An Implementation of a Semantic, Web-Based Virtual Machine Laboratory Prototyping Environment

Jaakko Salonen[1], Ossi Nykänen[1], Pekka Ranta[1], Juha Nurmi[1], Matti Helminen[2], Markus Rokala[2], Tuija Palonen[2], Vänni Alarotu[2], Kari Koskinen[2], and Seppo Pohjolainen[1]

[1] Tampere University of Technology, Hypermedia Laboratory, 33101 Tampere, Finland
{jaakko.salonen,ossi.nykanen,pekka.a.ranta,juha.t.nurmi,seppo.pohjolainen}@tut.fi
[2] Tampere University of Technology, Department of Intelligent Hydraulics and Automation, 33101 Tampere, Finland
{matti.helminen,markus.rokala,tuija.palonen,vanni.alarotu,kari.koskinen}@tut.fi

Abstract. Creation of virtual machine laboratories – simulated planning and learning environments demonstrating function and structure of working machines – often involve a lot of manual labor. A notable source of the labor is the programming required due to changes in structural and functional models of a system. As a result, rapid prototyping of a virtual machine laboratory becomes difficult, if not impossible. We argue that by using a combination of semantic modeling and prototyping with a web-based system, more rapid development of virtual machine laboratories can be achieved. In this paper, we present the design and implementation of a semantic, web-based virtual machine laboratory prototyping environment. Application of the environment to a case example is also described and discussed.

Keywords: Semantic Web, Resource Description Framework, Web Ontology Language, Prototyping, Virtual Laboratory.

1 Introduction

Virtual machine laboratories (compare [1]) are simulated planning and learning environments demonstrating function and structure of working machines. The creation of such environments has traditionally involved a lot of manual labor. In essence, a developer needs to understand how the machine has been designed and what kinds of planning information and formats are used. In this task, information from various design documents such as CAD drawings and models need to be obtained and integrated. Especially in the context of working machines, it is often desirable to use mathematical modeling based real-time simulations for added interactivity and realistic behavior of the machine, adding a next level of challenge to the creation process.

Since these design documents are primarily created for purposes of the machine's implementation, they often provide less semantic information than what is required to

L. Aroyo et al. (Eds.): ISWC 2011, Part II, LNCS 7032, pp. 221–236, 2011.

create functional prototypes in the form of virtual machine laboratories. Especially semantic information connecting various design domains – such hydraulic and mechanical designs – is often informally or implicitly documented, since such pieces of information are more rarely needed for purposes of manufacturing.

Another aspect of this problem is that many perspectives of the data required for this task are largely missing in the designs. For instance, detailed system parameters required for simulation model generation are often missing. Similarly, while the functional model of the system is often implicitly understood by its designers, it may not be explicitly documented. As a consequence, the information required for the virtual prototypes, may need to be manually re-written by a virtual machine laboratory developer. In the worst case, the developer may need to re-engineer or re-design some parts of the system. As such, this process is potentially laborious, lengthy, rigid as well as prone to errors.

In Semogen research project (Phase I during 2010-2011), we have taken an alternative approach to the virtual machine laboratory generation in seek of a more rapid development model. Since most of the problems could be avoided by making sure the design data produced by the primary design activities is complete, we have defined a semantic process for tracking data requirements and the related information objects [2]. Instead of re-engineering and re-designing, we have focused on improving the machine-readability, i.e. the semantic quality of the primary design documents, by using semantic web technologies.

On a general level, this approach is not itself a novel idea; semantic web technologies have readily been applied to other and related domains. For instance in neuromedicine, an ontology for semantic web applications has been readily defined [3]. Benefits for applying various knowledge representation languages to systems and software engineering practices has also been outlined [4]. Very related to our work is a recent effort of product modeling using semantic web technologies [5], as well as a work towards describing linked datasets with an RDF Schema based vocabulary [6].

Also within our domain of application, the need for rapid prototyping and more semantic design data has been also recognized in other ventures. In Simantics project, an open cross-domain modeling and simulation platform was implemented [7]. With an Eclipse Platform -based infrastructure, various simulation and visualization plug-ins were integrated together, resulting in a toolkit for ontology based modeling and simulation. In an another project, TIKOSU, a data model, workflow and a prototype of database-based system was designed [8].

As according to the current machine and virtual laboratory design process, various aspects of the system are designed with a multitude of modeling applications and formats, both of which are often proprietary. Instead of changing these applications, our semantic process provides a model according to which information objects encoded to design documents can be tracked [2].

Our approach to extracting information from the design documents is based on the concept of adapters. For each individual modeling format and/or tool, a specific adapter software is written. This adapter accesses the raw design data and outputs a machine-readable presentation of this information. The information is then integrated together into a semantic model that comprehensively defines the modeled machine. Based on this integrated semantic model, various aspects of the virtual machine laboratory can then be generated.

