Title	Open ontology-based integration platform for modeling and simulation in engineering
Author(s)	Karhela, Tommi; Niemistö, Hannu; Villberg, Antti
Citation	International Journal of Modeling, Simulation, and Scientific Computing (IJMSSC) vol. 3(2012):2, 36 pp.
Date	2012
URL	http://dx.doi.org/10.1142/S1793962312500043
Rights	Copyright © (2012) The Authors. Reprinted from International Journal of Modeling, Simulation, and Scientific Computing (IJMSSC). This article may be downloaded for personal use only

International Journal of Modeling, Simulation,
and Scientific Computing
Vol. 3, No. 2 (2012) 1250004 (36 pages)

DOI: 10.1142/S1793962312500043

World Scientific
www.worldscientific.com

OPEN ONTOLOGY-BASED INTEGRATION PLATFORM FOR MODELING AND SIMULATION IN ENGINEERING

TOMMI KARHELA

VTT Technical Research Centre of Finland
Vuorimiehentie 3, 02044 VTT, Finland
Tommi.Karhela@vtt.fi

ANTTI VILLBERG

Semantum, Tekniikantie 14
02150 Espoo, Finland
Antti.Villberg@semantum.fi

HANNU NIEMISTÖ

VTT Technical Research Centre of Finland
Vuorimiehentie 3, 02044 VTT, Finland
Hannu.Niemisto@vtt.fi

Received 6 April 2011
Revised 3 December 2011
Accepted 18 January 2012
Published 28 March 2012

The benefits of the use of modeling and simulation in engineering are acknowledged widely. It has proven its advantages e.g., in virtual prototyping i.e., simulation aided design and testing as well as in training and R&D. It is recognized to be a tool for modern decision making. However, there are still reasons that slow down the wider utilization of modeling and simulation in companies. Modeling and simulation tools are separate and are not an integrated part of the other engineering information management in the company networks. They do not integrate well enough into the used CAD, PLM/PDM and control systems. The co-use of the simulation tools themselves is poor and the whole modeling process is considered often to be too laborious. In this article we introduce an integration solution for modeling and simulation based on the semantic data modeling approach. Semantic data modeling and ontology mapping techniques have been used in database system integration, but the novelty of this work is in utilizing these techniques in the domain of modeling and simulation. The benefits and drawbacks of the chosen approach are discussed. Furthermore, we describe real industrial project cases where this new approach has been applied.

Keywords: Semantic data modeling; data driven approach; information management.

1. Introduction

Nowadays simulation appears increasingly as a self-reliant discipline in science and engineering. Conventionally, theoretical solutions developed for highly idealized systems have conventionally been confirmed by experiments and, subsequently, applied to industrial practice. This concept often leads to costly experiments in different scales before the desired operational practices and appropriate conditions are found. Computer simulation, unlike most theoretical methods, is related to real systems and provides an unprecedented access to true (industrial) conditions. In addition, simulation possesses few limitations with regards to operational conditions and parameters and both material and human resources, all of which can hamper experimental arrangements. Computer simulations have consequently often been credited with saving both time, labor and cost in a wide range of applications. The importance of modeling and simulation in both research and product development has been recognized worldwide. The National Science Foundation (NSF) report Simulation-Based Engineering Science (see Ref. 1) and the more recent update of it, International Assessment of Research and Development in Simulation-Based Engineering and Science (see Ref. 2), concludes that modeling and simulation are the key elements of modern science and engineering. Computational methods have even been referred to be the third pillar in terms of grounding science with theory and experiments.

Despite the importance of the computational methods in science and engineering, it can also be said that computational science and engineering software are in crisis. This is due to years of inadequate investment, a lack of useful tools, an absence of widely accepted standards and best practice and a shortage of third-party computational science and engineering software companies. The research and development needs are diverse, covering applications, programming models and tools, data analysis and visualization tools, and middleware.[3]

Even now, modeling and simulation researchers and engineers spend a large proportion of their time coupling different application programs or software systems, all of which detracts from the actual research and engineering activities. The limited interoperability of the tools and their complexity have become major hindrances to further progress. Today, a typical computational researcher or engineer must use several software applications, libraries, databases and data analysis systems from a variety of sources. She has to manage the different versions of her models and simulation results manually and share this information with the other project or team members in an outmoded way. Most of tools being used are incompatible and are most likely written in different computer languages, maybe even for different operating systems and in most of the cases they use different data formats and communication protocols. The need to integrate algorithms and application software is especially acute when researchers and engineers seek to create multi-level models supporting simulation of different levels of details within the systems. There is no single researcher or engineer who has the skills required to master all the

computational, application domain and software engineering knowledge needed to accomplish this task. To model such complex systems requires a multidisciplinary team of specialists, each with complementary expertise and an appreciation of the interdisciplinary aspects of the system. In addition a supporting software infrastructure is needed. This infrastructure should capture the expertise from multiple domains and integrate the results into a complete application software system. To be successful, application software must provide infrastructure for vertical integration of computational knowledge, including knowledge of the different disciplines.[3]

The crisis of modeling and simulation software explained above was a trigger for the development of a new integration platform. It has been clear from the beginning that the development model has to be as open as possible. This openness implies that the business model also has to change in the future. The future modeling and simulation business will no longer be in the platform solutions but rather in the simulation components and services that are running on top of an open operating system for modeling and simulation. This openness also means that a neutral democratic forum for the decision making on maintenance and further development has to be established. Due to these facts it was natural to use an open source development model and create an eco-system for the products and services running on top of the platform. Figure 1 below illustrates the eco-system approach.

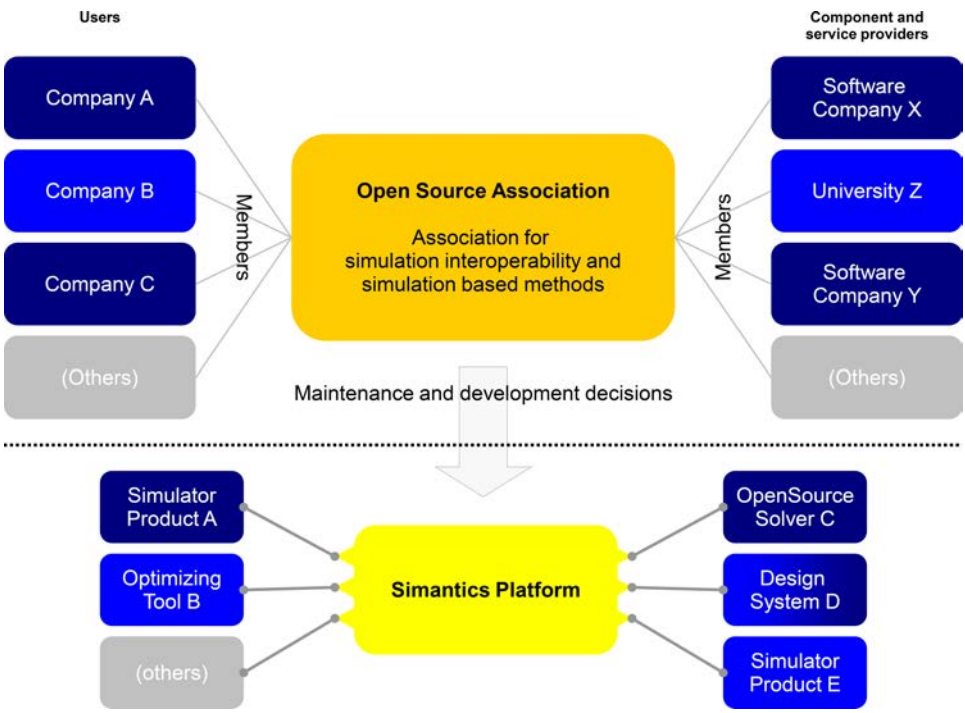

Fig. 1. Illustration of the eco-system on top of an open modeling and simulation platform.

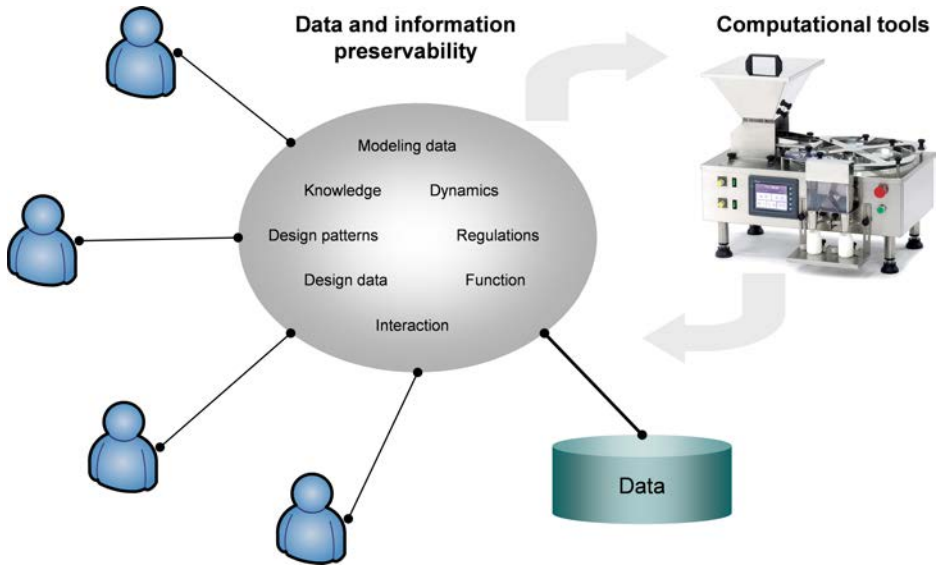

Fig. 2. Overview on the data driven approach.

Another starting point for the integration platform was to find a sufficiently powerful data modeling paradigm so that all the different data models of different disciplines and tools like simulators, design systems, data model standards, etc. could be described using this one unified language. This paradigm also has to enable the description of mappings between different data models. A Semantic, ontology-based integration approach has recently been used successfully in information system integration.[4–6] It has not been used to such an extent however in the field of modeling and simulation systems. As will be described in this article, the chosen approach utilizes heavily semantic data driven technique. All the data in the platform is described using semantic data models and ontolgies. Figure 2 above gives a high level overview of the idea.

We call this Open Ontology-based Integration Platform "Simantics". The article explains the main principles of the architecture of the Simantics platform. Article is arranged such a way that Sec. 2 analyzes the state of the art and describes the main needs and requirements of the platform, Sec. 3 explains the cornerstones of the architecture of the solution and Sec. 4 gives real life examples where architecture has been utilized. Conclusions and future work are discussed in Sec. 5.

2. Needs for the Integration Platform

2.1. *State of the art*

Before going into requirements for an integration platform we will give a short overview on the state of the art both on M&S integration standards and environments as well as on the use of semantic integration methods.

