HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Laboratory of Software Technology

Antti Villberg

Design Challenges of an Ontology-Based Modelling and Simulation Environment

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Espoo, October 3, 2007

Supervisor: Prof. Markku Syrjänen
Instructor: Tommi Karhela, D.Sc. (Tech.)

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Antti Villberg
Name of the Thesis:	Design Challenges of an Ontology-Based Modeling and Simulation Environment
Date:	October 3, 2007 Number of pages: 19+124
Department:	Department of Computer Science and Engineering
Professorship:	T-93 Knowledge Engineering
Supervisor:	Prof. Markku Syrjänen
Instructor:	Tommi Karhela, D.Sc. (Tech.)

This work is part of a strategic effort at VTT for creating a modeling and simulation environment with the goal of accelerating the adoption of simulation in engineering across disciplines. The modeling and simulation environment adopts rich semantic knowledge representation to enable integration and interoperability between existing simulation tools and engineering information management systems.

The vision of the modeling and simulation environment introduces the reader to the challenges and possibilities of simulation-based engineering, which includes comprehensive life cycle support for distributed modeling, simulation and semantic information management. Available technologies and tools are widely covered in the technology review. The treatment places emphasis on process modeling and simulation, which is the origin of this effort. The main contribution of this thesis is the identification and analysis of main design challenges of creating the environment presented in the vision.

This work discusses 14 design challenges reflecting the vision, technology review and prototype implementations developed during this work. The discussion is concluded with some general analysis and recommendations for implementation.

Keywords: ontology, process modeling, simulation, software architecture, software design

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN IIVISTELMÄ

Tekijä:	Antti Villberg	
Työn nimi	Ontologiapohjaisen mallinnus-ja suunnitteluympäristön suunnitteluhaasteet	
Päivämäärä:	3.10. 2007	Sivuja: 19+124
Osasto:	Tietotekniikan osasto	
Professuuri:	T-93 Tietämystekniikka	
Työn valvoja:	Prof. Markku Syrjänen	
Työn ohjaaja:	TkT Tommi Karhela	

Tämä työ on osa laajempaa strategista tutkimusta VTT:llä, jonka tavoitteena on kiihdyttää simulaation käyttöönottoa erilaisissa insinööritehtävissä luomalla mallinnus- ja simulointiympäristö. Mallinnus- ja simulointiympäristö pohjautuu ilmaisuvoimaiseen merkityksiä mallintavaan tiedonesitykseen, jonka avulla olemassa olevia simulointityökaluja ja suunnittelujärjestelmiä voidaan yhdistää toisiinsa.

Mallinnus- ja simulointiympäristön visio esittelee lukijalle haasteita ja mahdollisuuksia, jotka liittyvät simulaatiopohjaiseen insinöörityöhön, jossa hajautettu mallinnus, simulointi ja merkityksiin perustuva tiedonhallinta on elinkaariajatuksen mukaan kattavasti tuettuna. Kirjallisuustutkimuksessa käsitellään laajasti olemassa olevia teknologioita ja työkaluja. Työssä painotetaan erityisesti prosessien mallinnusta ja simulaatiota, jonka pohjalta tätä tutkimusta on alettu tekemään. Työn pääasiallinen tulos on esitetyn vision mukaisen ympäristön luomiseen liittyvien suunnitteluhaasteiden tunnistaminen ja analysointi.

Työssä käsitellään neljäätoista suunnitteluhaastetta, joita peilataan visioon, kirjallisuustutkimukseen, sekä työn aikana kehitettyihin prototyyppitoteutuksiin. Käsittely vedetään yhteen yleisten johtopäätösten ja toteutussuositusten muodossa.

Avainsanat: ontologia, prosessimallinnus, simulaatio, ohjelmistoarkkitehtuuri, ohjelmistosuunnittelu

Acknowledgements

I first express my thanks to my supervisor professor Markku Syrjänen and instructor Tommi Karhela. Your advice was wise even though I did not always take it.

Special thanks are due to all persons contributing to the modeling and simulation platform development, especially Kalle, Marko, Tommi, Toni and Tuukka. It has been a priviledge working with you.

Finally, to my family and friends, thanks for the support and patience.

Espoo, October 3, 2007

Antti Villberg

Contents

Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
AIM	Application Interpreted Model
APROS	Advanced PROcess Simulator
AR	Augmented Reality
ARM	Application Resource Model
ASCEND	Advanced System for Computations in Engineering Design
BALAS	A steady state simulation package for chemical processes
BFO	Basic Formal Ontology
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAPE	Computer-Aided Process Engineering
CAPE-OPEN	An interoperability standard for CAPE
CFD	Computational Fluid Dynamics
CL	Common Logic
CMMI	Capability Maturity Model Integration
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COSE	CAPE-OPEN Simulator Executive
DBMS	Database Management System
DCS	Distributed Control System
DDE	Dynamic Data Exchange
DIN	Deutches Institut für Normung (German organization for standardization)
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
EBNF	Extended Backus-Naur Form
EHSQ	Environment, Health, Safety, Quality

