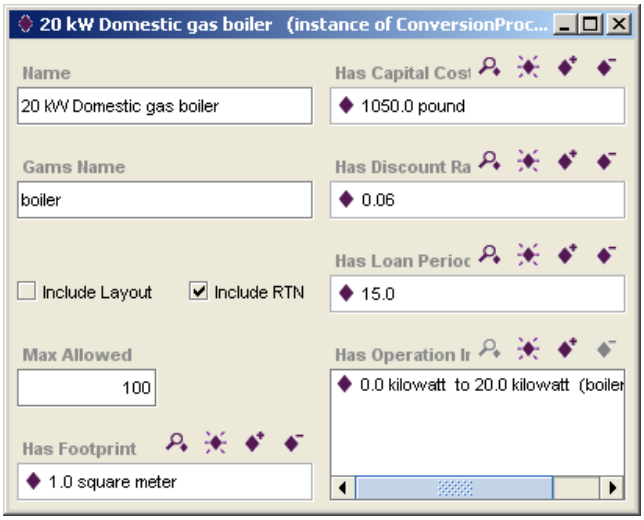


(a) Form for the Technology instance.

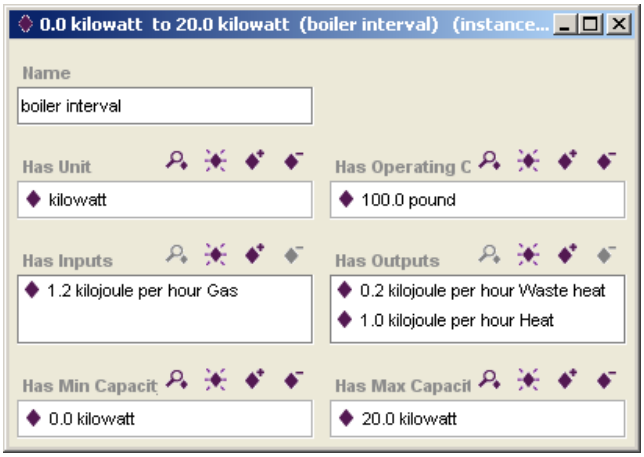


(b) Form for the OperationalConfiguration.

Figure 2: Screenshots of the STS ontology specification for a domestic gas boiler.



(a) Form for the ConversionProcess instance.



(b) Form for the ProcessInterval.

Figure 3: Screenshots of the SynCity ontology specification for a domestic gas boiler.

cations. The `include_layout`, `include_RTN` and `max_allowed` slots are good examples of these secondary characteristics as they control the inclusion of the given `ConversionProcess` within a specific SynCity sub-model.

More importantly, this case confirms that main difference between the two ontologies is that the STS ontology contains the concept of an `Agent`, representing an actor in the system (e.g. the Observer of the boiler process). This difference is not in any way an obstacle to the definition of concepts within the scope for which SynCity was developed – SynCity's agents are described exclusively in the Java library's agent-based model component – but it does mean that the specific socio-technical relationships as captured in the STS ontology cannot be expressed with it (such as ownership of a technology or contracts between agents for goods and services).

4. CONCLUSIONS

Two ontologies for modelling energy systems have been presented and compared. The analysis demonstrated that the intrinsic properties of the technical elements of an energy system could successfully be expressed in both ontologies discussed here. Even though they were developed independently from each other and for a different purpose, there are some clear similarities in the choices made in the design of the ontology. This suggests that it may be possible to establish a standard ontology for energy systems.

Ontologies are a proven tool for energy systems modelling: they enable modellers to work together and share building blocks as well as data sets. Especially when dealing with energy systems, where different disciplines meet each other, such a shared formal language is indispensable. Ontologies facilitate consistent software design and interoperability between models, even when they have been designed by different modellers, with different modelling techniques and in different domains. This is important for agent-based modelling in particular, because of its decentralised paradigm and the need for a communication language between the agents. A disadvantage is that using a fixed structure may be limiting the freedom of a modeller, but we believe this is outweighed by these advantages and the time saved not having to develop a new data structure if a suitable ontology is available. Furthermore, in energy systems by definition multiple infrastructures are integrated and capturing this in a single model requires the interface provided by a shared ontology.

If a common ontological structure is therefore possible, it is important to identify what characteristics it should have. We would suggest that there are two driving issues for good practice. The first is compatibility: the ontology should be usable in a wide range of applications. This means that class structures should make extensive use of abstraction: good examples of this include generic `Property` classes of both the STS and SynCity ontologies. In contrast, SynCity's mix of core concept characteristics and model representations may be convenient within a specific application domain but is ultimately limiting in the long term.