Generating a virtual machine laboratory with our reference technology required us to generate application-specific configuration documents from the semantic model [9]. In an ideal case, a virtual machine laboratory generation could be approached as a configuration management problem. However, in practice creating a functioning prototype has previously required us to make changes to the underlying software itself. For instance, introducing a new attribute type to a component requires programming. As a result, even with perfected source data, the automated virtual machine laboratory generation could not be realized in many cases.

In order to support more rapid prototyping, a new environment was designed. The fundamental idea was to allow a developer to generate a more light-weight virtual machine laboratory, directly from a semantic model. Technologies were chosen so that they would support easily adding new features to the system with no programming labor whenever possible.

In this article, we present the design and implementation of this light-weight, virtual machine laboratory prototyping environment. The article is organized as follows: In chapter 2, use cases and user requirements for the environment are presented. Additionally a case example used during prototype creation is presented. In chapter 3 the design and the implementation of our prototyping environment is presented. In chapter 4 we present and discuss the results of applying our prototyping environment to the case example. Finally, in chapter 5, we conclude our work.

2 Use Cases, User Requirements and a Case Example

2.1 Use Cases and User Requirements

By definition, we consider *virtual machine laboratory (VML)* to be any environment that can be used to demonstrate a function and structure of a working machine. Different kinds of VMLs may be built for different purposes. Especially three use cases have been identified:

- **Design support.** VMLs to support designers (hydraulics, mechanics, etc.) and collaboration between design areas of the system, by providing a real-time simulation based functional prototype.
- **Educational use.** VMLs for educational purposes are used for providing understanding of the phenomena and function chains of mechatronic machine systems. In additional, educational VMLs may need to consider integration to virtual learning environments as well as assignment of various learning tasks.
- **User guide.** VMLs that provide interactive maintenance and spare parts guides

While details of the systems may change depending on its use case, they come to share many features. Several of our latest VMLs, including a harvester simulator for educational purposes have been developed using our M1 technology [9]. As such, we consider these prototypes and this technology as our reference and as a source for our user requirements. However, regardless of the use case in question, similar features are often requested.

Support for Rapid Prototyping. The ability to make changes to the underlying design and quickly apply these changes to the related virtual environment.

Dynamic Real-Time Simulation. Running real-time simulation of the system (or specific part of the system) in action, including interacting with the model (using controls).

Web Browser Based User Interface. Using the system running either locally or remotely with a web browser (device independence, effortless launch and use with software as a service).

Semantic Search. Locating resources, especially components based on given search criteria (for instance: "find components with a specific material" or "find all components of a given size range").

Ability to Use Data with Potentially Complex and Evolving Schemata. Since data from multiple design domains needs to be integrated, the underlying schema for a VML is potentially very complex. Additionally, since new design domains may be introduced, we should be able to handle some schema evolution with minimal additional work.

Support for Dynamic 2D and 3D Visualizations. The laboratory should be able to represent designs as 2- and 3-dimensional visualizations. In addition, dynamic simulation data is often desired for visualizing function in the designs.

Functional and Integrated Views. Views that representing how various design domains integrate together, for instance to form a chain of actions, are often requested. As machine designs are potentially very complex, provide functional representations a valuable abstraction.

Measurements. Ability to measure and analyze both static (size, width, height, diameter) and dynamic (position, velocity, pressure) properties of a system and its components. Especially for measuring dynamic properties, measurement tools recording history of changes is potentially very useful.

Support for Controllers and Hardware-in-the-Loop. In order to interact with the dynamic system model, at least some controller support is required. Minimally controller devices can be simulated with a graphical interface. Optionally various peripheral devices such as joysticks can be used to emulate controllers. For the scenarios requiring genuine hardware, the system should be able to provide support for hardware-in-the-loop (with other parts of the system being simulated).

Simulation Controls. While continuously playing real-time simulation is sufficient for many uses, it may be useful to be able to control the simulations. For this task, simulation controls including changing simulation speed as well as recording and playing back are potentially useful features.

Our reference technology, M1, already provides many of these features. However, it notably meets only partially the following requirements: 1) support for rapid prototyping, 2) semantic search, 3) web browser based user interface (currently only partial), 4) ability to use data with potentially complex and evolving schemata, 5) support functional and integrated views. Since developing a full-scale alternative to our current technology would require a lot work, it would make sense to aiming at creating a prototype specifically addressing these lacking features.

2.2 Case Example

In order to create a case example, we received real design information and expertise from the industry partners of the Semogen project.

In an attempt to understand and formalize how working machines are designed, we have analyzed the design process. As a result of this work, we have defined a model and methods for analyzing and designing semantic processes and manipulating design information through validation, transformations, and generation applications [10].

As according to this semantic process model, we have identified individual *design activities* and *information requirements* as dependencies between given activities. Each of the design activities may produce design data using *a designing application* often specific for the given design activity. Design applications may use any *data formats* as containers for the *design information*.

In an optimal solution, these data formats are based on open and standard specifications for which processing tools are readily available. Especially extensible markup language (XML) based formats are therefore favorable. As a second to best option, the designing application's export capabilities can be used to export the design data. As a last resort, the designing application's application programming interface (API) capabilities can be utilized in order to create an export format to capture design information.