EJB	Enterprise JavaBeans
ER	Entity Relationship
EXPRESS	A STEP data modeling language
FACT	Facility for the Analysis of Chemical Thermodynamics
FCA	Formal Concept Analysis
FE	Finite Element
FEM	Finite Element Method
FOL	First-Order Logic
FOM	Federation Object Model
FTP	File Transfer Protocol
FVM	Finite Volume Method
GFO	General Formal Ontology
GOL	General Ontology Language
gPROMS	general PROcess Modeling System
HCI	Human-Computer Interaction
HLA	High-Level Architecture
HPC	High-Performance Computing
IAI	International Alliance for Interoperability , formerly Industry Alliance for Interoperability
IBIS	Issue-Based Information System
IEEE	Institute of Electrical and Electronics Engineers
IFC	Industry Foundation Classes
IFF	Information Flow Framework
IMTI	Integrated Manufacturing Technology Initiative
IPC	Inter-Process Communication
IRC	Internet Relay Chat
ISA	The Instrumentation, Systems, and Automation Society (American organization for standardization in

	automation)
ISO	International Organization for Standardization
ITU	International Telecommunications Union
JVM	Java Virtual Machine
KAON2	An ontology development platform
KIF	Knowledge Interchange Format
KM	Knowledge Management
LISP	LISt Processing
MAS	Multi-Agent System
MATLAB	A numerical computing environment and programming language
MIMOSA	Machinery Information Management Open Systems Alliance
MOF	Meta-Object Facility
NIST	National Institute of Standards and Technology
NSF	National Science Foundation
OASIS	The Organization for the advancement of Structured Information Standards
ODM	Ontology Definition Metamodel
OKBC	Open Knowledge Base Connectivity
OMG	The Object Management Group
OMT	Object Model Template
OPC	Open connectivity via open standards, formerly OLE for Process Control
OPC DA	OPC Data Access
OPC UA	OPC Unified Architecture
OSA-EAI	Open Systems Architecture – Enterprise Application Integration
OSGi	Formerly Open Services Gateway initiative. A service architecture or the associated organization

OWL	Web Ontology Language
OWL-DL	Web Ontology Language – Description Logic
P&ID	Process and Instrumentation Diagram
PDE	Partial Differential Equation
PFD	Process Flow Diagram
PKI	Public Key Infrastructure
PLC	Programmable Logic Controller
PLIB	Parts Library (ISO 13584)
PSE	Process Systems Engineering
PSK	Finnish organization for standardization in process industries
RDF	Resource Description Framework
RDL	Reference Data Library
RMI	Remote Method Invocation
RTI	Run-time Infrastructure
SBES	Simulation-Based Engineering Science
SD	System Dynamics
SEI	Software Engineering Institute
SESKO	Sähkö- ja elektroniikka-alan standardisointijärjestö (Finnish organization for standardization in electricity and electronics)
SFS	Suomen Standardisoimisliitto (Finnish Organization for Standardization)
SIR	Step Integrated Resource
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOM	Simulation Object Model
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Simple Query Language
SSH	Secure Shell

STEP	Standard for the Exchange of Product Model Data
SUO	Standard Upper Ontology
SWRL	Semantic Web Rule Language
TCP	Transmission Control Protocol
THTH	Teollisen hajautetun tiedonhallinnan yhdistys, (Finnish) THTH Association of Decentralized Information Management for Industry
UDDI	Universal Description Discovery and Integration
UI	User Interface
UML	Unified Modeling Language
UNIX	A computer operating system
V&V	Verification and Validation
VTT	Valtion Teknillinen Tutkimuskeskus (Technical Research Centre of Finland)
W3C	The World Wide Web Consortium
WSDL	Web Service Definition Language
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XML	eXtensible Markup Language

Glossary

access control	protection of resources against unauthorized access; a process by which use of resources is regulated according to a security policy and is permitted by only authorized system entitites according to that policy [Sh00]
algorithm	a step-by-step problem-solving procedure, especially an established, recursive computational procedure for solving a problem in a finite number of steps
auditing	the tracking of actions and activities in the system, including security related activities. Audit records can be used to verify the operation of system security [OP06]
augmented reality	combining computer-generated images into real-world images e.g. video streams
authentication	the process of verifying an identity claimed by or for a system entity [Sh00]
authorization	a right or a permission that is granted to a system entity to access a system resource [Sh00]
axiom	something that is accepted true without proof
capital facilities industry	a set of industrial branches involving large facilities such as production plants
coarse-graining	deriving simpler larger-scale models from more fundamental models
computational science	the study of scientific and engineering problems using numerical mathematics
conceptualisation	an abstract, simplified view of the world that we wish to represent for some

	purpose [Gr93]
data	symbols produced with some equipment and adhering to a coding system, with a potential meaning to the interpreter of the data [HKL95]
data integration	the process of combining data from different sources and providing the user with a unified view of these data
decision support	providing automated support for human decision making
declarative approach	describes what instead of how
dynamic simulation	a simulation, which involves calculation of the time-dependent dynamic or transient behaviour of a system
dynamic testing	testing of the dynamic behaviour of models or code. Typically involves running the code or performing simulations on the model
experiment	a test conducted on a simulation model in order to answer questions about the modeled entity
expert system	a program that uses available data, heuristics, and inference to suggest solutions to problems in a particular discipline
first principle model	a model that is based on fundamental physical and chemical laws
first-order logic	a logical language which includes predicates and quantification over variables
flowsheet	a graphical representation which connects units or operations with flows of e.g. material or data