2.1.1. *Integration environments and standards*

IEEE Standard 1516 high level architecture (HLA)[7] is a general purpose architecture for simulation reuse and interoperability originally developed by the US military and later adopted by IEEE as standard 1516 in 2000. The HLA distributed architecture links simulations and interfaces to live systems, collectively known as federates. It calls the set of federates working together a federation. Federates do not communicate directly to each other but through a runtime infrastructure (RTI). The HLA approach separates the data model and the architectures semantics from the functions or methods for exchanging information. HLA includes three core specifications. HLA Rules includes 10 rules of interaction and responsibilities for federates and federations. A federate interface specification includes the services and interfaces required of the RTI and callback functions which federates are required to provide. The object model template (OMT) specification includes means to specify data exchange capabilities of federates (simulation object model, SOM) and the data to be exchanged during federation execution (federation object model, FOM).[8]

CAPEOPEN[9] is a standard to enable the reuse of process simulator components. CAPEOPEN was developed in various EU projects in the late 1990s. The architecture of CAPEOPEN consists of the Simulator Executive (COSE) used to run the simulations, unit operation components for single process component implementations, thermodynamic and physical properties components used to provide values for material properties and chemistry calculations and numerical solver components used to solve the resulting mathematics. The components are connected using concepts of unit which is a single component calculation, stream which contains specification of multiphase flow between units, port which connects units to streams and flowsheet which is a collection of units and streams. CAPEOPEN uses standard middleware namely Microsoft COM[10] and OMG CORBA[11] for component implementation. CAPEOPEN specifies four internal architectures, which can be used to construct a simulator. The most used is the sequential modular approach common in steady-state solvers, equation-based approach common in dynamic simulators and the sequential and nonsequential modular and modular hierarchical variations, which are not commonly implemented. Many existing simulation vendors offer CAPEOPEN components. Similar interoperability standard as CAPEOPEN is functional mock-up interface.[12] This specification has its origin in the Modelica community.

An important set of industrial integration standards are the ISO TC184/SC4[13] standards familiarly known as STEP. The actual STEP standard ISO 10303 aims to provide capabilities to describe and manage industrial product data throughout the life of the product. STEP has been developed since 1984 and includes standards for various industrial fields and life cycle stages. A comprehensive introduction to STEP is given in Ref. 14. STEP involves engineering users with a need to exchange product data with customers and/or suppliers. STEP employs a two layered data

model. Application resource model (ARM) and STEP integrated resources (SIR). For each domain a mapping called application interpreted model (AIM) is used to map between SIR and ARM. A STEP application protocol (AP) consists of ARM and AIM. Important industrial STEP standards include AP 221 for Functional Data and Schematic Representation for Process Plant,[15] AP 227 for Plant Spatial Configuration.[16] The SIR are modeled using the EXPRESS language defined in Part 11 of the standard. The language uses a schema with powerful features such as constraints and algorithms.

There exists also open source efforts for M&S integration platforms. For example Salome platform[17] has its origin in French industry. MyGrid/Taverna[18] on the other hand is aimed more to the scientific community. These efforts do not utilize semantic data modeling approach.

2.1.2. *Ontology development environments and knowledge presentation*

The selection of an existing ontology development environment to build on would be an appealing solution for fundamental design challenges. Unfortunately as the application of ontologies has not yet gained wide popularity the tools for ontology development are still mainly tools for research. The existing solutions provide little support for essential requirements such as teamwork, security, temporality and real-time processing. For the Semantic Web technologies some tools Protg,[19] KAON2,[20] Jena[21] exist for reading and writing ontologies, building ontologies, performing DL and SWRL[22] reasoning and SPARQL queries. The popular Protg includes generic user interfaces for building models and an extensible plug-in architecture for adding features. The KAON2 platform also provides a standalone server providing access to ontologies in a distributed manner using Java remote method invocation (RMI).

Knowledge representation can be classified based on the amount of structure in the data. Structured data representations are based on schemas, which specify the allowed constructs in the data. Typical examples of structured data are databases and XML. Semistructured data models have a self-describing structure such that the data itself is used to specify the allowed constructs. Typical examples of semistructured data are first-order logical representations and graph-based representations such as OWL. In semi-structured data, the structure describing part is often called metadata. Querying semi-structured data is more difficult than structured data since it has no predefined schema. Indexing, browsing, paths and patterns are possibilities for semi-structured queries.[23] The W3C SPARQL[24] query language is based on the ideas of SQL. The open knowledge base connectivity (OKBC)[25] specification specifies a procedural query language for generic knowledge bases.

The dominant mechanism for semantics in knowledge representation approaches is first-order logic (FOL) and its subsets. The logical approach interprets data facts in the given logic and the semantics are based on a set of logical sentences usually called axioms. Such semantics are studied in model theory. The popular programming language Prolog[26] has been widely used in expert systems and is based on

a Turing complete subset of FOL. Recent W3C standard OWL[27] uses description logics (DL)[28] which are a subset of FOL with effective reasoning mechanisms. The closed world assumption regards unstated information to be false whereas the open world assumption leaves unstated information open. This decision has significant implications on computational requirements and the way of modeling. Other more *ad hoc* specifications include XML Schema[29] and OMG OCL,[30] which is used in unified modeling language (UML) related standards. The ISO STEP representation language EXPRESS[31] includes a procedural programming language. For storage, transfer and presentation some serialization of the representation is needed. Most representations have an associated human readable text serialization. The OMG specifications meta object facility (MOF)[32] and ontology definition metamodel (ODM)[33] can be used to represent ontologies accessible through popular UML tools.

2.2. *Simulation and design system integration*

Design systems (CAD) and simulation systems have traditionally been separate in many areas of engineering. Naturally there are exceptions like in electronic circuit design or piece good manufacturing processes where the design has been done in a simulation aided manner for a long time. The reason for this is also evident. The more deterministic the target process or product is the easier it has been to utilize computational models. This tradition has been different however in many engineering sectors, for instance, in the process industry, machine/vehicle and construction industries. 2D and 3D CAD systems have already been used for decades but these systems do not include simulation features. There are instead numerous separate simulation tools which can be utilized in different phases of the engineering process. The following description of integration needs is taken from the power plant industry, but similar needs also exists in other engineering sectors.

Simulation is an excellent tool for engineering and optimization in different design phases of an energy production system. During the conceptual design phase, the engineer wants to seamlessly import process data and design cases to drive his conceptual design of equipment and instruments in a plant design environment. The engineer wants to automatically manage the changes from the process design solution through to his basic and detailed design and flag the items that need to be updated, deleted and added. He wants to automate the creation of the deliverables in the conceptual design phase including revisions. In addition he wants to be able to link in his own solutions, such as engineering calculation tools, cost estimating, etc., to enhance or customize the work process. In order to do this, a connection from the plant design system to a steady-state or pseudo-dynamic process simulation solution is needed. For this type of connection standard approaches like CAPE-Open can be utilized for defining the common information model between the systems.[9] If this type of connection existed, the user would be able to minimize the time needed to execute design studies, increase the design quality, fast track the basic and detailed

design phase and automate the creation of the deliverables to ensure consistency and quality.

During the basic design phase there is a need for process and control engineers to work in a more integrated manner. Process engineers mostly work in PIDs whereas control engineer receive the automation requests from the PID and design their control solution using logic and function block diagrams. Dynamic process simulation models would enhance the communication between the engineers because they would have a common live model to play with. This approach would also reduce the risk of change during the execution of the project.

During the detailed design phase the control engineer wants to generate a more detailed dynamic process simulation model based on 3D-plant model information. Using this more detailed model, the engineer would like to verify the detailed design in the early phases to ensure the correct design has been made and parts/components have or will be ordered, by performing frequent and quick dynamic simulation tests to ensure the design is in line with engineering practices. He also wants to finalize the control design and generate the function block definitions for the actual control system configuration. Using an auto generated link between the simulation model and control system (PLC or DCS), he would be able to reduce the control system testing time. Standard protocols like OPC Unified Architecture could be used for the communication between the simulator and control system.[34] The solution would lead to risk reduction in project execution and construction. With the simulation assisted testing, the risk of late changes or errors will be eliminated, thus streamlining the start-up.

Both before the plant start-up and during the plant operation, training is essential to avoid operational errors. The plant management wants to reduce the operator training time, and increase the quality of the training. Dynamic process simulation models produced in previous lifecycle phases could also be used for this purpose. The plant operator wants to be able to quickly trouble shoot his plant to reduce down time. He also wants to have quick and seamless access to both plant asset data as well as live plant performance data to make quick operational decisions. The plant operator wants to be able to dynamically simulate the plant and see this in relations to the process design (on the PID), the control system configuration (function block diagrams) and the 3D model, to optimize the safe performance of the plant, and also to make safety studies and or revamp/expansion plans. Using simulation assisted techniques, the operator is able to reduce operational risk, optimize plant performance, reduce time and risk of maintenance and/or revamp activities. The following Fig. 3 summarizes the aforementioned situation in plant engineering.

2.3. *Advanced use environment for existing modeling and simulation tools*

There are many simulation solvers, both in academia and in industry, that have sophisticated simulation algorithms, but they lack a good use environment.

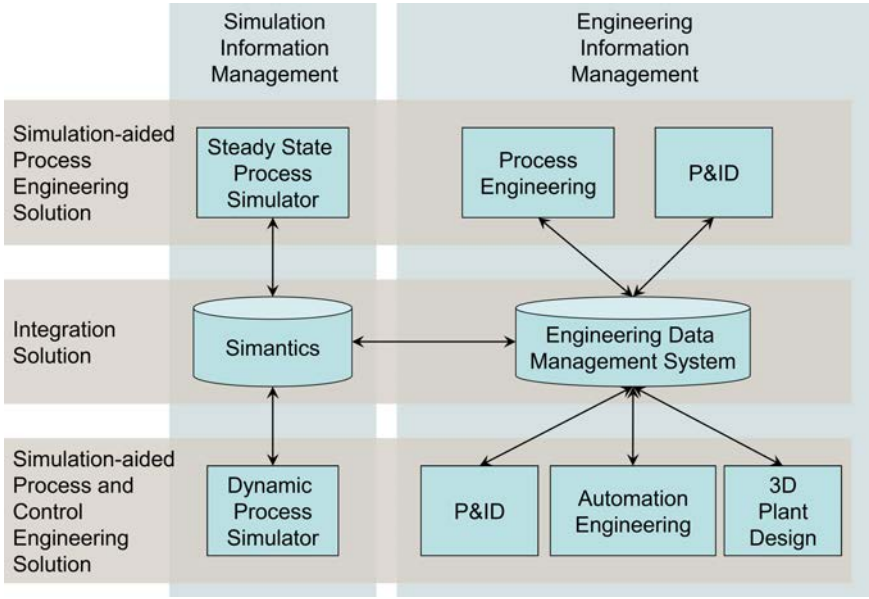

Fig. 3. Simulation driven approach in plant engineering.

Common needs for a use environment are certain preprocessing and post-processing capabilities as well as connections to other applications like design or control systems. Preprocessing capabilities include features such as 2D-flowsheeting support or 3D-geometry definition support, discretization support (meshing) as well as support for model validation, model structure browsing and editing, model component reuse, model documentation and searches, experiment configuration, model version control and team features. Post-processing capabilities include features such as 2D chart and 3D visualizations of the results, animations of the results both in 2D and 3D, experiment control visualization, etc. All these requirements, at a high level, are generic and thus there is no concept of different parties maintaining their own use environment. Instead one framework could be implemented which can be further specified according to these different purposes.

