ONTOLOGY APPROACH FOR BUILDING AND VISUALISING PROCESS SIMULATION MODELS USING 2D VECTOR GRAPHICS

Tuukka Lehtonen [*,1] **Tommi Karhela** [*,1]

[*] *Technical Research Centre of Finland, P.O.Box 1000, FIN-02044 VTT, tuukka.lehtonen@vtt.fi*

Abstract: Large-scale process simulation models are commonly created using 2D graphical user interfaces. Simulation data visualisation is typically done using textual monitors, symbol animations and colour coding. We propose an ontological approach for building and visualising 2D process models. An ontology-based approach using semantic networks for both of these tasks is presented. The key advantages of this approach are the unification of the data model (semantic graph), automation of modelling tasks by inference and synchronization between concepts of different ontologies via rule-based ontology mapping.

Keywords: Process Modelling, Process Simulation, Visualisation, Vector Graphics, Ontology

1. INTRODUCTION

Dynamic large-scale process simulation is used in industry for process and control design, control system testing and for operator training and support. Process modelling and simulation environments are usually separate software systems where simulation models are configured through graphical diagrams similar to so called flow diagrams or process and instrumentation diagrams. In addition simulation environments also visualise simulation data in these diagrams using textual monitors, symbol animations and colour coding. Software solutions for building such environments include several conceptual layers such as graphical concepts, process and instrumentation concepts, simulation solver configuration concepts and concepts of the mathematical solution layer. In this study a new approach for describing and mapping these layers is introduced. In this approach semantic technologies are used for ontology description and mapping. The emphasis is on an ontology for describing 2D vector graphics and on its mappings to other layers of large scale process simulation. The advantage of this approach is the flexibility of using different standards and approaches in different layers and changing them in a modelling environment that supports this approach.

First the ontologies related to this work are introduced in sections 2 and 3. In section 4 the method and ontology for ontology mapping is described. Section 5 describes our approach to ontological process modelling and section 6 deals with the simulation aspect respectively.

2. BASE ONTOLOGY

2.1 Semantic Graph

The World Wide Web Consortium (W3C) Semantic Web community promotes knowledge representation as a graph of *resources*. The semantic web is a directed graph of nodes and edges, where the nodes are typically called *resources*. The edges of

[1] http://www.vtt.fi

the graph describe various relationships between the resources and they correspond to some resource in the semantic graph.

The Resource Description Framework (RDF) is a standard for semantic information representation and data exchange in the web. RDF uses (subject, predicate, object) triples to represent graph data. Each triple defines a relation between two resources. Suggested approach uses similar triple-based model in lowest level data storage, versioning and retrieval framework.

2.2 Upper Ontologies — Layer Zero

To be able to describe actual knowledge higher level descriptions of concepts, i.e. ontologies, are required. Information is typically organized into domain specific ontologies which define themselves based on other ontologies. In a unified knowledge representation system the domain ontologies need to be based on some common *upper ontology*, which defines fundamental things, such as types and relations to be used in construction of higher level ontologies. Upper ontologies have been developed usually from many different viewpoints and agendas. W3C Web Ontology Language (OWL) is an example of a popular upper ontology. The scope of upper ontologies ranges from "all of human consensus reality" of the Cyc project (Cycorp, 1994) to "representation of oil and gas production facility life cycle" of ISO 15926.

Based on OWL and other such ontologies, a similar upper ontology called *Layer Zero*, has been developed. Its fundamental concepts are divided into *types* and *instances*. Instances are created from types by *instantiation* and the instances derive their *properties* from the types. All types can have properties, dubbed *qualifiers*, which are available to instances.

Base types of Layer Zero are *object types*, *property types* and *relation types*. Object types are used to define domain concepts, property types define the properties of domain concepts (object types) and finally relation types define the relations that can be used to semantically connect the domain concepts. Relation types correspond to the edges of the semantic graph and their domains and ranges are defined restrictively. Property types are used to contain primitive data or they can be composited.

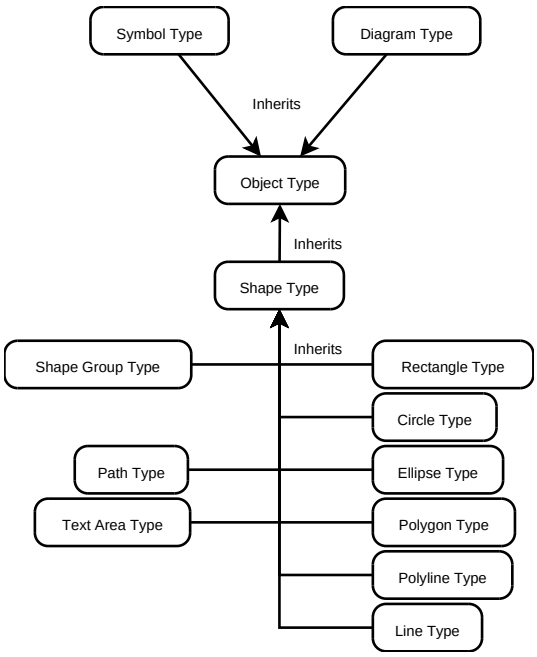

Fig. 1. Object type hierarchy of basic 2D shapes.