graph data model	a data model consisting of vertices and edges between those vertices
high performance computing	typically refers to supercomputers used in scientific research
historical data	time series or other data collected from simulations or measurements and stored for reuse
information	the understanding gained from human interpretation of data [HKL95]
knowledge	information based on experience or rationalization of the “truthfulness” of the information [HKL95]
life cycle	a progression through a series of differing stages of development
mapping	specification of a relationship between concepts or sets of concepts, which implies enforcing given conditions such as equations or morphisms
material property (thermodynamic)	intensive thermodynamic parameters which are specific to a given material
meta-data	usually data about data
middleware	software that serves as an intermediary between systems software and an application
model	a description of a system (often simplified)
model configuration	a definition of (or a work process of describing) the interconnections, parameter values and a set of specified initial states of modeling components
modeling	constructing a model. Includes both model configuration and model development i.e. defining process structure and behaviour

multi-domain	combining models from different fields
multimodal user interface	providing the user multiple modes of interface with a system e.g. voice, touch
multi-scale	combining models with different temporal or spatial granularity
object-orientation	a design paradigm, in which data and associated operations are bundled into objects
ontology	(1) philosophically, the study of what might exist [Fl03] (2) from the knowledge engineering point of view, an explicit specification of a conceptualisation [Gr93]
operator (process, plant)	a person responsible for the operation of a plant
optimization	the procedure or procedures used to make a system or design as effective or functional as possible. Usually involves finding the extreme values of a mathematical function in a given domain
plant design system	a software suite used for designing e.g. process plants
primitive data	pieces of numerical, textual or binary data
process	a sequence of chemical, physical or biological operations for the conversion, transport or storage of material or energy [In97]
process automation system	see process control system
process control system	the hardware and software involving automated control of the process equipment
process plant	facilities and structures necessary for performing a process [In97]
process synthesis	automatic generation of process models

	and designs
query	a question about a data model
real-time	(1) describes an application which requires a program to respond to stimuli within some small upper limit of response time. A hard real time application does not tolerate failures to meet deadlines, whereas a soft real time application does to some extent. (2) refers to an activity happening in actual “wall-clock” time
reasoning	the process of drawing a conclusion from a set of premises
relation	a semantic association between two resources in a semantic graph data model
representation	data coded with a set of symbols
resource	a node in a semantic graph data model
rule	a familiar name for Horn logic clauses of the form of an implication between an antecedent (body) and consequent (head)
schema	specifies constraints for the structure of a data model
semantic	with a meaning
semantic web	a vision of the World Wide Web with more machine-interpretable content
service	a self-contained piece of software with well-specified interfaces and no context- or state-dependence to other services
simulation	an experiment made with a model
social media	online technologies and practices that people use to share opinions, insights, experiences, and perspectives
soft computing	a family of methods that target complex phenomena, for which analytical solutions

	have not been found
software agent	anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [RN03]
software architecture	the fundamental organization of a system embodied in its (software) components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IE00]
software component	a unit of composition with contractually specified interfaces and explicit context dependencies only [SGM+02]
spatio-temporal	existing in both space and time; having both spatial extension and temporal duration
statement	represents a relation between two resources in a graph data model
static testing	testing of models and code without simulating the model or running the code
steady-state simulation	a simulation, which involves calculation of (time-independent) balances
syntax	the way in which symbols are ordered and connected
tacit information	as opposed to explicit information, information which has not been documented
taxonomy	classification into (usually hierarchically) ordered categories
topology	interconnections between e.g. simulation model components
tracking simulator	a simulator, which attempts to synchronize its model with measurements

	from the simulated system
training simulator	a simulator designed for training purposes such that the use of the simulator resembles the use of the actual physical system
transaction	and activity or request in software for which usually atomicity, consistency, isolation and durability are required
unit operation (process model)	Specified process functions that perform one or more defined transformations on process stream(s)
validation	the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements [IE90]
verification	the process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase [IE90]
version control system	a database capable of tracking the revisions made to an information model
virtual prototyping	using modeling and simulation instead of physical models for developing new products
wisdom	a kind of knowledge which has refined during a long time and is based on experience and learning [HKL95]

Tables and figures

1 Introduction

1.1 Background and Motivation

The application of modeling and simulation in engineering has experienced steady growth for decades. The emergence of a truly global economy places great challenges on industrial production but promises also great possibilities for adaptive and efficient players [LHV+01]. The fairly immature discipline of computational science has been identified as a major possibility for creating competitive edge at the increasingly competitive marketplace. Reports and workgroups around the world have a unique consensus about the growing importance of computational science for all engineering disciplines [Na06][In00]. The governmental interest has been concretely demonstrated in previous and ongoing *Finnish Funding Agency for Technology and Innovation* (Tekes) technology programs in Finland and in large investments for high performance computing systems around the world [AT06][HP07]. Successful application areas for simulation include decision support for design, virtual prototyping and training. The environmental, health, safety, quality (EHSQ) issues are also gaining economical and political importance and can be addressed using simulation models [BG02][Ko01].

This work is part of an ongoing strategic effort at the *Technical Research Centre of Finland* (VTT) aimed at creating an *ontology-based* environment for modeling and simulation. The roots of this development are in VTT research on *process simulation*, industrial information models and software architectures. Such work is also motivated in many recent reports e.g. [Sö05]. The work is based on the developments in the domain of process industry but important synergies have also been identified in other modeling and simulation domains researched at VTT and this work will also address selected issues from these other domains.