It has been a trend in academia to develop multi-purpose modeling and simulation environments. The needs in industry, however, are towards more specialized environments. Let us take an example. Modelica is an object oriented, equation-based modeling language.[35] The language and different Modelica tools are meant for multi-purpose system simulation. In e.g., business process simulation, the user wants to see a specified modeling view with causal diagrams in the form of mind maps. We should have a use environment that supports the creation of a new modeling view and under the hood the generic solvers are used for simulating the model. Domain specific visual modeling has been studied quite an extent in the literature.[36,37]

2.4. *Co-use of different modeling and simulation tools*

The need for the co-use of different simulation models rises from the same basis as the design system integration explained in the first section (Sec. 2.1). The products and production processes that are modeled are complex. Heterogeneous multi-level models are needed which can be utilized accross different engineering disciplines. Again the following requirement scenarios are taken from the process industry but the needs are also very similar to other sectors of engineering.

Dynamic system level process simulation tools are used for example in the energy sector for planning, operator support and training, operation state analysis, automation design and testing, safety analysis, and verifications in various stages of the power plant lifecycle. The advantages gained using these tools and methods can result in significant time and money savings, and improved safety. Steady-state process simulation tools are usually used in earlier phases of the plant lifecycle for dimensioning purposes and getting estimates in the offers phase. However, steady-state and dynamic process models can be co-used during the life-cycle in different ways. When the configuration of the steady-state and dynamic process model is described using the same neutral language, all of these different levels of co-use are easier to implement.

- Steady-state process models can typically act as initial value generators for dynamic simulations.
- The structure, once generated into the steady-state simulator, should additionally be more or less usable directly as a starting point for the dynamic model.
- Pseudo-dynamic mode often found from the steady-state simulators can be used together with a more detailed dynamic simulation model for co-simulation.

CFD (computational fluid dynamics) is used in the process industry for design and safety analysis. However, the 3D flow simulation of an entire process plant is not possible for performance and complexity reasons. There is a clear need for combined analysis so that certain process components are computed using CFD and the rest of the process using large-scale process simulation. This type of integration between large-scale process simulation and CFD exists in the literature. For example a prototype level co-simulation solution between FLUENT and Aspen Plus using CAPE-OPEN standard has been reported.[9] CAPE-OPEN is a standard for communication between computational components in process modeling environments (PME), suitable especially for sequentially modular process simulators. CAPE-OPEN has been used in the implementation of Integration Toolkit for Aspen Plus and Fluent,[38,39] where in the bi-directional coupling, information of flow rate, temperature, pressure and species components can be exchanged. The commercial applications have included chemical reactor, fuel cell system, coal-fired power plant and natural gas combined cycle plant.[40,41] Component-based integration platform CHEOPS has recently been implemented by using CORBA for chemical process modeling and simulation.[42] The simulation tools tested included Fluent, Aspen Plus, gPROMS

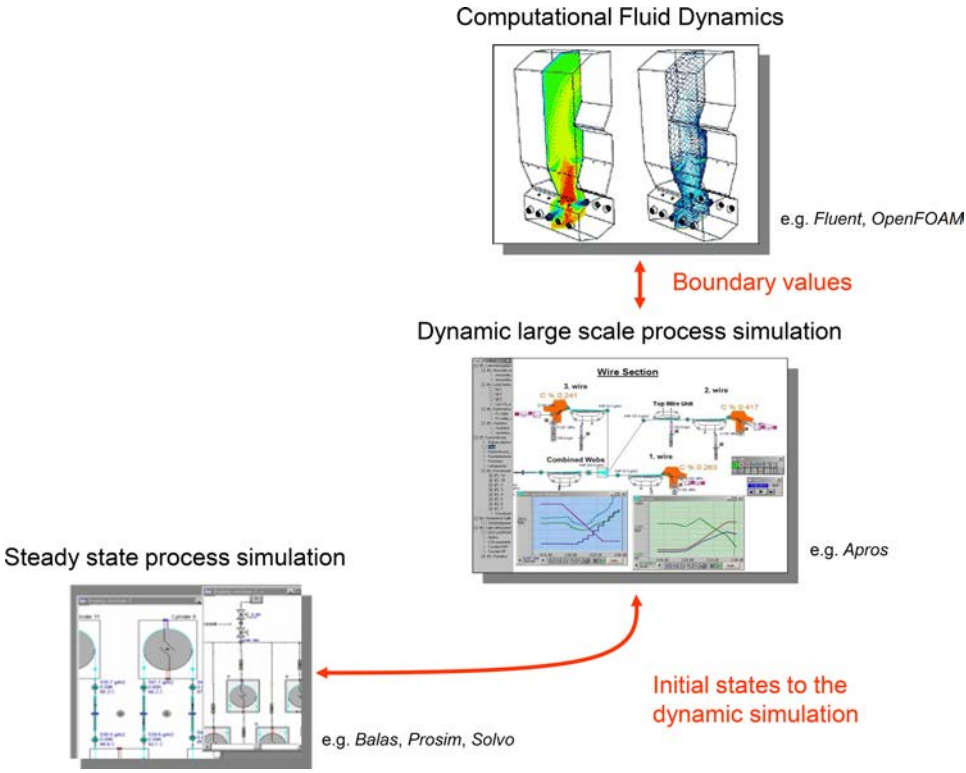

Fig. 4. Different commonly used levels of details in process simulation.

and Parsival. As an application, the dynamics of crystallization were studied with multi-scale coupled simulation by using Fluent and Parsival.[43] The following Fig. 4 illustrates the different levels of details from steady-state through dynamic process simulation to CFD computation.

These three different levels of detail are quite commonly used in the process industry. One can move however from these levels to even more detailed computations such as flow and structure interactions in piping and piping support or even to the simulation of the basic physical and chemical phenomena. One can also move to the more coarse direction. Physical production processes can be seen as subcomponents of a larger process, namely the business process. Both levels utilize models in their respective engineering, operation and maintenance. In a typical case a company might be contemplating a novel new product and its production. If the product is a manufactured component, FEM-models can be used to evaluate the physical aspects of the product and event-based techniques and models to evaluate and optimize the product line and transitions from one product to another. If the product is, e.g., chemical, then a steady-state chemical process simulator could be used to evaluate its thermodynamic soundness, and a dynamic process simulator to

evaluate its practical production. Environmental models are used to evaluate the environmental impact of the new product and production chain models to evaluate and to optimize the supply chain of the product and its derivates. Market models, competitor models and investment models take the economic aspect into consideration — brand models deal with marketing, advertising and media. All of this has to be combined at a business level so that people who make decisions can create 'what-if' scenarios, sensitivity analysis and test different strategies under uncertainties — with the help of all these models and simulators. The major goal is to have all the different approaches available to evaluate different concepts so that the case can be seen simultaneously from all angles.

In addition to the support of different levels of detail, users have also need to combine optimization computation and model uncertainty assessment into their simulation experiments. Optimization has been used in process modeling and simulation e.g., to tune process simulation model parameters and to find out the best process design concepts, utilizing a steady-state model. Traditionally simulation studies have as a result produced a single number or a time series. This result has then been used as a basis for decisions, often large and far-reaching. It has been noticed that the (often implicit) assumptions underlying the simulation models have uncertainties in them. Thus the results of the simulations are uncertain as well. This uncertainty has to be quantified and taken into account in order to make the model-based decision making more robust. Model uncertainties as such have been widely studied both quantitatively and qualitatively, but they are seldom included in the simulation environments. Together with optimization methods, this is an area needed by all the modeling and simualtion experts and thus should be included in a common platform, which all the integrated simulation tools could then utilize.

It could be concluded that there are many tools for modeling and simulation of a production facility, working in different levels of detail. A common integration platform is needed, however, to really interface with these different levels and to achieve a functional co-use between these levels. The following Fig. 5 summarizes the requirements.

2.5. *Simulation and control system co-use*

In this section the requirements from the point of view of simulation and control system co-use are discussed. We use process automation as an example scenario but the requirements in the field of building and machine automation are quite similar.

Traditionally, the factory acceptance test of a control system concentrates on the process interface and operator displays. It has been difficult or even impossible to test the more complicated functions. Thus a lot of testing has had to be done at site, which is expensive due to labor costs and possible loss of plant operation time.

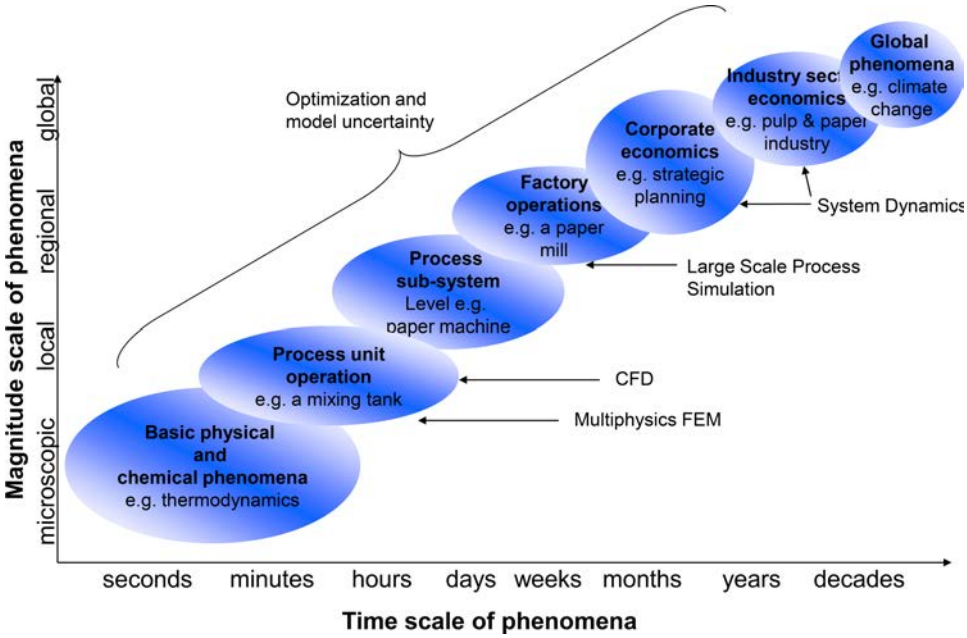

Fig. 5. Summary of the requirements.

A potential remedy has been seen in dynamic process simulation, and simulation has been successfully used in process and control concept design, control application testing and operator training. So far, the functionality of the control system has been included in the model either using the automation component libraries of the simulation tool, or a vendor-specific link to the real control system implementation. Both ways, the costs of using simulation have been high either due to the manpower needed to do the nonreusable control system configuration on the simulator tool, or due to the costs of real control system hardware and the vendor-specific, often tailor-made, software bridge between the simulator and the DCS.

During the last decade there has been a lot of development of automation domain information technology. This has opened new opportunities. In particular, the shifting of the DCS's towards PC-based systems and the development of specifications for open connectivity like OPC and more recently OPC Unified Architecture, have made it possible to expand the use of simulation-aided methods during an automation delivery.[34] The development of simulation tools and the increase of computing power have contributed to the more extensive use of dynamic process simulation.

Linking the control system to a dynamic simulation model of the process makes it easy to test even complicated control functions and to tune parameters. This way, a great deal of the work traditionally done at site can be done during the factory acceptance test. This improves the quality of the DCS delivered to the plant, cuts

down the time used to commission the control system, and enables an earlier start up of the plant's normal operation.

A dynamic simulation model linked to the DCS can also be used for integrated process and control engineering, to train the operators with the simulator prior to the start up, and for operator support throughout the lifecycle of the plant. Parts of the dynamic process model can even be used for model predictive control if they are fast enough for the purpose.