3. MODELLING AND SIMULATION ONTOLOGIES

3.1 2D Graphics Ontology

The 2D graphics ontology is defined on top of Layer Zero. It is a vector graphics representation based on the Scalable Vector Graphics (SVG) specification by W3C. SVG documents can be used to create animatable graphical objects where the animation can be bound to time or based on dynamic SVG documents via scripting with standard JavaScript.

This approach focuses on taking elements and *Modules* of the SVG specification (SVG 1.2 Tiny) related to the diagram methods of process modelling and simulation domain and constructing an ontology from them. At the moment only the bare necessities are mapped, but further mapping is possible if deemed necessary.

To describe basic graphical shapes and compose them the *SVG Shape Module* is needed to define general paths and basic shapes. An element for grouping is needed to create composite elements. To support textual elements part of the *SVG Text Module* is also needed.

Derived from these needs, the object types of the 2D graphics ontology and their inheritance hierarchy is depicted in Figure 1. These types form graphical representation part of the ontology. Each primitive is inherited from *Shape Type* to give each Shape Type some common or optional properties, such as an affine transformation or styling attributes.

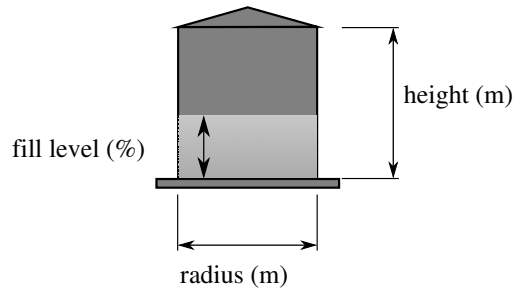

Fig. 2. Example parametrization of a Tank process component symbol. Here the *fill level* parameter could be changed during simulation to visually illustrate the fullness of the tank.

All of the corresponding SVG elements have attributes which are mapped directly to string-form property types of the object types in Figure 1. Examples of such attributes are the center point and radius of a circle or a path outline definition string. How these primitives are rendered depends on their styling attributes which again are mapped to optional properties of each primitive element. Stroke and fill related attributes are the main styling attributes for basic shapes. These attributes completely determine the actual appearance. Configurability of visual appearance is a high priority. It is needed to support the creation of useful and illustrative user interfaces for different domains.

Using these shape type definitions graphical objects without any meaning can be created by instantiating them in a hierarchy in the same fashion as in SVG.

In order to support real-time visualisation of e.g. simulation results a mechanism for modifying the appearance and shape of the different graphical elements at runtime is needed. This can be achieved by either modifying the general affine transformation of a shape or modifying the type specific parameters of a shape such as the definition of a general path or the contents of a text area. Modifying the transformation gives limited possibilities for animation but the possibilities of modifying any parameter of the graphical elements are limited only by performance issues.

There are several ways to perform the modification of the properties of graphical objects. This approach relies on parametrization of graphical shapes and mapping of the parameters to the primitive shapes of a particular shape composition. Such a parametrized shape is called a *Symbol Type*. Symbol Types may have many graphical representations, i.e. hierarchies of Shape Type instances connected to them. One of the purposes of this type is to give a meaning to the graphical presentation, for instance by creating a symbol for a process component such as a tank parametrized as shown in Figure 2.

Finally to support building graphical representations of process models a *Diagram Type* object is introduced to aggregate the actual model layout. Diagrams are made up of symbol instances where the symbols may be e.g. standard process components, textual monitors or visual monitors such as different kinds of trends.

3.2 Pipeline Structure Ontology

The pipeline structure ontology has been defined for describing the Process and Instrumentation (P&I) model data content and on the other hand for exchanging information from the functional design to the detailed 3D design. It is formed on the basis of a standard draft proposed by the Finnish process industry standardisation centre (PSK). It can be used to define pipelines, branches and pipeline components and connect these together.