The modeling and simulation environment aims to drive the adoption of modeling and simulation by removing some barriers and providing higher benefits. The work addresses the needs of current distributed network economy [LHV+01] and focuses on user interfaces with high usability and productivity. A strong emphasis is given to emerging areas of multi-domain and multi-scale modeling by means of integration and interoperability between existing and new modeling tools and technologies. Integration of simulation into current engineering systems is also important. Additional focus will

be placed on providing means for offering advanced automated decision support services for the modeling user based on the semantically rich ontology-based information model.

1.2 Objectives and Scope

The focus of this work is on identifying and finding possible solutions for the major *design challenges* of designing an ontology-based modeling and simulation environment. The initial scope and application area of the modeling and simulation environment will be concentrated on process modeling and simulation, but a widening of the scope will be anticipated already in this work. Since this work is about the overall design and architecture of a large software system, the treatment will remain at high level of abstraction to capture the most relevant challenges of the system as a whole and to restrict the unavoidable wide scope of this work.

1.3 Structure of the Thesis

This thesis is divided into 6 chapters.

Chapter 1 is this introduction with background, motivation, objectives and scope for this thesis.

Chapter 2 is a statement of vision of an ontology-based modeling and simulation environment. As this work is part of a larger strategic effort this chapter provides further background and insight into the nature of this work.

Chapter 3 is a review of relevant technologies regarding this work. The selected technologies cover a wide range of technologies needed to implement the system as well as technologies, which will be integrated to the modeling and simulation environment.

Chapter 4 reiterates the problem statement of this work. The discussion in the first three chapters is collected to give an elaborated account of objectives and scope.

Chapter 5 is an analysis of the design challenges of an ontology-based modeling and simulation environment. A selected set of major design challenges are discussed and the discussion is further analysed and some suggestions given.

Chapter 6 concludes this work with some evaluation and prospects for the future.

2 Vision

2.1 Overview

A discussion of an environment for modeling and simulation requires a proper definition for the concept of modeling. As it happens, this turns out to be one of the most important design challenges. The concept of a model and the act of modeling are *inherent* for all thinking beings. A child begins to model the world right after his birth and proceeds to form complex *mental* as well as *physical* models of the surrounding world. Each human being has a distinctly personal way of modeling the world and the need for *common understanding* has been identified already at the early stages of the development of culture. Through their pursuit of science, men have been able to accumulate a wealth of knowledge of the surrounding world and its ways. *Based on this groundwork* the discipline of engineering has emerged, which has had its effect on almost all aspects of modern human reality. Large parts of the overwhelming success of the engineering discipline can be attributed to practical usage of modeling. Engineers have been able to take the rigorous *scientific models* of physicists, chemists and mathematicians and apply them to *real world problems*. Often this has required *simplifications* and *approximations*. It can be argued that actually the ability of novel simplification, approximation and decomposition has been one of the major attributes behind the success of the engineers. Another essential factor has been *standardization*, which has enabled the crucial common understanding between engineers along with industrial advances such as mass production.

2.2 Towards semantic models

Engineering use of modeling requires a *semiotic* study of *representation* and *interpretation*. Representation is needed to store and communicate models and interpretation is needed to utilize them. Representation is based on some *syntax* i.e. a set of rules for the structure of the representation. Syntax can range from *informal* such as natural language to *formal* such as mathematics. Interpretation is the act of determining the *semantics* i.e. the *meaning* of a model. Determining the semantics can be subjective and complicated and includes questions from *pragmatics* and *epistemology*. Pragmatics considers the difference between the *intended* and *interpreted* meaning of a representation. Epistemology deals with the subjective issues of knowledge and belief. A careful consideration of these semiotic issues is at the heart

of all information systems [Sh99]. The following categorization [HKL95] provides insight into interpretation.

> *Data:* symbols produced with some equipment and adhering to a coding system, with a potential meaning to the interpreter of the data.
> *Information:* the understanding gained from human interpretation of data.
> *Knowledge:* information based on experience or rationalization of the "truthfulness" of the information.
> *Wisdom:* a kind of knowledge which has refined during a long time and is based on experience and learning.

A major challenge in information management is to bridge the gap between data and information i.e. to capture and quantify the intended interpretation. The above categorization implies that representations of models can only consist of data and that the higher levels are a product of *intelligence*. Engineering models have traditionally utilized formal or semi-formal syntax with some well-defined semantics i.e. mathematics such that the data could be most effectively turned into information by the engineer manually or using a suitable software. Another popular approach has been the introduction of *meta-data* (data about data), which somehow describes the actual data. Popular meta-data approaches include *schemas*, which limit the usage of syntax and thus introduce some semantics.

2.3 Towards modeling-based engineering

The introduction of computers has transformed data management and manipulation. The capabilities of computers to store and process data vastly outperform humans. The application of computers has introduced a relatively new discipline of *computational science*. This combined application of *physical* scientific models and *numerical* engineering models has been increasingly successful in industry. Industrial simulation has been used for decades in various industries but in many domains, its use has been limited. In many cases, the benefits of applying modeling and simulation have been outweighted by the costs incurred. A top-level view of the costs and benefits are

> *Costs:* software costs, building, maintaining and using the models.
> *Benefits:* decision support, training.
>
> *A key goal of this work is to explore designs needed to lower the costs and to magnify the benefits in order to make modeling and simulation more attractive in the designed environment.*

The slow adoption of simulation into the engineering industry as a whole has been recognized worldwide and several activities are under way all around the world to address this. In Finland Tekes has been a major contributor to the dissemination of simulation and modeling technologies. Tekes has funded several technology programs addressing questions related to modeling and simulation such as the current *Modelling and Simulation* (MASI, 2005-2009) [AT06]. In the United States a National Science Foundation (NSF) report on Simulation-Based Engineering Science (SBES) [Na06] concludes

> *SBES* is a discipline indispensable to the nation's continued leadership in science and engineering. It is central to advances in biomedicine, nanomanufacturing, homeland security, microelectronics, energy and environmental sciences, advanced materials, and product development. There is ample evidence that developments in these new disciplines could significantly impact virtually every aspect of human experience.