In integrated process and control engineering, the process and automation are designed simultaneously by engineers of the different domains and various options are tested even during the meetings between the engineers and the customer. This way the verification of the control application can already be started in the design and implementation phase. Furthermore, the process dynamics and controllability are taken into account earlier than in the traditional workflow.

Co-use of simulators and control systems sets requirements to the real-time communication, syncronization and simulation control facilities of the integration platform. If these are implemented in a neutral and efficient way they can also provide a solid bases for the communication and syncronization of different dynamic simulation tools or for the high level parallelization of several simulation experiments. High level parallelization here refers to the process level parallelization i.e., several simulator instances running in parallel, not to the code level parallelization of one single simulator. High level parallelization is also very useful in optimization and uncertainty quantification assessment cases, as discussed in the previous section.

2.6. *Team work and modeling and simulation information management*

Industrial information management has gone through many changes during the last few decades. In plant engineering for example, computer-assisted methods took the engineers away from the drawing tables and file cabinets (generation 2) and planted them in front of computer screens to draw process, automation, electrical and mechanical drawings. In many cases, we are still living the era of document management and document hotels (generation 3). However during the last decade, 3D-plant modeling has brought about yet another new way to design and model a plant. As electronic data management is becoming more efficient, we are currently moving away from document and drawing-based design towards plant model or product model-oriented design, where a conceptual model is defined in advance for the plant and product structures. The plant engineering project then uses this model and transfers the planning data between the various actors in this structural form (generation 4). Instead of the handover of documents, this method enables the handover of objects and models. The traditional drawings or 3D-models are views into the product model in this approach. Similarly, it can be anticipated that in the future we will be able to see how computational models

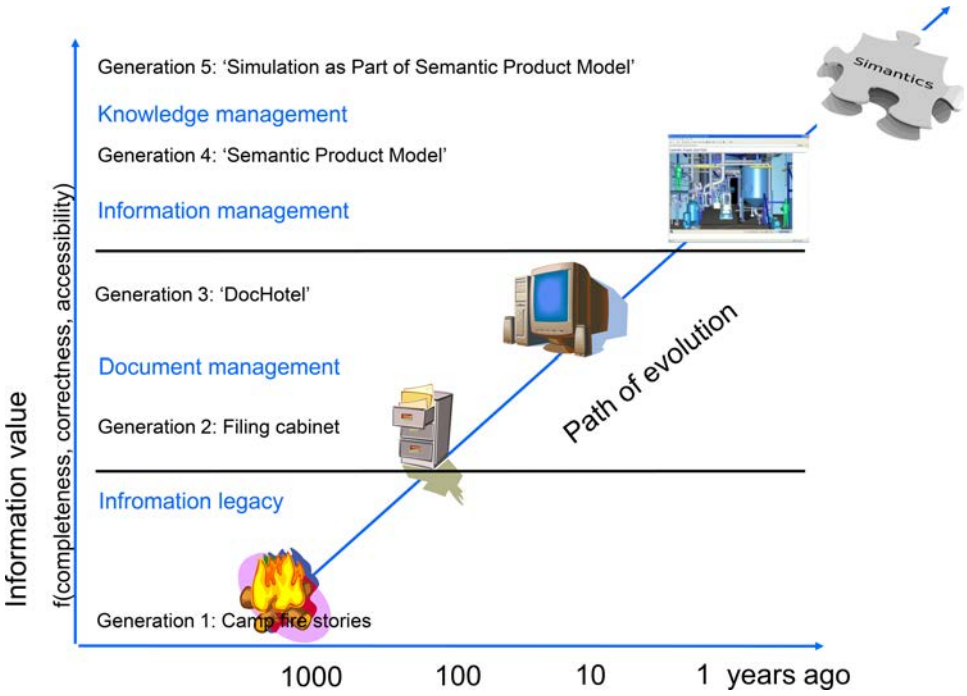

Fig. 6. Different generations of information management in industry.

will be integrated into the product models. Behaving product models will not only carry information on the structure of the product but also functions of the product in the form of algorithms (generation 5). The following Fig. 6 illustrates the situation.

Another big change in engineering is its increasingly networked nature. Participation in the design is increasingly global and involves people from various organizations. This, in turn, sets increasing demands on the design and simulation environment. Until now modeling and simulation environments have been almost always standalone applications. The only way to share the modeling work between users has been by sending files stored from the modeling environment. There has been a similar procedure with the simulation data. There has been very little team work support in term of sharing experiment results with the other users. The results have very often only been stored to the hard disc of the user who ran them and the only visible item for the others is a report document. With this paradigm there is no way that others could repeat the experiment or create a slightly modified version of the experiment. Another problem is the version control of the model configuration and simulation results. Nowadays both of these are quite often handled manually. We may have simulation results in our hand, but we do not know with which model or design version these were generated. There is a clear need for the

future integration platform to support team work and version control of both model configuration and simulation results.

The modeling and simulation environment involves extremely heterogeneous users. There are users who develop new and more efficient solvers and datastructures. Additionally, there are users who design reusable model libraries or users who use these libraries to model real world systems. Furthermore, there are users who only use these ready configured models for decision making. The team features of a simulation environment do not only mean the features to support distributed work of one of these user levels. The team features are needed in all these levels and also across them. This is a requirement that few current modeling and simulation environments tackle.

Modeling and simulation are considered to be too laborious to be included in everyday engineering work. With current simulation environments this is often true. It takes too long to model the target from scratch. However, with a connection to the design systems introduced in the first section and also with a clever reuse of model configuration components, this time can be significantly reduced. Model configuration reuse is one of the key requirements of the integration platform. In the lowest level this means that model configuration is formed in a hierarchical manner i.e., model components can consist of other model components. Efficient reuse also requires infrastucture for publishing and sharing the model components with others inside the same model, project, company or even more publicly.

A computational model is usually constructed for a specific purpose. Often the models may however also be applied to other purposes. For example, a dynamic process simulation model that has been constructed to support design can also be of service in automation testing and operator training as well as in performance analysis and optimization during operation of the plant. Nevertheless, speaking in terms of software technology, the model is often too tightly linked to its original application environment. In this sense, modular flexibility of software components in computational models is becoming an increasingly important requirement. In this way, e.g., the computational models developed and used in the design phase could also be used to support model-predictive control or maintenance in intelligent field equipment in an integrated way. Integration of the computational models of various phases of engineering into the plant model would enable seamless combination and thereby simulation of the operation of the various entities. This would support the introduction of new simulation-based working methods based in plant design.

2.7. *Stakeholders*

Simantics is designed to be a system to manage the complexity of large heterogeneous modeling projects. Due to this, Simantics itself is a large and complex system. To bring some order to this complexity, there are different roles, stakeholder profiles, for different tasks. They are described in this section.

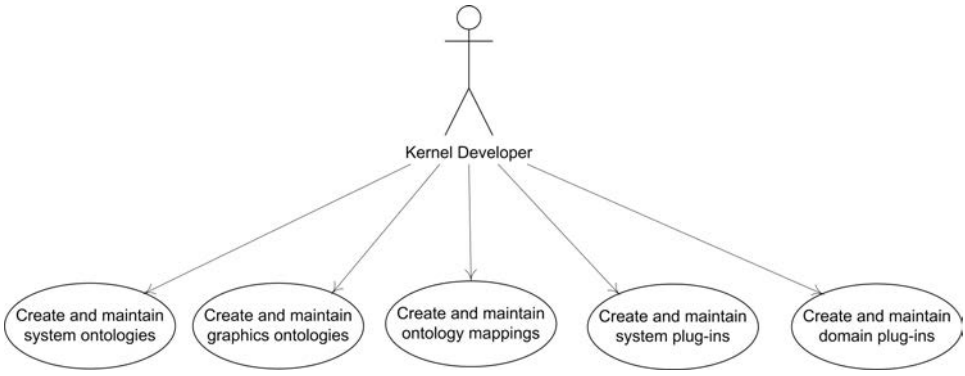

Fig. 7. Use cases of a kernel developer.

2.7.1. *Kernel developer*

Kernel developers are seasoned experts in modeling and simulation and are responsible for developing the modeling concepts and algorithms. They can be found in universities, research institutes and research and development departments of companies. Their use cases are shown in Fig. 7. Generally speaking, kernel developers are the users who do all the groundwork needed for higher level users to be able to work. They are responsible for creating and maintaining the base conceptualizations, i.e., ontologies for domain information models which are extended by library developers. In the 2D graphics context they define the ontologies used for graphical illustration of the information models, which allow higher level developers to create graphical elements as representations of domain model concepts.

In the ontology-based environment one goal is to keep models of different ontologies as separate as possible, e.g., for better reusability of models. The separation is also somewhat inherently assisted by the use of a binary relational graph data model. Yet, something is needed to bind the two models together to describe how they correspond to each other and furthermore, keep the models synchronized with each other. An example of this case is synchronizing a domain information model and a graphical illustration. To address this issue, kernel developers define ontology mappings.

Kernel developers create or customize existing user interfaces for domain model creation. They also create and customize graphical user interfaces and tools for editing the graphical illustrations of the information models.

2.7.2. *Library developers*

Library developers are domain experts who develop and maintain libraries of modeling constructs, which can be used to build models. Both domain information and graphics constructs need to be defined to support graphical modeling. The high level use cases are shown in Fig. 8.

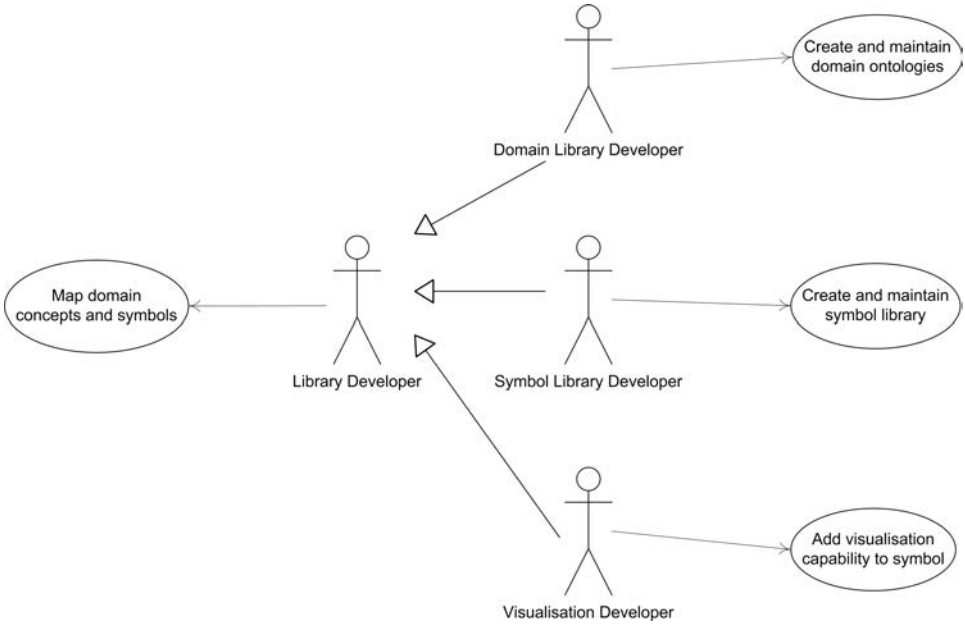

Fig. 8. Use cases of a library developer.

Domain library developers extend the domain ontologies created by kernel developers and other domain libraries. They define for example new process components for use in piping design. Companies may also want to define domain concepts for their own products.