A subset of the design data and process was chosen as a case example. As a scope of the example, we chose to study a single functionality ("boom lift") within the studied machine. By doing so enabled us to include a heterogeneous and a covering sample of various design activities and materials that potentially are linked by the case functionality in the level of design information. A listing of design activities, their outputs and designing applications used in the case example are provided in table 1.

Table 1. Design activities in the case example

Activity	Output(s)	Designing application(s)
Conceptual design	Conceptual and requirements design documentation.	PDF and Word documents
Hydraulic design	Hydraulic circuit diagrams (2D)	Vertex HD
Mechanical design	Mechanical models (3D)	Vertex G4, SolidWorks
Controller area network (CAN) design	Network and object models	Vector ProCANopen
Simulation design	Simulation models	Simulink

Also within the scope of the case example is some supplementary material. This includes some spare parts documentation and component information data sheets in Portable Document Format (PDF). It must be also noted that simulation models were not directly received from industrial partners, but instead were designed in-house.

3 Environment Design and Implementation

In this section, we will describe the design and implementation of our prototyping environment. Firstly, the semantic process and model are described. Secondly, we will describe details of implementation of the associated semantic viewer application.

3.1 Semantic Process

As described, our approach is based on empowering designers to use the tools they are familiar with. In order to support generation of machine-readable, semantic data, we formalize design activities into a semantic process that produces a semantic model that captures created design information.

In a high level overview, the design activities are formalized into a process. The design activities in our case example includes: concept design, hydraulic design, mechanical design, CANopen design, simulation design and system design. As an example of this process, let us consider the hydraulic design activity from our case example (Fig. 1).

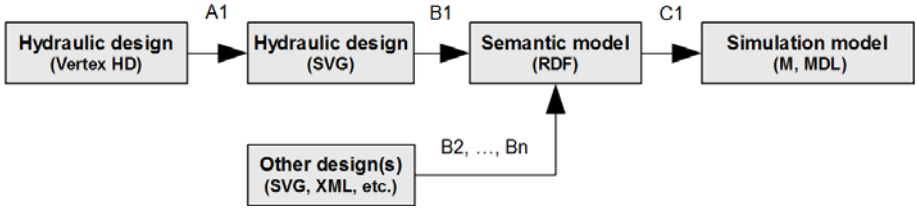

Fig. 1. Semantic process implementation example

Hydraulic diagrams were designed using a domain-specific application, Vertex HD. Following the design similar to data processing and visualization pipelines [11], we can now process the design data in various ways. In the first step (*A1*), the diagrams are then exported to an easily accessible format (SVG; Scalable Vector Graphics). Use of SVG enables us to further process the data with standard XML tools.

In the second step (*B1*), semantic data from the hydraulics SVG can then be extracted. In our case example we realized this by creating a custom XSL transformation. With the transformation, the semantic data encoded into the SVG file was captured and modeled in RDF [12] with a canonical XML serialization [13]. Use of a canonical serialization enabled us to process and validate the data with standard XML tools. For instance, rudimentary input data validators could be written with Schematron (http://www.schematron.com/). Note that as an input to this step, any SVG file containing required information may be used. Thus, other design applications producing conforming SVG can used in the making of hydraulics designs as well.

Similar to hydraulic design, other design activities may also produce input data (steps *B2, …, Bn*). The semantic model of a target system can then be simply formed by collecting together the RDF documents generated by the pipeline. RDF data model enables us to trivially aggregate these documents together to form a cross design-domain semantic model of the target system. By convention, we have chosen to store this aggregated instance data in a single document (`system.rdf`). For convenience, the definition of a related, domain ontology is passed along as well (`schema.rdf`).

Once the semantic model is generated as a result from running the data processing pipeline, it can be used to generate various portions of a VML. One important use

case for the semantic model is generation of simulation models and their templates (step *C1*). Simulation models are required in order to provide VMLs with real-time simulations that enable various dynamic visualizations. While details of simulation model generation are outside the scope of this paper, they are discussed in our other works ([10], [2]).

3.2 Semantic Model and Ontology

The semantic model forms the core of our prototyping environment. The semantic model integrates together design information from various activities. Very importantly, it provides us a standardized mechanism for not only creating machine-readable representations of design data inside the design domains, but also to connect these domains together. We strive at creating a comprehensive semantic model that could be used to generate any aspects of a VML as well as provide basis of integrating design data from many of the disciplines participant machine design. With a schema, we can also provide validation and integrity checks to the data.

Initially, the semantic model was created as an *ad hoc* aggregation of various RDF files from design data adapters. A semi-formal schema was written with SKOS (Simple Knowledge Organization System; [14]). This approach was sufficient for environment bootstrapping as well as for some rudimentary use cases such as using SKOS broader concept to interconnect design domains.

In order to effectively manage instance data, features of RDF Schema [15] were used. We added `rdfs:Class`, `rdfs:subClassOf`, `rdf:Property`, `rdfs:domain` and `rdfs:range` assertions to create classes and their hierarchies as well as properties associated to classes. Adding these assertions was important to provide the semantic model with rudimentary schema-based validation capabilities, including checking for valid properties and valid property values.