Another notable statement of vision for future modeling and simulation includes the Integrated Manufacturing Technology Initiative (IMTI) roadmap [In00]. It calls for full integration of information, complete interoperability among systems and intelligent utilization of these resources to supply real time decision support and automation from engineering to enterprise utilizing science and experience with modeling and simulation at the heart of all operations. For process industries, a similar vision has been stated by FIATECH [FI04].

2.4 Challenges

A major contribution in modeling and simulation cost comes from the work required to build the models. Accurate modeling requires a wide range of expertise in the application and in the solution methods. Building a dynamic process model of a typical power plant for training purposes typically takes 3+5+9 (process + control system + operation displays) person-months of work [HPK+96]. A way to reduce this cost is to use available digital design data about the modeling target. This requires solutions for typical information management systems identified [KF95] as

> *Redundancy*: Too much information and same information distributed in many places.
> *Islands of information*: Information exchange between distributed storage systems is poor.
> *Black box problem*: Information extracted from the systems is not in usable format and needs to be re-inserted manually.

These are issues of poor interoperability between different engineering systems. A fundamental cause of poor interoperability is the highly subjective semantics of the data [He95]. Increasing the interoperability between the design information systems can substantially cut costs for both the modeling and maintenance of the models. NIST reports [Na04a][Na04b] estimate the cost of inadequate interoperability in the U.S. capital facilities industry to be $15.8 billion per year in 2002 and the cost of inadequate supply chain integration to be in excess of $5 billion for the U.S. automotive industry in 2004. Improved interoperability between information systems and modeling tools offers increased benefits for the modeling user. Such benefits include possibility for *multi-domain* modeling in which different modeling domains are combined to acquire a more holistic view of the system. Benefits are also available in *multi-scale* modeling. Physical simulation can be characterized by the length-and time scale of simulated phenomena

> *Macro-scale* simulations are usually economical and strategic simulations, which are very hard to model robustly.
> *Meso-scale* is the typical scale for industrial applications.

> *Micro-scale* simulation can be used to discover basic properties of materials and some behavioral laws for the meso-scale models.

Combining models and simulations at different scales robustly has been the holy grail of physical modeling. This would allow modeling and simulation truly based on *first principles of physics*. An important goal would be to use some coarse-graining formalisms to generate meso-scale models from micro-scale models. These coarse-grained models could be used in large-scale meso-scale models and the micro-scale models could be coupled to the meso-scale model to provide possibilities to make more precise local analyses. CSC [Pu03] has reviewed the state of multi-scale modeling in Finnish industry.

> *This work approaches interoperability problems by introducing an integration solution. The proposed solution is an ontology-based information model, which can represent and connect information from different information systems.*

Key issues in this approach are the *data model* and the *connection mechanism*. To be able to perform integration the data model must be expressive enough. To be able to connect data from different sources some well-defined semantics are needed. The amount of connections must also be kept to a minimum to keep the system maintainable. Both these issues can be addressed by *standards*. Various international and national standardisation bodies such as ISO or SFS have published numerous standards, which have been applied to various degrees in engineering. In some domains a de-facto standard has emerged and in some fields competing standards exist. The amount of agreed semantics varies depending on the structure of the standards. Often the standards only define vocabularies or *taxonomies i.e.* hierarchical classifications and to interpret the semantics of the defined concepts requires domain-specific and/or even *common sense* knowledge about the subject matter.

An effective integration platform should be able to support various relevant systems and tools. The field of modeling and simulation is highly heterogenic with several model categorizations such as

Does the model behavior depend on time?
Does the model configuration depend on time?
Is time continuous or discrete?
Is the model deterministic or stochastic?
Is the model physical or empirical?

This work will not go into the details of different modeling domains but rather takes a higher-level approach considering *models* and *algorithms.* Here the models are collections of data that can be translated as an input for an *algorithm* that produces answers to *questions about the model* as outputs. Interesting algorithms for this work include

Simulation, which uses a model to imitate real-life phenomena. Traditionally mathematical models have been used. Simulation can be used to explain and predict the behavior of the real system.
Optimization which searches for best configuration for the model given some criteria. Optimization can be used on top of simulation to search for best solutions.
Verification and validation, which involves e.g. checking, model configurations against formal specifications and running tests with the models.
Reporting which extracts answers to questions about the model.

A key element of this work is to study the interface between the models and the algorithms.

As opposed to traditional knowledge management systems, a simulation environment needs to deal with real time data requirements. The simulation algorithms consume and

produce in real time large amounts of data which needs to be stored or processed. An even more challenging task is the integration of other real time systems such as sensors and measurements into the simulation system. Such integration requires advanced algorithms and data processing capabilities of the environment.