In the flowsheet graphics context, symbol library developers use the graphical illustration ontologies and tools to define symbols. Symbols are considered two-dimensional graphical representations that may illustrate a domain concept or a part of a domain model. A single person may take on the tasks of both domain library and symbol library developer. This separation simply emphasizes that they are most likely not the same person.

Visualization developers add support for visualization capabilities in symbols to allow domain model information to be expressed graphically more intuitively for model configurators and model users. For example the fill level of a tank or the rotation speed of an engine could be depicted by the symbol itself in addition to seeing the values of these properties. Symbols are generally built for use in a particular domain, which also dictates to a degree what kind of visualizations may be useful. The amount of useful visualizations is somewhat dependent on both the internal and external complexity of what is represented by the symbol. Again, a single person may be both a symbol library and visualization developer, and this is a very likely scenario.

In order to keep domain models synchronized with graphics models, the ontology mappings defined by kernel developers may require extra information to be defined

by library developers. However, this depends on how a particular ontology mapping works.

Domain concept design is most likely to constitute the largest part of all library development. This is because all domain concepts, such as different process components and different products of companies, need to be defined separately whereas symbols can be reused for many similar domain concepts. Often it is highly desirable for symbols to have a standard look to them so that domain experts quickly associate the graphics with particular concepts. It may also be that some library developers, such as companies, are too busy to define symbols for their products thereby reusing standard symbols. On the other hand, it should be made possible for library developers, such as companies, to later create fancier and visually more capable symbols specifically for their own products. This implies that library developers need to be able to define multiple symbols for domain concepts and model configurators need to be able to use them interchangeably.

2.7.3. *Model developer*

Model developers are engineers who are developing models from the building blocks and tools created by library developers and kernel developers. Their high level use cases are shown in the following Fig. 9. In flowsheeting this development focuses on using the symbols and tools provided by library and kernel developers to build diagrams that may be mapped to domain models. Model developers do not define ontologies nor ontology mappings.

For the created diagrams, model developers may configure sets of monitors, charts and animations. Monitors and animations are a mechanism for graphically visualizing selected values from the illustrated information model. Charts are used for graphically viewing the simulation results.

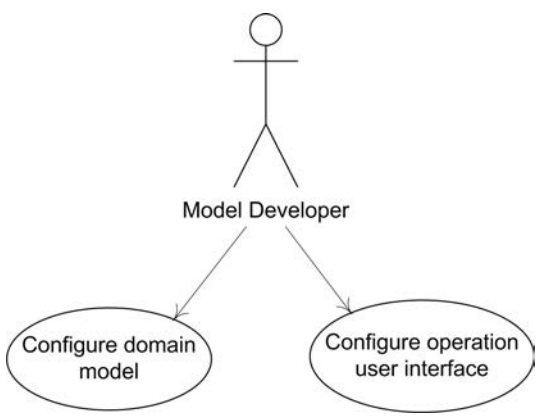

Fig. 9. Use cases of a model developer.

If possible, model developers may also want to test their models by simulating them from time to time. In this way they also act as model users. Especially when working with research simulators model developers are often also model users.

There is a large amount of possible individual user groups for 2D drawing tools. These include, for example, mechanical designers, piping and instrumentation designers, building designers, district heating network designers and system dynamics modelers. All of these users need somewhat different tools to be able to configure models intuitively, although the created graphics models may prove to be highly similar or even generalizable.

2.7.4. *Model user*

Model users use existing models to acquire information using simulations or other types of analysis. In some sense they are end-users of everything that has been created and configured by the previous user groups.

Just as with model developers, model users are not concerned with ontologies but use the tools provided for them. Any tools that require programming effort are created by kernel developers. It is also possible that some tools with simple form-type functionality could be created simply by means of graphical modeling. In this case they could also be created by model developers.

As stated previously, a model developer may often adopt the role of model user. However, there are also pure model users. A common factor for model users of all domains is that they put already configured models to use by tuning the parameters and analyzing them. Model users do not change the model structure. It may even be that model developers and model users use the same tools. On the other hand, the user interfaces created for model users may also be completely customized. One example of a case for customization is plant operators. Operators may be offered 2D views of e.g., gauges and meters to observe the state of a process more intuitively than by using a model developer user interface.

3. Cornerstones and Main Concepts of the Architecture

3.1. *Semantic data driven approach*

The single most important cornerstone of the Simantics architecture is the open and extensible semantic data model, which is used to represent the use environment and result deliverables of a simulation and modeling task. The data model is semi-structured, which means that the data contains the rules regarding its own structure. This approach allows for co-existence of different interlinked data models which can also be augmented with new pieces of data when needed. The data-centric approach places an emphasis on high quality representation, which increases the usefulness of the produced results. Simantics comes with standard models developed for common simulation and modeling patterns. As an example, a generic conceptualization of hierarchical, connected and parameterized models can be used as a

basis for different domain models. The data models are organized in layered conceptualizations i.e., ontologies which can be developed before or during modeling. Basic programming interfaces include a reactive programming query mechanism and a query and transformation language as well as means for attaching software interface implementations to data model concepts. Simantics aims at supplying an open solution library for most common simulation and modeling tasks. Layered plugin-deployed software packages can be further developed by users for automated modification and various analyzes of the models. For example, the same data model is used for tailoring specific model user services such as user interfaces based on generic and configurable preprocessing and post-processing capabilities. The data model is hosted in a database solution which includes distributed teamwork and version management services.

3.2. *Ontology-based mapping environment for integration*

As an operating system for modeling and simulation, Simantics needs to be able to handle various different data models related to different tools, computational methods and modeling methodologies. Successful co-use of these different domain services requires powerful integration and mapping tools between these different models. Simantics addresses these needs by supplying an ontology-based mapping framework for mapping and transforming models inside Simantics. The general approach is to import domain models into Simantics as they are and then transform the data further by using semantic mechanisms, which have been studied for some time e.g., see Refs. 44–46. The advantages of this approach include the fact that each specific data model is kept clean and separate. The semantic modeling and translation of the specific models as they are can also be in many cases easy or even automatic (e.g., XML or other standard formats). Elaborate data transformation mechanisms are also useful for generating models from other models. From the use case point of view the mechanisms for mapping and transformation enable the co-use of different domain models as well as the co-use of models of different levels of detail. Simantics offers the user a special simantics constraint language (SCL) for developing user-configurable mappings and transformations.

3.3. *Scalable solution from transient and fast-changing to persistent and shared semantic data*

The cases for modeling needs in a data-driven simulation and modeling system are highly versatile. To be able to fully establish the first cornerstone of data driven architecture, Simantics supports a wide range of mechanisms for extending the application range of the semantic data model. Simantics offers seamless support for four levels of persistence for semantic data in a unified model. Memory persistent parts of semantic data can be used to model quickly changing and transient structures so that the structures only exist during a modeling session. Workspace persistent structures are only stored in the user's local hard disk and can be used to

represent various cache or preference structures, which are generated or otherwise not publishable to all users of the distributed database. Database persistent parts of the data model are shared by different clients of a database server. Database persistent data is also fully versioned. Finally database persistent data can also be published and synchronized between a hierarchical configuration of database servers across organizations. In many cases such modeling databases can contain a huge amount of data which is normally indexed and accessed using a query language. In Simantics the data client accesses the data structure directly but only partially by using a local browse interface. In many cases the database clients employ their own indexes of relevant data. This direct manipulation of the raw data model enables fast client-side data representation and modification and change propagation albeit using more client memory for representing its working set of the data model. The semantic representation does consume a lot of memory but in many common huge data scenarios the data is primarily primitive and can be modeled using a primitive data structure modeling system included in Simantics.

3.4. *Seamless interface between simulated or measured real-time data and semantic graph data*

The wide range or persistence levels and different representations is especially common in simulation cases. For a semantically same attribute we can have input values in engineering systems, permanent configuration values in simulation models, different sets of e.g., dimensioning values in simulation models, computed result values and time series, real-time dynamic simulation or measurement values, etc. The representation of values for the same attribute can be modeled in completely different ways or not at all. Some attributes have many values, some are time dependent, some are persistently stored and some are not. To fulfill the integration goals the system needs to be able to represent and manage all these different pieces of data and most importantly to associate the data semantically together so that the data can be integrated. Simantics addresses these issues by semantic modeling of variables and their values and experiments and by specifying a software interface which is used to obtain values for a given configuration of semantically modeled variables. The interface imposes a semantic connection from a data value to the concepts of the data model while allowing for free aquisition of values from any source. In many cases the obtained values are backed by the semantic data model but can also be directly obtained from e.g., a simulator or measurement device. This framework for simulation data management makes Simantics unique among data modeling frameworks.

3.5. *High level concepts*

Simantics is an ontology-based integration platform for modeling and simulation. The Simantics system has a client-server architecture with an ontology-based modeling kernel and Eclipse-based client framework with plug-in interface.[47] This

section introduces some high level concepts of the Simantics Platform. Many of the high level concepts are the same as those used by Eclipse. However, Simantics augments these with the ontology-based appraoch and also introduces new concepts.

3.5.1. *Client-server concepts*

We call the semantic triplestore of Simantics a Core. It gives persistent storage for the semantic graph information and manages databases stored in the file system. Core provides an API that is available currently only for Java, but a .NET version is also under development. There is a proprietary optimized communication protocol between client (Core API) and server (Core). Cores can be linked into a core hierarchy so that every core can have one parent core and several children. Core stores semantic graph and versioning data in clusters. Only a subset of all clusters is stored into a local core. Core chaining enables work flow where smaller groups of modelers work with their local cores and later commit their results to a common remote core. Furthermore, these models can be commited to a root core by several different modeling groups. Version control features are also handled inside one core hierarchy.

There are also commercial triplestores for semantics modeling. These were evaluated during the Simantics development. There were several reasons why it was decided to build our own triplestore. The first was the scalability and performance requirement set for the open source triplestore. It was evaluated that a large-scale simulation model of, e.g., a power plant, consists of tens of millions of triples. It was possible to optimize our own triplestore for these needs. Another reason was the need for a browsing type of interface to the semantic graph. Many available triplestores only provided the query type of interfaces. The third reason was that the aim to produce an open source solution and, thus, a commercial solution to the core of the system, was unattractive. Some other reasons could also be listed, such as requirements for team features and version control.

Databoard is used in Simantics to control the solvers and to synchronize their execution and transfer data between the solvers and between solvers and the workbenches. There are several ready made components in the workbench e.g., for graph browsing, ontology development, 2D flow sheeting, validation, post-processing, etc. Components can be assembled into perspectives and furthermore components and perspectives can be composed into modeling and simulation products. Figure 10 shows some of the arhitectural concepts and their relationships.