Our desire was to provide engineers with a graphical user interface for schema development. While modeling-wise RDFS would have provided us with most of the features required for schema writing, we could not find suitable tools for easily managing an RDFS-based schema. The most suitable software for this purpose was Protégé-OWL ontology editor (http://protege.stanford.edu/). Thus for mostly practical reasons, the original schema file was maintained in Web Ontology Language (OWL) format [16].

3.3 Environment

In terms of a practical implementation, we used a set of tools on Eclipse Platform [17] along with some specific conventions.

In order to simulate a “real-world” engineering process, our multidisciplinary team of researchers used the platform for collaboration through Subversion (http://subversion.tigris.org/). Since we run our environment as “stand-alone”, a version control system was necessary to simulate rudimentary product data management (PDM) system functionality (See e.g. [18]).

A new Eclipse project was used to represent an individual VML prototype. Individual design activities were modeled as folders. For each of the activities we further defined (information) requirements, and resources. The resources managed

outside the prototyping environment (original CAD drawings) were labeled as external, while resources generated due to data processing were separated under a folder labeled as generated.

We implemented data processing pipelines using Apache Ant (http://ant.apache.org/). Each individual processing step was designed as a new target in Ant. By specifying dependencies between these targets, a pipeline of activities could then be executed in the environment. In addition to XSL transformations that Eclipse supports out-of-the-box, we also configured the environment to enable adapter development with Python (http://www.python.org/).

The rest of the project was organized to following subsections (folders): 1) various tools and adapters required by the pipeline, 2) design application-specific libraries, 3) schema specification. We separated these items from the design activities since they all can be potentially shared between multiple projects.

3.4 Semantic Viewer Application

As a core component of the prototyping environment we implemented a semantic viewer application ("Semogen Player"). Key idea in the viewer application was to make it possible to directly use the semantic model, with no further data processing required for viewing and running a specified machine model.

After reviewing some suitable technologies, a stack consisting of open source components written in Python was chosen. The application architecture was based on the common Model-view-Controller pattern.

Web server was implemented with Tornado (http://www.tornadoweb.org/). A key requirement that lead us to choose Tornado was the need for real-time simulation support. Sufficiently low-latency (10-100 ms) for the simulation interface could be achieved with WebSocket protocol [19], a technology which Tornado was found to readily support.

As for model, we decided to use RDFLib 3 (http://code.google.com/p/rdflib/) along with SuRF (http://code.google.com/p/surfrdf/). With SuRF, RDF triples can be accessed as resources representing classes, properties and their instances. The library also provides various methods for locating and accessing these resources, including a SPARQL [20] interface. SuRF also enables us to support several different RDF triplestores including Sesame 2 (http://www.openrdf.org/) which may need to be used in larger data models. Thus, while we now chose to use RDFLib for practical reasons, other more efficient datastores could be used as well.

New functionality of the system was encapsulated in several Python modules. Framework-like features of the system were placed as part of Semolab model. These features included semantic model bindings, real-time simulation interface as well as application model and various utilities. The main module (*semoplayer*) was used for providing controllers to various views of the system. For view generation, we used Tornado's built-in template engines (HTML with embedded Python code blocks).

User interface components were designed with JavaScript. For interactivity and Ajax handling, jQuery (http://jquery.com/) was used. In order to create a more desktop-like user interface, we used jQuery UI (http://jqueryui.com/) and jQuery UI.Layout Plug-in (http://layout.jquery-dev.net/) as well as several other jQuery plugins. For 2D diagrams, SVG was used. 3D views were implemented using X3DOM (http://www.x3dom.org/).

4 Virtual Machine Laboratory Prototype

Based on the source design materials we received, the defined prototyping environment was used for adapting the source materials to the semantic model. As a result, various virtual machine laboratory views were generated from the design data. In this section we will describe how the semantic modeling and data processing was performed for various design materials, as well as how these materials were connected together. As examples of the design domains, we have included hydraulics and CANopen network designs. The described approach is similarly applied to other design domains such as mechanics, but for brevity are not covered in this paper.

4.1 Hydraulics

For the case example, a simplified hydraulic diagram of the boom lift functionality was drawn with Vertex HD. The diagram was carefully designed for machine-readability (Fig. 2) as follow: 1) for each hydraulic component, a title as well as model name was specified, 2) hydraulic ports were associated with individual components, 3) hydraulic pipes were connected semantically into various ports in the components.

Unique identifiers for hydraulic resources (components, ports and hydraulic pipes) were readily available in the exported design data. Only local uniqueness (per diagram) of the identifiers was guaranteed. In order to generate globally unique identifiers, a document specific prefix (`X`) was generated by combining a predefined project URI with the filename. Thus, for instance the full URI of the cylinder presented in figure 2, would resolve to `http://project-url/file/#comp-3_21`. In addition to URIs, the resources were identified with Dublin Core identifiers [21] containing local ID in textual format.