2.5 The state of the art

To continue the analysis of costs and benefits of simulation a set of fundamental modeling concepts need to be discussed. Modern engineering tasks include modeling increasingly large and complex systems. An important constraint in all human intellectual activities is our limited ability to handle complexity and the so-called working memory [BH74], which can only hold a small number of information at a time. Powerful methods for handling complexity are *simplification* and *abstraction*. With abstraction, the complexity of a model of an engineering concept can be reduced to manageable level. Since usual engineering models are inherently complex, the complexity should and cannot be abstracted away, but can be taken into account by applying *hierarchical decomposition* of abstractions. Hierarchical decomposition has been a dominant solution for handling model complexity. Simulation models are usually based on robust and rigorously studied theories such as first principles of physics or mathematics. Often these models are too complex to handle and simplifications are needed. Some mathematical simplification methods e.g. discretization and function approximation give somewhat precise information about the loss of expressive power due to the simplification. Sometimes a model needs to be constructed solely from some experimental data. In some cases, only a stochastic model can be constructed. Typical mathematical simplification is the fitting of a data set into a parameterized mapping e.g. a polynomial.

> *Most successful large-scale engineering tools use some form of abstraction and hierarchical decomposition. This provides a common ground useful in integration.*

Abstraction and hierarchical decomposition are potential tools for a modeler but they also introduce a critical caveat. The simplifications implied by the abstractions limit the expressive power of the model. In fact, usually the engineering models can only partially express the properties of the modeled system even in the designed conditions.

In unintended conditions, the models can lose practically all their resemblance to their real-world counterpart. A major problem in simulation algorithms is *garbage in - garbage out* i.e. most simulators will happily produce results from any unsuitable data but unfortunately, the results will be completely *wrong*. This is a major problem in easy to use automatic model generation, which can lead to *computer-aided mistakes*. Therefore, the process of *verification and validation* of the models becomes a deciding issue. Verification through measurements and comparison or using analytical methods is tedious work and requires wide expertise. If simulation models can be augmented with some design criteria such as designed input ranges, formal validation i.e. automatic processing could be used. If the simulation models include data from measurements such as sensors, the problems of validation and verification become even more apparent. A robust treatment of the problem would include introduction of a measure of *uncertainty* in the results.

An important factor in the cost of modeling is the efficiency of model *reuse*. Model reuse has strong relation to abstraction and hierarchical decomposition. A successful reuse mechanism is the introduction of *repositories of libraries* of model *templates*. A good abstraction usually serves as a good template and with hierarchical decomposition a single template can include a large model.

For the formulation and usage of models in a computer environment, the issues of *human-computer interaction* (HCI) need to be addressed. Especially the impacts of *usability* are essential. Key aspects of usability require the system to be accessible and productive for users at different phases of the learning curve. As with human interactions, the first impression is often crucial in human-computer interactions. An often-stated assumption is that a novice user should be provided with a graphical user interface and the experienced user is more productive with more fundamental tools such as text input and keyboard commands. This is certainly true for many older graphical user interfaces. Nevertheless the human brain has a trait of pattern matching, which makes graphics ideal for describing the structure of different models. Dynamic behavior of a model can also be effectively visualized using graphics. Engineering models often have a physical counterpart, which can be visualized with graphics. From such visualizations often important properties can be visually identified rapidly and with ease. Emerging technologies such as virtual or augmented reality can add

important aspects to such visualizations. Examples of visualizations are included in Figure 1 and Figure 2.

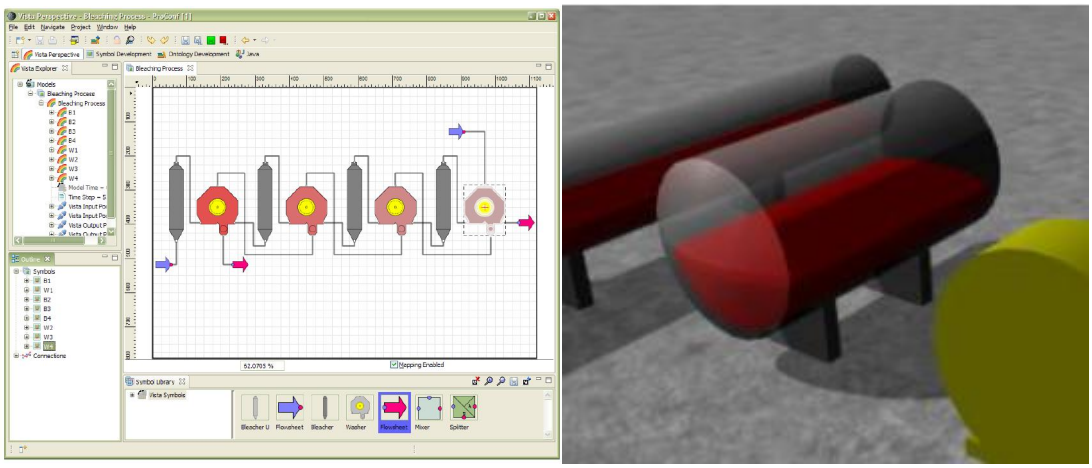

Figure 1: Process simulation editor uses colors to visualize simulated pH [Le07].
Figure 2: 3d model of a tank uses CSG to visualize liquid level [Lu07].

Another important aspect of usability is connected to the concepts of abstraction and hierarchical modeling. When designing a modeling and simulation environment for a broad group of users, an easy way is to use some generic mechanisms for information representation and supporting features. The goals and needs of the end users of the environment are on the other hand defined on high-level domain concepts. Taking the abstractions to the user interface level can prove to be essential for user acceptance and productivity.