3.5.2. *User interface concepts*

Many of the user interface terms used in Simantics are the same as in Eclipse.[47] The term Workbench refers to the desktop development environment. The Workbench aims to achieve seamless tool integration and controlled openness by providing a common paradigm for the creation, management and navigation of workspace

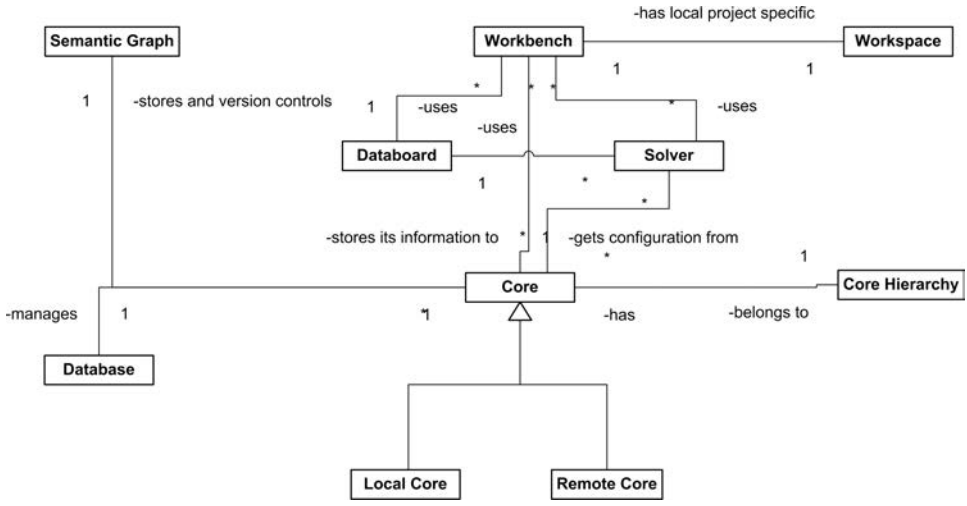

Fig. 10. Client-server architectural concepts.

resources. Each Workbench window contains one or more perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars. More than one Workbench window can exist on the desktop at any given time. Simantics utilizes different workbenches for the management of projects and for the management of the set of tools used in the projects.

Each Workbench window contains one or more perspectives. A perspective defines the initial set and layout of views in the Workbench window. Within the window, each perspective shares the same set of editors. Each perspective provides a functionality set aimed at accomplishing a specific type of task or piece of works with specific types of resources. For example, the Modeling and Simulation Perspective combines views that you would commonly use when modeling and simulating some real world system, while the Team Features Perspective contains the views and editors with which the user can share modeling information with other modelers. As a user works in the Workbench, he frequently switches perspectives. Perspectives control what appears in the menus and toolbars. They define visible action sets, which the user can change to customize a perspective. The user can save a perspective that he builds in this manner, making his own customized perspective which he can open again later. This information is stored in the local workspace information.

Most perspectives in the Workbench are comprised of an editor area and one or more views. Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the Workbench window contain operations that are applicable to the active editor. Tabs in the editor area indicate the names of resources that are currently open for editing. By default, editors are stacked in the editor area, but the user can choose to tile them, in order to view

source files simultaneously. 2D-flowsheet and textual editors are implemented in Simantics as editors.

Views support editors and provide alternative presentations as well as ways to navigate the information in the Workbench. For example, the Graph Explorer and other navigation views display model components and other resources that you are working with. Views also have their own menus. Some views also have their own toolbars. A view might appear by itself, or stacked with other views in a tabbed notebook. The user can change the layout of a perspective by opening and closing views and by docking them in different positions in the Workbench window.

Project management is a separate application that is used for managing different projects and workbenches related to those projects. Different cores are also managed through the project management. Different user interface concepts are shown in Fig. 11.

3.5.3. *Concepts related to the distribution of the software components*

As with Eclipse software components, Simantics components are also distributed through update sites. Simantics actually has two distribution channels, one for the software components and another one for the models. Kernel and library developers develop plug-in components that are distributed through the update sites. Model developers develop models that are used by model users and distributed through Simantics cores. However, these two channels are invisible to the end user. If e.g., some model needs certain plug-ins for the execution, the system seeks automatically for these from the update site. The user sees only the confirmation of the operation. However, if the user wants to control the distribution by hand, he can do so through the project management application.

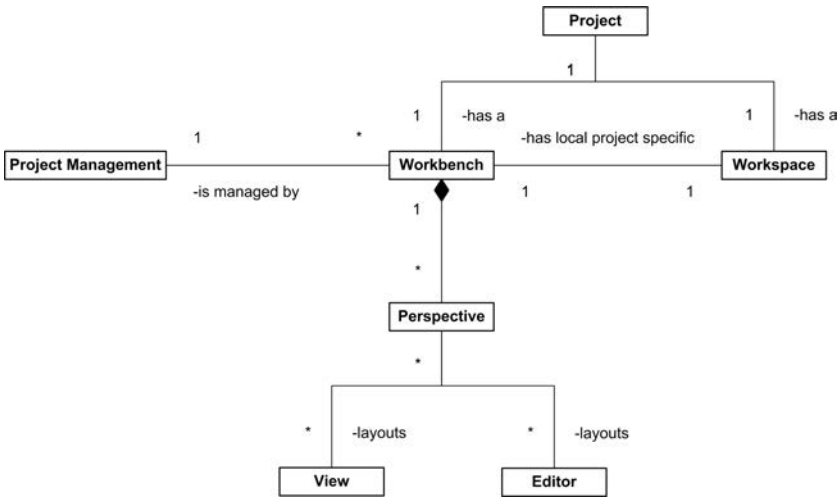

Fig. 11. Main user interface concepts.

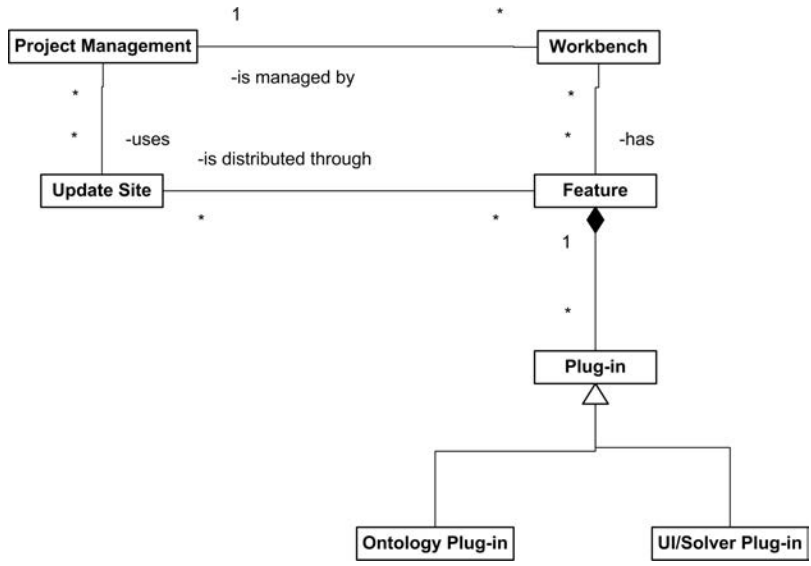

Fig. 12. Distribution of the software components.

The developed ontologies are composed of plug-ins. There are two type of plug-ins. Plug-ins that contain a semantic graph i.e., ontologies and plug-ins that contain a program code. This program code can be a user interface code or solver code. Plug-ins are furthermore assembled into features that are distributed through update sites. Figure 12 illustrates the situation.

3.5.4. *High level concepts for organizing data*

Semantic graph information is stored in a core. There are certain high level structures how the data is organized inside a core. The data in the core is organized into the libraries. Library is a grouping concept that can be used in different levels. Project information is stored in the core. A project is a container for models and other entities that are somehow related. Besides being just a container, the project also specifies which features are available in the Simantics-workbench and contains some configuration, for example the computational resources that are available for the simulation.

A Model is a container for interrelated modeling items. The following categories of items can be found: Data items, Activities and Views. The items are organized under the model in a hierarchical fashion: the items can be in libraries or under other modeling items. All data referred to by the items must also be included in the model, either by composition (ConsistsOf) or aggregation (IsLinkedTo). Every item has to be a part of (PartOf is an inverse of ConsistsOf) exactly one model, project or ontology. Usually the items linked to the model are part of the project or some ontology. The items can be graph or workspace persistent or transient.

Graph persistent items are those that are stored in the core. Workspace persistent items are only stored in the local workspace files and thus are user specific in the similar way to the layout of the user interface components. Transient items only exist during that certain session. The models are also typed. A model type typically contains a template for creating an empty model. It restricts which kind of modeling items can be created and asserts that certain data items are included in the model (for example some standard libraries).

Activities are defined as something that can be run. An experiment is an example of an activity. A simulation is an experiment made with a model. The results of the specific experiment are stored under experiment results. Figure 13 gives an overview of the concepts.

3.5.5. *Layered structure of semantic models*

Semantic modeling in Simantics is layered. At the bottom level, we have semantic graph that consists of triples. Subject, predicate, object triples or so-called statements are atomary elements of the semantic modeling paradigm.[48] A semantic graph has strong expressive power and is very similar to natural language. As in natural language, there are similarly many different ways to form a model description. An ontology language definition is needed to limit these possible ways. Web ontology language (OWL) is an example of such a language.[49] The W3C Semantic Web projects like OWL aim to develop technologies to improve the usability of the data and knowledge in the Web. Due to the fact that no-one can know what

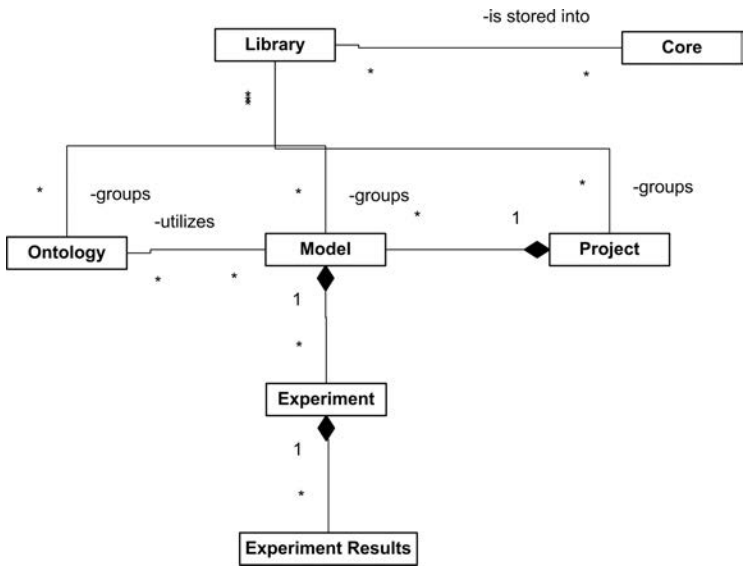

Fig. 13. High level concepts for organizing data.

information can be found from the Internet, the basic assumption of the knowledge world is permissive; the Semantic Web is based on the open world assumption (OWA): if something is not specified in the data model, it can still be correct, it is just undefined. This apparently simple assumption in the Semantic Web makes it difficult to apply the Semantic Web technologies for e.g., managing of modeling data of system simulation. This is due to the closed and well-defined nature of modeling domains. To set restrictions and rules for modeling data, to enable automatic model validation, the use of OWA would make the domain ontology development a demanding task. In technical systems modeling, it is common that a large amount of data is managed during the modeling process. An example of this is the finite volume method, in which the modeled domain, e.g., the geometry of a pipe, is divided into a set of control volumes that fill the whole domain; this is called discretization. The description of the discretized geometry, the mesh, usually contains large amount of data, which means that, tables and vectors are the natural choices for its representation. In OWL, these data structures are missing, which also decreases its attractiveness for technical systems modeling data representation.