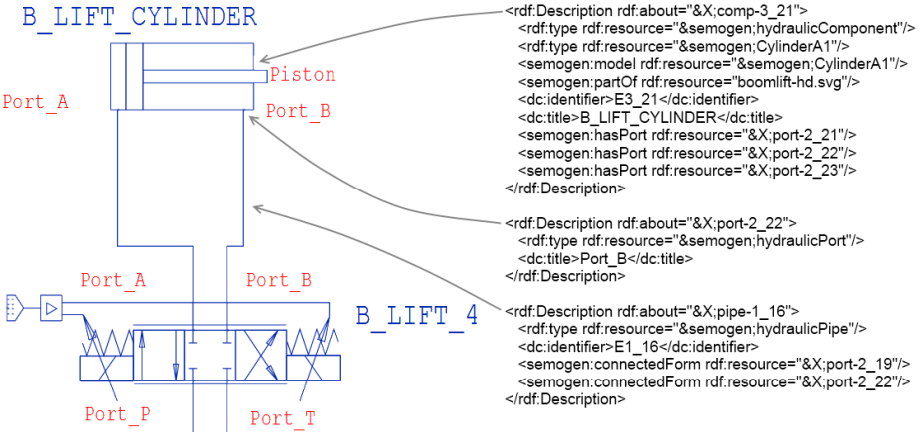

Fig. 2. Excerpt from hydraulic diagram and extracted RDF content

In terms of references, we also encoded model names of various hydraulic components into the design data. Any URI can be specified as a reference to a model name. The component model identifier specified in the source data resolves to two assertions in RDF 1) as an `rdf:type` assertion as well 2) as a `semogen:model` assertion. While model properties could be obtained via the type assertion, the latter was found more practical, since – due to type inference – a component may contain multiple type definitions, from which usually only one refers to a component's model.

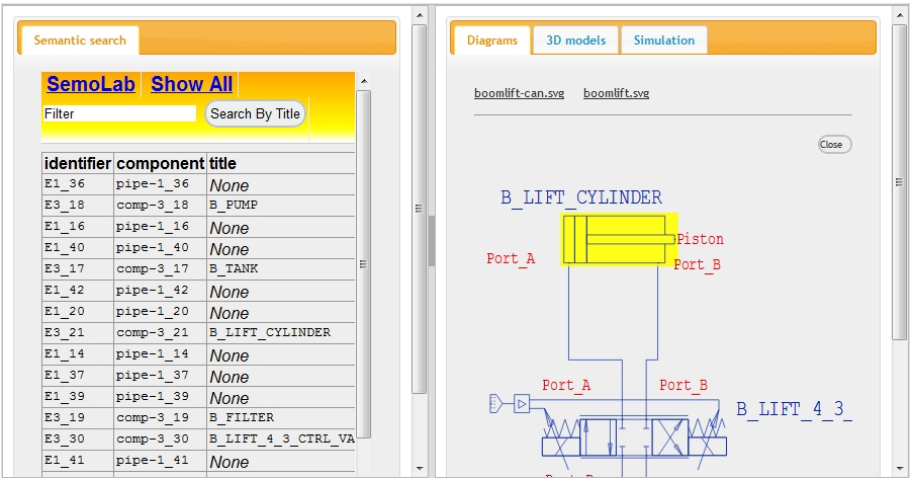

Fig. 3. Semogen Player user interface for hydraulic diagram with semantic search

Based on the exported SVG diagram and extracted RDF data, the data could be then viewed in our viewer application (Fig. 3). The viewer readily detects hydraulic diagrams from the input data (top-right corner). A semantic view (left side) can be used to list all components and pipes as well as their properties. Finally an interactive hydraulic diagram can be presented (right side). A component can be selected by hovering over it with a mouse providing visual linking between the search and the diagram visualization. In addition, any component can be clicked to display all the available RDF data for it.

4.2 CANopen Design

CAN design for the boom (lift) functionality consists of three different CAN buses. These buses were planned using ProCANopen software [22]. Each bus design includes all CAN nodes and their signal routing [23, 24]. This information was then converted from software's native output format (DCF) to an RDF model (Fig. 4). The designed RDF model includes buses, nodes, object and signals.