2.6 The environment of industrial modeling and simulation

A high-level user categorization based on [Ka02] is used in this work for user analysis.

> *Kernel developers* are seasoned experts on modeling and simulation and are responsible for developing the modeling concepts and algorithms. Kernel developers can be found in universities, research institutes, and research departments of companies.
>
> *Library developers* are domain experts working on creating libraries of modeling constructs, which can be used to build models. Such constructs could include models for specific products.

Model configurators are engineers who are actually developing models from the definitions created by library developers and kernel developers.

Model users are users, who are using some existing models to acquire information using simulations or other types of analysis.

Key factors in the adoption of modeling and simulation are the issues of *knowledge creation and transfer*. In traditional information management systems, the information content provided by the design engineers has been data describing some final product of design. While this data is sufficient for building and operating a plant, a wealth of hidden or *tacit* information of the model exist only in the mind of the engineer. This hidden information would be valuable in later stages of the model development e.g. retrofitting and optimizing and for new projects based on the design. If the engineer leaves the company, much of the value of the model is lost. Nonaka and Takeuchi [NT95] discuss the issues of tacit information, its creation and transfer. The information system can address the issues of tacit information by promoting its storage and dissemination. The task is difficult and involves sociality more than technology. [Me04]

The rapid progress of *globalization* and the introduction of *digital network economy* [LHV+01] are transforming industrial modeling and simulation. In many cases, an industrial development project involves subcontracting from an international group of companies. The challenges of such networks have been studied in a recent Tekes program [RST+06] and outlined further in a Tekes technology review [Ke06a]. The studies emphasize the need for semantic information sharing using service-oriented technologies. Another major issue in industrial information management is the *life cycle* of modeling projects such as industrial plants. Large amounts of highly varying information is produced and consumed in different phases of the project life cycle by different organizations. In many cases, some of the information needs to be reproduced during the life cycle because of poor interoperability or lack of information management.

This work addresses the life cycle issues in digital network economy by adopting a common information repository.

The repository approach [BD94] includes a database of engineered artifacts with emphasis on *services* for control and management required to perform tool integration. The common information store approach to information management is also used in a series of publications from VTT [KKM+01][KKL04][Ka02]. These information models include a *typing system* based on *hierarchical structural modeling*. The repository approach brings about questions of *concurrent use*, *access control*, *version control* and *auditing*. These issues are sources for some of the most important design challenges analyzed in this work and many contributions in this work are based on the ideas developed for these VTT specifications. Others have also presented similar ideas [SHT+05][BMM+06][MN03]. The common information repository model is adopted in many commercial *data warehouse* solutions. The critics of the common information model argue that in the distributed and dynamic world, more distributed approaches such as Semantic Web or Multi-Agent Systems (MAS) should be employed. This work acknowledges these issues and searches for suitable solutions.

This work concentrates mainly on the various applications of modeling and simulation in process industry. There are ongoing projects to develop the environment into an integration platform between different simulation domains at VTT. These domains include

Process modeling and simulation in e.g. power production and paper industry
Industrial vehicle dynamics using rigid-body dynamics
Construction and community technology performing simulation on internal conditions of houses and district heating
System dynamics, which can be used to model business processes

Such integration requires multi-domain and multi-scale modeling capabilities and would need to cover multiple international standards and simulation and engineering tools.

To conclude this introduction the NSF report on SBES lists the following core topics of challenges, barriers and opportunities for simulation-based engineering and science.

Multi-scale modeling and simulation
Verification, validation and uncertainty quantification
Dynamic simulation systems, sensors, measurements and heterogeneous simulations
New simulation software
Big data and visualization
Next-generation algorithms

This work will concentrate on most of these topics. The major trend of high performance computing (HPC) around the world [HP07] has been intentionally left out of the scope of the modeling and simulation environment at this stage. This work will also not go into the details of simulation algorithms.

2.7 Simulation-aided process engineering

2.7.1 Process industry overview

Process industry is a large and traditional branch of capital facility industry. The largest branches of process industry in Finland include chemical and specialty chemicals, fine chemicals and pharmaceuticals, oil refining and petrochemicals, food and drink, pulp and paper, metallurgical, plastics and polymer and energy industries [KT01]. In general, the Finnish and European process industry is a very mature industry. Globalization and high capital investment required in building new production plants has led to a situation in which few new process plants are built. Instead, old plants are retrofitted to adapt to the changing marketplace. In a mature industry, the operating profits are tight and small process improvements can mean substantial increases in profits. The emerging technologies such as biotechnology also promise to provide additional growth for the process industry in the future.

2.7.2 Process modeling overview

The modeling of a process plant is a multi-discipline task. Different tasks include process, automation, electrical and structural design, business modeling (e.g. costing and scheduling) and modeling of operations and maintenance. For most of these tasks separate computer-aided process engineering (CAPE) tools are used. Some tools come from software vendors and some are in-house developments. The importance of CAPE in all stages of the life cycle of a process has been widely recognized, but the vision of CAPE support throughout the life cycle is far from realization [Ko01][MS98a]. In modern process industry investment projects, a large project consortium is needed to supply the different parts of design, material, construction and maintenance of a plant. In the global economy, these different sub-contractors can be geographically dispersed so that project management and communication become key factors in a successful project. A process plant is often a multi-billion € investment and its lifetime often spans multiple decades. From the modeling point of view the process plant life cycle can be divided roughly into four stages [KT01]:

Research stage is carried out before the actual plant design. In this stage models are developed.