Simantics has its own ontology description language called Layer0. Layer0 has similarities with OWL but has been specially defined for the description of engineering and simulation system ontologies. These domains also consist of more complicated information structures and data types than some other modeling targets. Simantics ontologies are defined using Layer0. The defined ontologies form a layered structure where the reusable generic ontologies like Structural Modeling Ontology are defined first and the product specific ontologies like Apros Process Simulation Ontology is defined by using these generic ontologies. By utilizing this layered definition approach software components like UI components can be programmed by using the more generic ontologies and thus reused in different modeling and simulation products. Layered structure is illustrated in Fig. 14.

4. Deployment Examples of the Architecture

Simantics platform development has been based on about 20 integration projects, during a five year time period. These have been done simultaneously with the platform development. The idea in the development has been to generalize the common needs from these different cases. This section gives an overview of the different case projects.

4.1. *Case base for the platform development*

The integration of simulation tools into Simantics started from the system level engineering simulators. The Advanced Process Simulator, Apros, is mainly used in power, pulp and paper industries.[50] It is used for process and automation design, automation testing, operator training and support, safety analyzes and for research and development. The modeling approach is dynamic and mechanistic. The control system components are modeled in the same simulation environment or the real

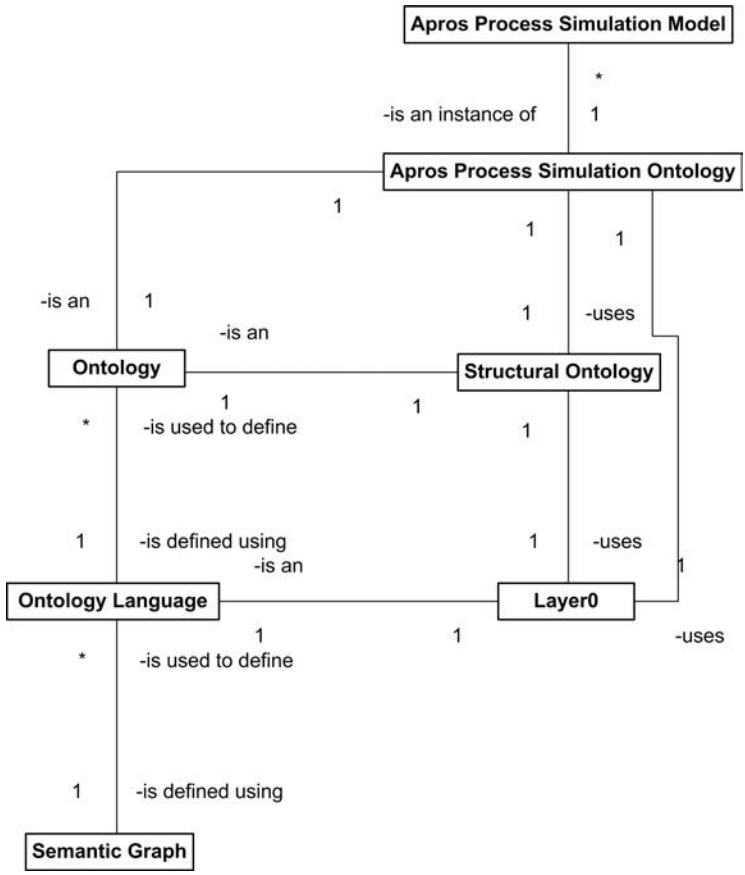

Fig. 14. Layered structure of semantic models.

control application can be connected to the process model. Apros has been developed by the Technical Research Centre of Finland and Fortum. Balas on the other hand is a steady-state process simulation tool developed by Technical Research Centre of Finland mainly for the pulp and paper industry.[51]

Modelica is an open standard modeling and simulation language developed by an international effort started in 1996. The Modelica Association, an international nonprofit organization, has been developing the open standard since then.[52] The Modelica language is intended to be used in modeling the dynamic behavior of technical systems which consist of components from different domains. It can be used especially to model large, complex and heterogeneous systems. It is an object-oriented high level language which can be used with systems that need high computational performance.[53] OpenModelica is an open-source environment, the purpose of which is to provide tools to build, compile and simulate models made by using the Modelica language. It is intended to be used for both industrial and academic

demands. The development and promotion of OpenModelica is supported by the nonprofit organization Open Source Modelica Consortium.[54]

Apros, Balas and Modelica/OpenModelica have been working as a significant case base for the Simantics development. They all represent cases of system simulation. As a result Apros 6.0 and Balas 4.0 software products will be released on top of Simantics. A system dynamic business process modeling and simulation environment has been built on top of Simantics and OpenModelica.[55] In addition a prototype of a generic Modelica modeling environment has been built on top of Simantics. Modelica cases have brought a different perspective to the platform development. Apros and Balas represent more traditional configurable simulation solvers where the simulation algorithms are hard coded to the solver and they are initialized with configurable process description. In Modelica the model configuration also includes the equations for the simulation. This description is compiled to form an executable solver. Both cases are possible for the semantic description and in the Modelica case, an ontology for the Modelica language has been defined. This way Modelica description can be serialized between the textual representation and semantic graph representation. All these three simulators have the needs listed in Sec. 2 and thus they benefit a great deal from the integration into the platform.

Another interesting case driving the development has been a Community Planning application, integrating a district heating network simulator and a building simulator. This application uses a two-dimensional graphical user interface approach with a map enhancement and multi-solver communication for the coupled DEvS (discrete event simulation) simulation on the Simantics platform. The Community Planning focuses on the different stages of a community design from the heating energy point of view during the community life-cycle. Typically, the community life-cycle is long, and estimating the heating energy performance of the buildings and network is challenging at the various phases of the design process. The first prototype of the application supports the design phase.[55] This application has especially contributed to the areas of simulator co-use and to the user interface.

Model checking is a formal method that can be used for verifying a hardware or software system design. A model of the system is constructed, and that model is then checked against the system requirements. The difference with more conventional verification and validation techniques is that the testing is exhaustive and the analysis covers all possible behaviors of the system model.[56] In order to ease the application of model checking at the early phases of the system design process, a set of tools have been created on top of Simantics to either automate some of the tasks in the model checking process or guide the modeler in those tasks that still need human interpretation. Thus, formal model checking tools benefit from the design system connections provided by Simantics as well as from the ontology mapping features through which the model generation can be automated. In addition many of the user interface components of Simantics have been reused in the work.

In addition to the system level simulators, some efforts have been already made to integrate 3D space discretized simulators like CFD- and FEM-based solvers to the

Simantics platform.[57] In the process industry for example CFD is used for design and safety analysis. However, the 3D flow simulation of an entire process plant is not possible for performance and complexity reasons. Instead 1D system codes are used for simulating the process of the plant eventually with co-operation or coupling with 3D simulation of the most important plant components, connected at inflow and outflow boundaries of the CFD model. In this kind of case, the Simantics platform provides a powerful co-use environment for these simulators.

As described in Sec. 2, the integration of design systems and simulation systems is inevitable in the future. There have also been some projects in this direction in the Simantics case base. The Siemens Comos system and the Intergraph Smart Plant product family are among the most popular engineering tools in the process industry.[58,59] Prototype level integrations into both of the systems have been implemented in order to map process simulation models from the PID and 3D information and additionally map function block diagrams from the automation design. Both of the connections are currently being developed further. Another case of a design system integration is the information management platform Sefram. Sefram is a product developed by THTH — Association for De-Centralized Information Management in Industry.[60] It has connections to e.g., Cadmatic and Autocad PID and 3D systems through XMpLant specification.[61–64] The aim with the Sefram integration has also been to map process simulation models from the PID and 3D models. There has also been a more research oriented effort to integrate product configurator system Tacton to Simantics.[65] The idea behind the integration is to use different possible product configurations in the system dynamic business process simulations and to use ontology mapping between the models.

There have also been more research oriented solvers that have been integrated to Simantics. The phase field modeling approach is used in physics e.g., for predicting microstructure in solidification process, such as in the casting of metals or polymers.[66] A phase field solver for metal casting was integrated to Simantics and a user interface was developed to the platform to manage the simulation runs through the web, to generate result databases from the simulation runs and to search and visualize results. Simantics contribution was an infrastructure for simulation result information management, for team features and for user interface components. In another very similar project case, several different nano cellulose solvers were attached to Simantics. Their execution was chained using a data flow type of experiment controller where the defined subset of outputs of one solver were given as an input to another solver. The simulation was also distributed to several servers. In addition to the contribution mentioned in the previous case, this case also used the co-simulation features in the Simantics framework.

Yet another case project worth mentioning is the thin client web solution called Simupedia, developed on top of Simantics.[67] Simupedia is a web portal for finding collaborative solutions to complex design and decision-making problems and answers to difficult questions using the power of simulations. More precisely, Simupedia is a publication channel where the users of the service can distribute and

use simulation related information (simulators, simulation codes, models, numerical experiments run on simulators published in Simupedia, raw data and documents such as manuals or scientific papers). Simupedia offers web browser-based user interface to the Simantics platform. Models and experiments can be viewed and executed through this browser-based interface. The Simantics workbench is needed for editing the model structure.[68]

5. Conclusion and Future Work

Over recent years, modeling and simulation has proven its benefits in industrial projects. The primary bottlenecks in the use of simulation are the costs and the timely availability of the simulation model. Both of these are primarily due to the fact that the model development is not integrated into the engineering work flow and data management. Recent experiences demonstrate that a sufficient system level model can be created entirely based on engineering data. It has been additionally concluded that the co-use of different modeling and simulation tools is currently insufficient. Multi-scale models combining different levels of detail would especially benefit from both better configurational and run-time co-use of different simulators. Furthermore, the co-use of the simulation environments and real-time systems such as control systems is inadequate. The challenge of integrating both design system features, simulation features and real-time control features including measurements into the same software architecture has been identified. Current simulation environment also lack team features that are essential in modern globally networked engineering projects.

We have introduced a solution that utilizes a semantic data modeling approach and combines this expressively powerful ontology-based design with fast data accessing capabilitites to simulation, measurement and control data. The data driven approach gives possibilities for automatic model validation, reporting, processing, annotations and linking. The layered structure of ontologies enables extendability and reusability. The heart of the integration solution is the ontology-based mapping mechanism that enables rule-based syncronization of different engineering and simulation models. The idea is to integrate the data from the different background systems into the environment "as is" using native data models. The model mapping is done inside the platform using ontology-based mapping rules. The bottleneck in the semantic approach is usually its scalability and processing speed. The suggested solution has been optimized for industrial use cases. It has also been designed to offer programming interfaces for the use cases from transient and fast changing to persistent and shared semantic graph data.

When designing and implementing an integration solution, it is higly important that it is as open as possible. This openess implies that the business model also has to change in the future. The future modeling and simulation business will no longer be in the platform solutions but rather in the simulation components and services that are running on top of an open operating system for modeling

and simulation. This openess also means that a neutral democratic forum for the decision making on maintenance and further development has to be established. Simantics platform has been published as an open source under Eclipse Public License (EPL).[69] For the democratic decision making, we have also established Simantics Division under the THTH association.[60] At the time of writing this article, there are 25 company, university and research institute members in the association.