The ontology describes concepts to define *plant models* that consist of *plant objects*. Models are made out of *pipelines* consisting of *branches* which in turn consist of different *pipeline components*. Pipelines and branches connect process equipments such as pump and tanks together. All the defined plant objects are considered *functions* since they fulfil some functional purpose in a process, thus defining the process in an abstract manner. An example of such a function is pumping. Functions in the pipeline generally describe operating, dimensioning, and testing values as properties which are normally defined by the person modelling the process.

The concept of a *product* is introduced to describe actual products that are chosen to fulfil functions in the pipeline such as a *pump X by manufacturer Y*. Product specifications give the properties of each product their individual nominal and boundary operating values which affect the choice of a product for a function.

In order to support plant life cycle management a concept of a product *individual* is also included in the ontology. Individuals are instances of a given product that in turn have their own individual operating values.

3.3 Companion Model Ontology

The companion model approach suggested by Juslin (Juslin, 2005) describes a generic framework for modelling and simulation of industrial processes. The companion model ontology created from this framework defines a *simulation flow network* consisting of *nodes* and edges called *branches*. It is used here merely to emphasise the

differences between process modelling and simulation level concepts.

4. ONTOLOGY MAPPINGS

To visualise the results of a simulation with the three specified ontologies, the simulation model values need to be carried to symbol parameters in a configurable way. Vice versa to configure the simulation model in the first place we need to be able to construct the simulation model graphically from the modelled pipeline. Also any changes made directly to a pipeline data model should propagate to both the graphics model and the simulation model.

Thus a method for describing similarities between the ontologies and propagating changes in one ontology to another with an arbitrary mapping is needed. This makes it possible to keep the different models synchronized.

To perform the needed functions, multiple kinds of mappings are needed. The simplest mappings merely synchronize property values between objects of different ontologies where as more complex mappings even generate new or remove old objects based on changes in the models. Logical inference has been previously applied for similar problems in ontological 3D visualisation (Kalogerakis *et al.*, 2006).

The mapping ontology defines concepts for modelling *rules* which define arbitrary reactions to given states of the data model. These reactions can then be used to perform simple property mappings and even complex generative updates.

5. AN ONTOLOGICAL PROCESS MODELLING AND SIMULATION APPROACH

There are multiple ways of building process and simulation models. The most common way is to use a graphical user interface for visually drawing the model. Simulators are often separate software packages and thus modelling applications may need to be able to export their graphical models into formats desired by the simulator interfaces. To provide feedback the simulation results need to be retrieved and interpreted. Current modelling environments are usually tightly bound to a single graphical, structural and simulation model depending on the separation of models. It is also possible to directly create structural and simulation models, that do not contain any graphics information and feed them to a simulator. Yet this is often tedious and inconvenient and does not provide for illustrative and in context visualisation.

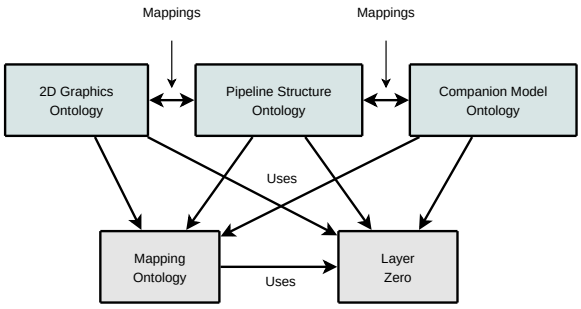

Fig. 3. The dependencies and relationships between the described ontologies.

The main idea in this approach is to create tools based on a semantic graph data model for graphically building graphical models of processes, e.g. P&I diagrams, and create mapping rules to take care of the models of related ontologies as automatically as possible. This means the rules must handle automatic generation of models of other ontologies and propagation of changes in one model to another where possible. Depending on the desired level of automation these rules may become quite complex. Figure 3 illustrates the case described in this paper.

To graphically build a model top-down, one could insert the main boundary components into the model such as tanks, draw pipelines or create other connections between them. Afterwards the connections or pipelines could be modified by inserting equipments such as pumps or valves in between. Another way is to start from a certain equipment and incrementally draw pipelines while at the same time appending equipments into them. In either case the pipeline is bound to be modified during modelling or simulation thus requiring the possibility of freely adding into and removing parts from the pipeline structure without the user having to start all over.

By requiring the user to insert every element into the graphical model that corresponds to some structural model element, i.e. a manual one-to-one mapping, creates unnecessary burden for the user. E.g. a pipeline specification may state that certain types of collars and flanges must be inserted between equipments and pipes. To ease the modelling burden these specifications can be utilized to automatically generate necessary components into the pipeline by creating rules to enforce these policies.