Abstraction is also vital to the second characteristic: scalability. As shown in Table 1, ontologies can become very large as a user base begins to deploy the tool to various applications. If the ontology is not designed from the outset with clear abstractions, then there is a significant risk that new applications will require changes to the ontology that may "break" other models.

A final point is that a common energy systems ontology (or even a more generic framework, such as the STS system) should encourage community interaction. This has two aspects. First, existing tools should be used where possible; for example, SynCity's use of the JScience unit framework reduces development effort and, where such external tools are well-designed, avoids the introduction of poorly conceived model structures. Secondly, it is desirable that such a tool is open to the community. Existing modelling libraries are often proprietary, either in content or format (e.g. [7] and some instances in the STS ontology), and this restricts their use in modelling and hinders interoperability. Openness also enables learning from the choices made by others.

Further research will attempt to swap ontologies, designing models for SynCity using the STS ontology and vice versa, and study how the ontologies can be linked or unified. Following this exercise, we would be interested in establishing an open community effort to build a standardised modelling ontology.

5. ACKNOWLEDGMENTS

James Keirstead would like to acknowledge the support of BP through the BP Urban Energy Systems project at Imperial College London. Koen van Dam would like to acknowledge the Next Generation Infrastructures Foundation for their financial support.

6. REFERENCES

[1] W. Abrahamse and L. Steg. How do socio-demographic and psychological factors relate to households' direct and indirect energy use and savings? *Journal of Economic Psychology*, 30(5):711–720, October 2009.

[2] R. Borst, J. Akkermans, A. Pos, and J. Top. The PhysSys ontology for physical systems. In *Proceedings Workshop Qualitative Reasoning '95*, Amsterdam, 1995.

[3] E. J. L. Chappin, G. P. J. Dijkema, and L. J. d. Vries. Carbon policies: do they deliver in the long run? In P. Sioshansi, editor, *Carbon Constrained: Future of Electricity*, Global Energy Policy and Economic Series, pgs. 31–56. Elsevier, 2009.

[4] J.-M. Dautelle. JScience, 2009.

[5] L. Girardin, M. Dubuis, N. Darbellay, F. Marechal, and D. Favrat. Energis: A geographical information based system for the evaluation of integrated energy conversion systems in urban areas. *Energy*, 35(2):830–840, 2010.

[6] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[7] IEA Energy Technology Systems Analysis Programme. MARKAL: Frequently asked questions, 2010.

[8] J. Keirstead, N. Samsatli, A. M. Pantaleo, and N. Shah. Evaluating integrated urban biomass strategies for a UK eco-town. In *Proceedings of the European Biomass Conference*, Hamburg, 2009.

[9] J. Keirstead, N. Samsatli, and N. Shah. SynCity: an integrated tool kit for urban energy systems modeling. In *Proceedings of the 5th Urban Research Symposium*, Marseille, 2009.

[10] I. Nikolic. *Co-Evolutionary Process For Modelling Large Scale Socio-Technical Systems Evolution*. PhD thesis, Delft University of Technology, 2009.

[11] SCBIR (Stanford Center For Biomedical Informatics Research). Protégé Frames, 2009.

[12] A. Sirikijpanichkul, K. H. van Dam, L. Ferreira, and Z. Lukszo. Optimizing the location of intermodal freight hubs: an overview of the agent based modelling approach. *Journal of Transportation Systems Engineering and Information Technology*, 7(4):71–81, August 2007.

[13] N. Strachan and R. Kannan. Final report on DTI-Defra scenarios and sensitivities using the UK Markal and Markal-macro energy systems models. Technical report, UK Energy Research Center, 2007.

[14] K. H. van Dam. *Capturing socio-technical systems with agent-based modelling*. PhD thesis, Delft University of Technology, Delft, 2009.

[15] K. H. van Dam, A. Adhitya, R. Srinivasan, and Z. Lukszo. Critical evaluation of paradigms for modelling integrated supply chains. *Journal of Computers and Chemical Engineering*, 33(10):1711–1726, October 2009.

[16] W. Zeiler, M. van Houten, G. Boxem, P. Savanovic, et al. Flexergy: an ontology to couple decentrilised comfort systems with centralised energy infrastructures. In *SASBE2009 proceedings, Smart and Sustainable Built Environments*, Delft, 2009.