The industrial examples in this article have been selected mostly from the process industry. The process industry has been the first application domain for the Simantics platform. However, the problems also exist in other domains e.g., in the building industry, in infrastructure, in the machine and vechicle sectors. The same technology approach for the integration could also be applied. In the machine and vechicle sector for example, there would be a need for an integrated open source alternative for the combined simulation of mechanics (MBS), control and electrical systems, structural analyzes (FEM) and flows (CFD). Open source pieces to all of these areas already exist but there isn't a good open integration solution. This kind of integrated solution could be utilized e.g., in the design and simulation of wind turbines. The building industry has also made considerable effort to define a common building information model specification IFC.[70] This product model can be utilized for integrating information from different CAD tools to different building simulation environments. There has already been some efforts to define Simantics ontology from the IFC building information model specification.

The current Simantics platform implmentation still has limitations. The lack of good 3D framework that would integrate with the semantic graph would be needed e.g., to integrate 3D space discretized tools like FEM and CFD tools or 3D plant or building design information into Simantics. There have been some efforts to create such a framework but there is no support in the current version.[57,71] Another limitation in current implementation is the lack of access control at the graph level. Scalable and easy to use access control would be needed for the team features of the platform to be fully functional. However, the development of such an access control model for the semantic graph is very challenging.[72] Yet another limitation is the lack of graphical ontology and mapping tools. In the current implementation, the ontologies and mappings are defined textually. There are some primitive UI components for the definition of ontologies but a diagram-based tool is needed in the future.

Acknowledgments

Simantics has been developed in several national and EU-projects. We especially want to express our gratitude to the main funding parties: The Technical Research Centre of Finland, Finnish National Technology Agency for Technology and Innovations, Fortum and Foster Wheeler.

References

1. The NSF Blue Ribbon Panel, Simulation-based engineering science, Report, National Science, Foundation, p. 88, 2006, http://www.nsf.gov/pubs/reports/sbes_final_report.pdf.
2. WTEC Panel, International assessment of research and development in simulation-based engineering and science, report, World Technology Evaluation Center, Inc., p. 426, 2009.
3. Computational Science: Ensuring Americas Competitiveness, Presidents Information Technology Advisory Committee (PITAC), 2005, http://www.nitrd.gov/pitac/reports/20050609_computational/computational.pdf.
4. Brandt S. C., Morbach J., Miatidis M., Theissen M., Jarke M., Marquardt W., Ontology-based information management in design processes, *9th Int. Symp. Process Systems Engineering*, Garmisch-Partenkirchen, Germany, 2006.
5. Kalfoglou Y., Hu B., Reynolds D., Shadbolt N., Semantic integration technologies survey, Technical Report 10842, Southampton, UK, 2005.
6. Szulman P., Hefke M., Trifu A., Soto M., Assmann D., Doerr J., Eisenbarth M., Using ontology-based reference models in digital production engineering integration, *Proc. 16th IFAC World Congress*, Prague, Czech Republic, 2005.
7. Defense Modeling and Simulation Office (DMSO), High Level Architecture website, https://www.dmso.mil/public/public/transition/hla/index_html.
8. Dahmann J. S., Fujimoto R. M., Weatherly R. M., DoD high level architecture. an update. *Proc. 30th Winter Simulation Conf.*, Washington, USA, pp. 797–804, 1998.
9. CAPE-Open Laboratorien Network, http://www.colan.org/.
10. Microsoft Corporation, COM: Component Object Model Technologies, http://www.microsoft.com/com/default.mspx.
11. The Object Management Group (OMG), ISO/IEC 19500, Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, 2004.
12. Functional Mock-up Interface specification, http://www.modelisar.com/.
13. International Organization for Standardization (ISO) TC184/SC4. TC184/SC4, Setting the Standards for Industrial Data, http://www.tc184sc4.org/.
14. National Institute of Standards and Technology (NIST), STEP The Grand Experience, 1999.
15. International Organization for Standardization (ISO), Geneva, Switzerland, ISO DIS 10303221: 2005(E): Industrial automation systems and integration product data representation and exchange Part 221: Application protocol: Functional data and their schematic representation for process plant, 2005.
16. International Organization for Standardization (ISO), Geneva, Switzerland, ISO DIS 10303227: 2005: Industrial automation systems and integration product data representation and exchange Part 227: Application protocol: Plant spatial configuration, 2005.
17. Salome open source platform, http://www.salome-platform.org/.
18. MyGrid/Taverna open source tools, http://www.mygrid.org.uk/.
19. Stanford Medical Informatics, The Protégé project, http://protege.stanford.edu/.
20. Information Process Engineering (IPE) at the Research Center for Information Technologies (FZI), Institute of Applied Informatics and Formal Description Methods (AIFB) at the University of Karlsruhe, Information Management Group (IMG) at the University of Manchester, KAON2, http://kaon2.semanticweb.org/.
21. The Jena Team, Jena A Semantic Web Framework for Java, http://jena.sourceforge.net/.

22. Horrocs I., PatelSchneider P. F., Boley H., Tabet S., Grosof B., Dean M., W3C Member Submission, 2004, http://www.w3.org/Submission/SWRL.
23. Abiteboul S. *et al.*, Querying semistructured data, *Proc. 6th Int. Conf. Database Theory*, Delphi, Greece, pp. 1–18, 1997.
24. World Wide Web Consortium (W3C), SPARQL Query Language for RDF, W3C Candidate Recommendation, 2007, http://www.w3.org/TR/rdfsparqlquery/.
25. Chaudhri V. K., Farquhar A., Fikes R., Karp P. D., Rice J. P., Open knowledge base connectivity 2.0.3. specification, Artificial Intelligence Center, SRI International and Knowledge Systems Laboratory, Stanford University, 1998.
26. Sterling L., Shapiro E., *The Art of Prolog. Advanced Programming Techniques*, 2nd edn, The MIT Press, 1994.
27. World Wide Web Consortium (W3C), OWL Web Ontology Language Overview, W3C Recommendation, 2004, http://www.w3.org/TR/owlfeatures/.
28. Baader F., Calvanese D., McGuinness D., Nardi D., Pate Schneider P. (eds.), *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2003.
29. World Wide Web Consortium (W3C), XML Schema Part 0: Primer, 2nd edn., W3C Recommendation, 2004, http://www.w3.org/TR/xmlschema-0/.
30. The Object Management Group (OMG), UML 2.0 OCL Specification, 2003.
31. International Organization for Standardization (ISO), Geneva, Switzerland, ISO IS 1030311: 1994: Industrial automation systems and integration product data representation and exchange Part 11. Description methods: The EXPRESS language reference manual, 1994.
32. The Object Management Group (OMG), Meta Object Facility (MOF) 2.0 Core Specification version 2.0, 2003.
33. The Object Management Group (OMG), Ontology Definition Metamodel. Sixth Revised Submission to OMG/ RFP ad/20030340, 2006.
34. OPC Foundation, http://www.opcfoundation.org/.
35. Modelica Association, http://www.modelica.org/.
36. Vangheluwe H., Sun X., Bodden E., Domain-specific modelling with AToM3, *Second Int. Conf. Software and Data Technologies (ICSOFT). Special Session on Metamodelling*, INSTICC Press, pp. 305–314, 2007.
37. Sen S., Baudry B., Vangheluwe H., Domain-specific model editors with model completion, *Multi-Paradigm Modelling Workshop at MoDELS*, Nashville, TN, 2007.
38. Zitne S. E., Multiscale modeling and simulation of advanced power generation systems, *DOE/NSF EPSCoR 2005 Conf.*, Dearborn, USA, 2005.
39. Osawe M., Fluent CAPE-OPEN COM/CORBA bridge and CO-compliant unit operation, *2nd Annual U.S. CAPE-OPEN Meeting*, Morgantown VW, 2005.
40. Syamlal M., Zitney S., Osawe M., Power plant analysis using CFD and process simulation, http://www.colan.org/CO%20Update/COUpdate07_Fluent_Article.html.
41. Sloan D., Fluent CAPE-OPEN COM/CORBA bridge and CO-compliant unit operation, *2nd Annual U.S. CAPE-OPEN Meeting*, Morgantown VW, 2005.
42. Schopfer G., Yang A., von Wedel L., Marquardt W., Cheops: A tool-integration platform for chemical process modelling and simulation, *Int. J. Softw. Tools Technol. Transf.* **6**(3):186–202, 2004.
43. Kulikov V., Briesen H., Marquardt W., Scale integration for the coupled simulation of crystallization and fluid dynamics, *Chem. Eng. Res. Des.* **83**(6):706–717, 2005.
44. Maedche A., Motik B., Silva N., Volz R., MAFRA a Mapping FRAmework for distributed ontologies, *Proc. 13th Int. Conf. Knowledge Engineering and Knowledge Management*, Siguenza, Spain, pp. 235–250, 2002.

45. Pierra G., The PLIB ontology-based approach to data integration, *Proc. IFIP 18th World Computer Congress*, Toulouse, France, pp. 13–18, 2004.
46. Qian P., Zhang S., Ontology mapping meta-model based on set and relation theory, *Proc. First Int. Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*, Hangzhou, China, Vol. 1, pp. 503–509, 2006.
47. Eclipse Foundation, http://www.eclipse.org/.
48. W3C RDF Primer, http://www.w3.org/TR/rdf-primer/.
49. W3C OWL Language features, http://www.w3.org/TR/owl-features/.
50. Advanced Process Simulator - Apros, http://www.apros.fi.
51. Balas Process Simulator, http://balas.vtt.fi/.
52. Modelica Association, http://www.modelica.org/association.
53. Fritzson P., *Principles of ObjectOriented Modeling and Simulation with Modelica 2.1*, Wiley-IEEE Press, 2004.
54. Open Source Modelica Consortium, http://www.openmodelica.org.
55. Simantics Demonstrations, https://www.simantics.org/simantics/demonstrations.
56. Lahtinen J., Valkonen J., Bjrkman K., Frits J., Niemel I., Model checking methodology for supporting safety critical software development and verification, *ESREL 2010 Annual Conf.*, Rhodes, Greece, 2010.
57. Gayer M., Kortelainen J., Karhela T., CFD modelling as an integrated part of multilevel simulation of process plants semantic modelling approach, *SCS Summer Simulation Conf. 2010*, Ottawa, Canada, 2010.
58. Comos Industry Solution, http://www.comos.com/solutions.html?&L=1.
59. Intergraph Smart Plant Solution, http://www.intergraph.com/ppm/default.aspx.
60. THTH Association, http://www.ththry.org.
61. Cadmatic, http://www.cadmatic.com.
62. Autocad, PID http://usa.autodesk.com/adsk/servlet/pc/index?id=8877989&siteID=123112.
63. Autocad Plant 3D, http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=13794692.
64. XMpLant Specification, http://www.noumenon.co.uk/.
65. Tacton Product Configurator, http://www.tacton.com/en/.
66. Majaniemi S., Provatas N., Deriving surface-energy anisotropy for phenomenological phase-field models of solidification, *Phys. Rev.* E **79**: 011607, 2009.
67. Laine J.-P., Thin-client architecture for Simantics simulation environment, Master's thesis, Helsinki University of Technology, 2009.
68. Simupedia, http://www.simupedia.com/public/.
69. Simantics web site, http://www.simantics.org.
70. IFC Overview, http://www.iai-tech.org/products/ifc-overview.
71. Luukkainen M., Karhela T., Ontology approach for co-use of 3D plant modelling and large scale process simulation, *Proc. 48th Scandinavian Conf. Simulation and Modeling (SIMS 2007)*, pp. 166–172, 2007.
72. Kalajainen T., An access control model in a semantic data structure: Case process modelling of a bleaching line, Master's thesis, Helsinki University of Technology, 2007.