It is also possible for the user to have an existing structural model of a process, possibly in an interchange format exported by another modelling tool. In this case a graphical model could be generated from the structural information. It is possible that there is not enough information to generate an orderly graphical model, yet the generated version can serve as a good starting

point for further modelling instead of manually constructing the graphics.

While the process is being modelled, the simulation companion model is kept synchronized to the structural model by a set of mappings as depicted in Figure 3. It would also be possible to map graphics directly to the companion model ontology in order to graphically build companion models. Being very low-level, the model would be harder to understand than a graphical pipeline structure model.

One of the main advantages of this approach, is the freedom of association provided by the semantic graph data model. It allows for arbitrary relations to be created between any elements in the graph. Yet in order to take advantage of these relations a user interface must be aware of their semantics. This means the user interface is only as reusable as are the ontologies it is based on. On the other hand the semantic graph data model allows for unified data containment of any ontologies and models making it easier to create new or improve old user interfaces with new semantic features based on another set of relations. Also to add support for a new ontology in a tool the user interface need not necessarily be replaced completely or at all. E.g. if the pipeline structure ontology was to be replaced with an ISO 10303-221 Application Protocol (AP-221) based ontology, the graphical modelling user interface could be reused as long as appropriate mappings could be created between the graphics ontology and the companion model ontology.

Some possible general semantic relationships and their uses for queries in a 3D modelling context are outlined in (Kalogerakis *et al.*, 2006). Similar queries could be applied in this context.

The greatest challenges in this approach lie in making the use and definition of ontology mappings easy enough to be used by any commonplace user without huge amounts of domain knowledge. E.g. a modeller does not necessarily need to know whether his structural models are being represented using the PSK ontology or an AP-221 based ontology.

6. VISUALISATION OF PROCESS SIMULATION

Visualisation in this context refers to somehow taking advantage of the results of a simulation for showing them to the user in various ways to illustrate how the modelled process actually works. This is nothing new for current process modelling and simulation software.

To get simulation up and running, first a simulator is needed. An external simulator could be used but for better integration and also evaluation of the semantic graph platform a companion model solver based on the semantic graph has been developed. Next a companion model needs to be created. With suitable assumptions the companion model is close to a one-to-one or one-to-many mapping of the structural model into Nodes and Branches thus making automatic generation and propagating changes to and from the simulation model feasible.

As outlined in the previous sections the changes to the structural model need to propagate to the parameter properties of graphics model objects in order to visualise the current simulated state. The only thing remaining is to input new outlines for the graphical shapes that have changed to the rendering engine and re-render the model.

An obvious challenge arising from this approach is performance. Usually in large-scale dynamic process simulation the changes in the simulation results are not very rapid. The important thing for the observer is that the visualisation conveys the actual results, not that the visualisation runs in real-time. If the visualisation only conveys the current value of a property, we might end up dropping the value of some time step in the simulation. On the other hand if the property values are visualised as trends showing a history of simulation results, they can be updated less frequently without actually dropping results.

During simulation the propagation of changes happens mostly in one direction, i.e. the simulator to the graphics model. On the other hand changes will be propagated or made directly to the simulation model when the user wants to tweak model parameters. In order to have fast visualisation of simulation results, the propagation of simulation results needs to be very efficient. Since the semantic graph consists of versioned and persistent triples it is too heavy to be used to convey simulation results on every simulation time step. Therefore a separate shared memory channel called *Value Set* is used for the propagation of property value changes during simulation.

7. CONCLUSIONS

In this paper, we presented an ontological approach for building and visualising 2D process models using a semantic web based unified graph data model. First we overviewed the used upper ontology and the ontologies for representing process models in the graphical, structural and simulation domain. Next we proposed a general rule-based mapping approach to synchronize the contents of the models from different ontologies. This restricts the ontology coupling to the map-

ping descriptions. If appropriate mappings between ontologies can be formed model reuse and ontology changeability can be achieved. Finally our approach to graphical process modelling and simulation was described. The biggest challenges in this method lie in the complexity of ontology mapping. As the mappings grow in number and complexity, unseen performance and scalability issues are likely to surface.

REFERENCES

Cycorp (1994). Cyc. `http://www.cyc.com/`. [referenced 15.5.2006].

Juslin, K. (2005). *A Companion Model Approach to Modelling and Simulation of Industrial Processes*. VTT Publications. ISBN 951-38-6660-2.

Kalogerakis, E., S. Christodoulakis and N. Moumoutzis (2006). Coupling ontologies with graphics content for knowledge driven visualization. In: *Virtual Reality, 2006*. IEEE. pp. 43–50. 25-29 March 2006.