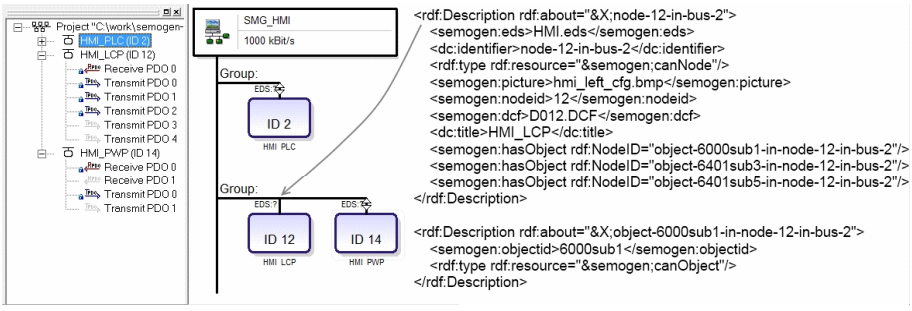

Fig. 4. A network in ProCANopen user interface annotated with related RDF data

In terms of identifiers, original design readily had unique identifiers for CAN buses. Further, each CAN node could be identified by its *NodeID* (`semogen:nodeid`) similarly made available in the original design. As a *NodeID* was only locally unique, a project specific prefix (`X`) was combined with *busID* and *nodeID* to create globally unique node identifier.As of nodes, each of them included object dictionaries, storing related data objects [23]. These objects are identified by a hexadecimal index number. CAN signals carries these object values to other nodes [23, 24]. So we generated these signals to RDF model and linked them to corresponding CAN objects.

The resulting RDF model was used to generate an SVG-based view of the CAN bus design (Fig. 5). Each component (node and bus) in the SVG contained an element which refers to the RDF model, so the user interface can be used to fetch more information and link to an RDF model on any component in the SVG diagram.

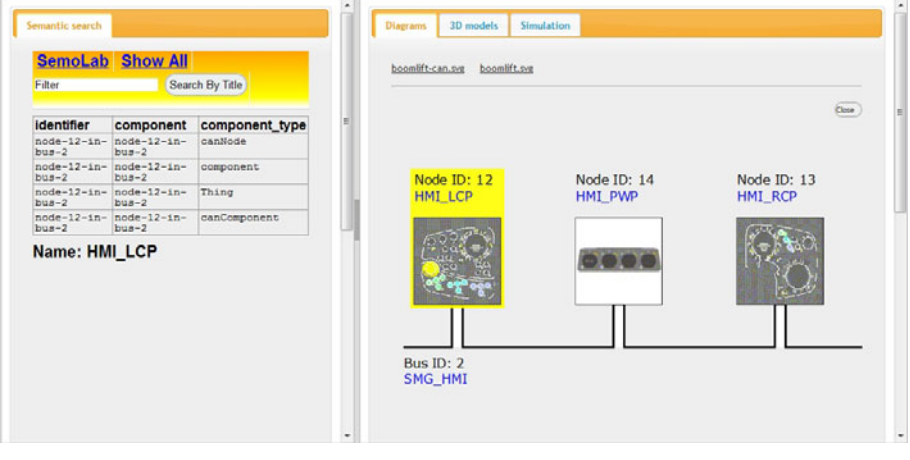

Fig. 5. CAN network visualization with component (node) selection

The RDF data was also used in generation of simulation model. Generator creates Simulink model with CAN nodes and buses. In the future, objects and signals can be used to parametrize these simulated nodes as well.

4.3 Connecting the Design Domains

While designs with various domains are typically created as "self-contained", there are several reasons why cross-references connecting the design domains may be needed. Firstly, an individual physical component such as *cylinder* may appear in multiple designs, potentially in different roles. Secondly, we may need to describe function chains – processes that describe how the various parts of the design work together to implement a functional – for which connections between various designs need to be provided. Finally, even additional cross-references may need to be provided especially for the generation of an integration simulation model.

In some of the current industry practices, engineers use arbitrary mapping tables, for instance implemented as Excel tables (or lists) of references. In the simplest case, these tables contain rows that provide mappings between domain-local identifiers as well as optionally define a global identifier. Additionally the mapping table may encode design information that either is not defined elsewhere or is scattered or otherwise hard to locate from the other design materials.

In order to provide similar mappings within our semantic model, a more formalized approach was needed: the cross-references need to be defined by using full identifiers (URIs). For a rudimentary approach to this mapping, we added an RDF file containing mapping assertions (`mappings.rdf`).

As an example, let us consider mapping various roles of a physical component, a hydraulic cylinder present in examples of sections 4.1 and 4.2. The cylinder appears both in hydraulic design diagram (`&X;comp-3_21`) as well as in CANopen design (`&X;node-1-in-bus-1`) resulting in two different URIs describing the same component. If a mapping assertion between these identifiers is to be defined as an RDF triple, its predicate depends on which relationship it describes. For instance, mapping from hydraulic component (role) into a CAN component (role), can be defined with specific predicate (`semogen:hasACANRole`).

Functions, abbreviations and other meta data that may be embedded into mapping tables can instead be described as new resources inside the mappings file. For instance, each new functionality of the machine can be created as an instance of `semogen:function` for which additional attributes can be defined. For each function, references to various components depending on their roles (for example `semogen:isControlledBy`) can be added as well.

4.4 Discussion

Rather than creating the data model by firstly designing a normalized schema, we created the environment from features rising from actual design documents and processes. In this design, an interactive process model was employed. During this process, various adapters, semantic models as well as features of the viewer application were iteratively developed.

In terms of VML features, our environment is already able to provide the rudimentary aspects of the required features. Firstly, it has proven to support rapid prototyping with the ability to use data with potentially complex and evolving schemata as somewhat demonstrated with the implementation examples. Secondly, the environment provides limited support for semantic search, as well as functional and integrated views to the data (mapping table view). Thirdly, while not demonstrated in this paper, rudimentary support for real-time simulations as well as measurement views have been implemented in the viewer, covering all of the most requested features for a virtual machine laboratory.

In terms of scaling our approach to full machine models and genuine engineering environments, some further work needs to be done. Especially data integration between various domains is seen as a challenge. In principle use of globally unique identifiers allows us to create cross-references between any designs. In practice, manual creation of cross-references is potentially cumbersome and thus impractical, requiring us to look into more automated solutions for scalability. For instance by providing a graphical user interface for the mapping generation, technical details of the RDF data model could be hidden from the system's designer. In addition, to reduce the manual work required for the mappings, the designer could be allowed to provide some general purpose rules for the mappings (for instance: “map together all hydraulic and CAN components that have identical titles”).

Another problem with our approach is that in order to integrate new design domains, new adapter needs to be written per new input data format. However, since our data model is based on use of standard RDF, a sophisticated designing application could overcome this problem by readily providing RDF export capabilities, thus leaving only the challenge integrating of the various data models.

Some important lessons were learned during the implementation process. Firstly, it seems that technically semantic web modeling and implementation tools are readily usable and mostly mature. What came as a partial surprise is the lack of well-defined, open vocabularies, schemata and ontologies for the modeling domains in question. For a wider application of semantic modeling, these definitions would be clearly needed. Finally, while some best practices and design patterns for semantic modeling especially in the domain of product modeling, have been defined [5], they are needed in a more wider deployment.

We applied our semantic modeling and process approach to the domain of virtual machine laboratory. As a direction of extension, we see that this process and the tools could be well generalized for other design and engineering practices as well. Especially a similar approach could be applied to other production modeling and engineering domains.

5 Conclusions

In this article, we presented a design and implementation of a virtual machine laboratory prototyping environment. The key approach in the presented environment was the use of understanding machine's and its virtual laboratory's design as a semantic process with defined information objects providing semantic links between various design activities. In the core of our approach was a semantic model

implemented in RDF, RDFS and OWL as well as a viewer application for a using conforming data.

In terms of our case example, the given technologies were found as a feasible way to model the underlying data. We see that this semantic modeling approach provided us with some fundamental benefits. Firstly, by using a design of adapters and integrating model, we were able to manage the design data in various domain-specific designing applications. Secondly, it enabled us to create a global, comprehensive representation of the all design data with references between various resources regardless of their design domain. By doing so, we were – in overall – able to understand and build formalized models of how various engineers understand the designs. Thirdly, we recognized that by using an RDFS/OWL based data model, we can fairly easily and quickly adapt the prototype to changes in a data schema. For instance, introducing new properties or classes can be done trivially, with no additional programming work required. Finally, we see that the use general-purpose semantic modeling maximizes data re-usability. For instance, simulation models as well as various aspects of a virtual machine laboratory, can both be generated from the same model.

A notable challenge in our approach is the management of the complexity arising from the heterogeneous process and modeling environment. As virtually any design tools or activities can be introduced to the semantic process, it can potentially become very complex. For instance, while implementation and use of various adapters for integrating data from the tools is often required, can exhaustive use of them result in unmanageable complexity in software design and maintenance. Similarly the design process may become overly complex, resulting in inefficiency in form of poorly manageable structures.

We see that the key in successfully applying this semantic process approach to machine design, requires identification and understanding of the concerns that cross-cut through various design activities. These concerns include, but are not limited to management of global identifiers as well as representation of functional, machine-level models. Without a systematic, process-oriented approach, the risk is that instead of integrating design information, we fall back to *ad hoc* methods of encoding the design data resulting in scattered, unconnected blocks of design data that hinders the re-usability of the data and scalability of the methods.

A lesson learned in the context of semantic web technology application is that while the technologies themselves can be considered mature, some known methods for efficient semantic modeling would have potentially proven to be valuable. We see further room for improvement for instance in introducing an efficient method for managing and integrating local and global identifiers. For industrial deployment of the approach, having more standard, domain vocabularies would be crucially important. Also even though best practices and patterns have already been somewhat extensively recognized and described, we see a room for further studies. Especially since several parts of the modeling technologies (update language, rule languages) are still under standardization and development, a potential of benefit exists from having definitions of more general purpose design patterns.

Individual engineering organizations may not find the motivation for developing domain vocabularies and ontologies. As such, the whole industry would likely benefit from utilizing a more open, collaborative process in the development of these artifacts

that are often a requirement for semantic model utilization. In our visions this work could lead to development of an open entity graph within the industry, similar to collaborative databases like Freebase (http://www.freebase.com/).

Within the working machine industry, we see that successfully applying integrated semantic modeling would open up entirely new possibilities for organizing design work. By introducing a semantic modeling-based generation of virtual machine laboratories for design, more agile research and development could be potentially realized. Especially a simulation-driven virtual prototyping process could lead to a new level of efficiency in machine design processes.

References

1. Grimaldia, D., Rapuanob, S.: Hardware and software to design virtual laboratory for education in instrumentation and measurement. Measurement 42(4), 485–493 (2009) ISSN 0263-2241
2. Nykänen, O., Salonen, J., Markkula, M., Ranta, P., Rokala, M., Helminen, M., Alarotu, V., Nurmi, J., Palonen, T., Koskinen, K., Pohjolainen, S.: What Do Information Reuse and Automated Processing Require in Engineering Design? Semantic Process. Journal of Industrial Engineering and Management (2011) (in Review)
3. W3C (2009) Semantic Web Applications in Neuromedicine (SWAN) Ontology. W3C Interest Group Note 20 (October 2009), http://www.w3.org/TR/hcls-swan/ (accessed August 26, 2011)
4. W3C (2006) Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. Editor's Draft (Fubruary 11, 2006), http://www.w3.org/2001/sw/BestPractices/SE/ODA/ (accessed August 26, 2011)
5. W3C (2009) Product Modelling using Semantic Web Technologies. W3C Incubator Group Report (October 8, 2009), http://www.w3.org/2005/Incubator/w3pm/XGR-w3pm-20091008/ (accessed August 28, 2011)
6. W3C (2011) Describing Linked Datasets with the VoID Vocabulary. W3C Interest Group Note (March 3, 2011), http://www.w3.org/TR/void/ (accessed August 28, 2011)
7. Simantics (2010) Simatics Platform - Details about Simatics platform, the software architecture, and its applications, http://www.simantics.org/simantics/about-simantics/simantics-platform (accessed June 23, 2011)
8. VTT (2011) TIKOSU - Tietokantakeskeinen koneenohjausjärjestelmän suunnittelu ja toteutus, http://www.hermia.fi/fima/tutkimus/tikosu/ (accessed June 23, 2011)
9. Helminen, M., Palonen, T., Rokala, M., Ranta, P., Mäkelä, T., Koskinen, T.K.: Virtual Machine Laboratory based on M1-technology. In: Proceedings of the Twelfth Scandinavian International Conference on Fluid Power, Tampere, Finland, May 18-20, vol. 1, pp. 321–334 (2011)
10. Markkula, M., Rokala, M., Palonen, T., Alarotu, V., Helminen, M., Koskinen, K.T., Ranta, P., Nykänen, O., Salonen, J.: Utilization of the Hydraulic Engineering Design Information for Semi-Automatic Simulation Model Generation. In: Proceedings of the Twelfth Scandinavian International Conference on Fluid Power, Tampere, Finland, May 18-20, vol. 3, pp. 443–457 (2011)

11. Nykänen, O., Mannio, M., Huhtamäki, J., Salonen, J.: A Socio-technical Framework for Visualising an Open Knowledge Space. In: Proceedings of the International IADIS WWW/Internet 2007 Conference, Vila Real, Portugal, October 5-8, pp. 137–144 (2007)
12. W3C (2011) Resource Description Framework (RDF), http://www.w3.org/RDF/ (accessed June 23, 2011)
13. Nykänen, O.: (2011) RDF in Canonical XML (RDF/cXML), http://wiki.tut.fi/Wille/RDFcXML (accessed June 26, 2011)
14. W3C (2011) SKOS Simple Knowledge Organization System – Home Page, http://www.w3.org/2004/02/skos/ (accessed June 23, 2011)
15. W3C (2004) RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation (February 10, 2004)
16. W3C Web Ontology Language (OWL), http://www.w3.org/2004/OWL/ (accessed June 23, 2011)
17. The Ecllipse Foundation (2011) Eclipse Platform, http://www.eclipse.org/platform/ (accessed June 23, 2011)
18. Bilgic, T., Rock, D.: Product Data Management Systems: State-Of-The-Art And The Future. In: Proceedings of DETC 1997, ASME Design Engineering Technical Conferences, Sacramento, California, September 14-17 (1997)
19. W3C (2011) The Websocket API. Editor's Draft (June 21, 2011), http://dev.w3.org/html5/websockets/ (accessed June 23, 2011)
20. W3C SPARQL (2008) SPARQL Query Language for RDF. W3C Recommendation (January 15, 2008)
21. Dublin Core Metadata Initiative (2010) DCMI Metadata Terms. DCMI Recommendation, http://dublincore.org/documents/dcmi-terms/ (accessed June 23, 2011)
22. Vector (2011) ProCANopen – Project Planning Tool for CANopen Networks, http://www.canopen-solutions.com/canopen_procanopen_en.html (accessed June 23, 2011)
23. IXXAT Automation GmbH (2011) Process data exchange with PDOs ("Process Data Objects"), http://www.canopensolutions.com/english/about_canopen/pdo.shtml (accessed June 23, 2011)
24. Vector (2011) CANopen Fundamentals – The CANopen Standard, http://www.canopen-solutions.com/canopen_fundamentals_en.html (accessed June 23, 2011)