Conceptual design is the stage in which the processes of the plant are designed as unit operation flowsheets.

Detailed design is the activity of making the detailed engineering decisions such as the actual equipment selection and automation design.

Operation and maintenance stage is the activity of operating the plant and keeping it running with optimal performance in different operating situations.

Other important stages include *operator training*, *de-commissioning* and *construction*. During the long life cycle of a process plant the processes also need to be constantly redesigned to respond to new business challenges and opportunities. This involves the use of various information models for different modeling and operating tasks [JSS04]. Inadequate interoperability, integration and lack of management may cause relevant information to be lost during the plant life cycle as depicted in Figure 3. Main goals of this work include the integration, interoperability and evolution of these models.

Modeling of process facilities involves a wide variety of standards for many different disciplines of industrial design. Relevant standardization bodies in Finland include SFS (ISO), SESKO (IEC) and PSK [PS04]. International standards for managing the life cycle information of process plants are available, but there is currently no standard solution for the integration and interoperability problem [SP06].

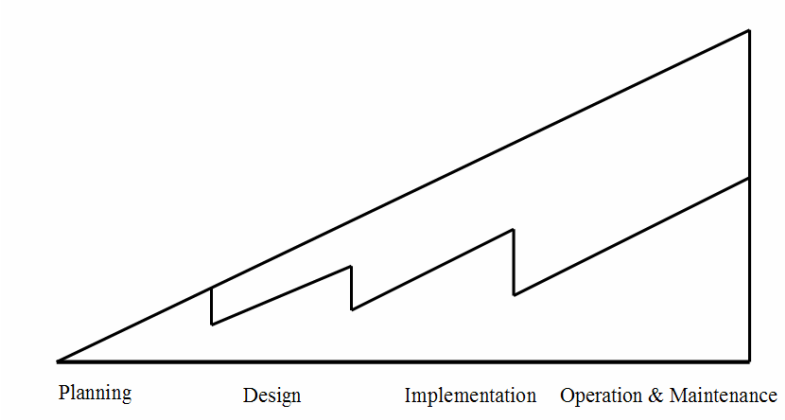

Figure 3: The amount of modeling data gathered during plant life cycle.

Process systems engineering (PSE) is a science providing tools for the process industry to *design* and *operate* chemical processes effectively and efficiently [KT01]. Main areas of PSE include [GW00][HC01b]

> *Process design*, which involves design of industrial processes e.g. selection of process equipment and piping
> *Process control*, which involves design of process plant automation e.g. controlling individual equipment to react desired operating setpoints
> *Process operations*, which involves scheduling, planning, diagnosis, optimization, and training of process plant operations

The goal of PSE is to optimize the *return on investment* for a plant. This includes design, construction, operational and decommissioning costs as well as operating profits. Also regulatory and political issues such as environmental, health, safety, quality (EHSQ) issues are increasingly important. PSE technologies can provide

decision support for humans throughout the plant life cycle and ultimately to provide [Ko01]

> *An optimal designed safe and reliable plant, operated hands-off at the most economical conditions.*

A basic method in PSE is the mathematical modeling of processes and their study. Prominent fields of PSE include [BG02]

> *Process integration*, which involves general, holistic system-level planning, usage and management of industrial processes. Methods in process integration involve *optimization*, *modeling of business processes* and *pinch-technologies* considering the use of energy in process streams [Sö05]
> *Process intensification*, which involves discovering novel unit operations or microsystems that integrate several functions and that can potentially reduce the cost and complexity of process systems
> *Process synthesis*, which involves automatic generation of designs for given design criteria

The challenges of PSE in the future [GW00] involve more complex models and methods with increasing integration between modeling domains and scales. Such developments enable holistic decision support ranging from prediction to syntesis and optimization in the business level.

Modeling of processes has traditionally involved the production of design documents including

> *Process flow diagrams (PFD)*, commonly used in process engineering to indicate the general flow of plant processes and equipment. The PFD displays the relationship between

major equipment of a plant facility and does not show minor details such as piping details and designations.

Process and instrumentation diagrams (P&ID), which are schematic diagrams showing the interconnection of process equipment, and the instrumentation used to control the process. For notation in P&ID diagrams several standards such as ISA S5.1 and DIN standards exist.

3d models, in which the geometry of the plant with all relevant piping, equipment and structures can be modeled. The tools are based on specification standards for different equipment and contain enough information to generate other diagrams such as P&ID. From these diagrams also lists of equipment and materials can be generated.

The state of modeling and simulation in Finnish universities and industry has been reviewed by CSC [HJL00] and Tekes [HHJ+01] respectively. The most frequently used modeling methods included differential equations, simulations, optimization and statistical methods. Needs were also identified in soft computing, visual modeling, model identification and economic modeling. The focus of this work will be on simulation, optimization and visual modeling.

2.7.3 Process simulation overview

The traditional application of process simulation has been the meso-scale or macro-scale calculation of important process characteristics such as pressures, temperatures, phase fractions and concentrations of flows and storages of fluids, which can contain concentrations of numerous *reactive* substances. The traditional way of modeling has been the configuration of interconnected *unit operations* into a *flowsheet.* Examples of flowsheets are given in Figure 4 and Figure 5.

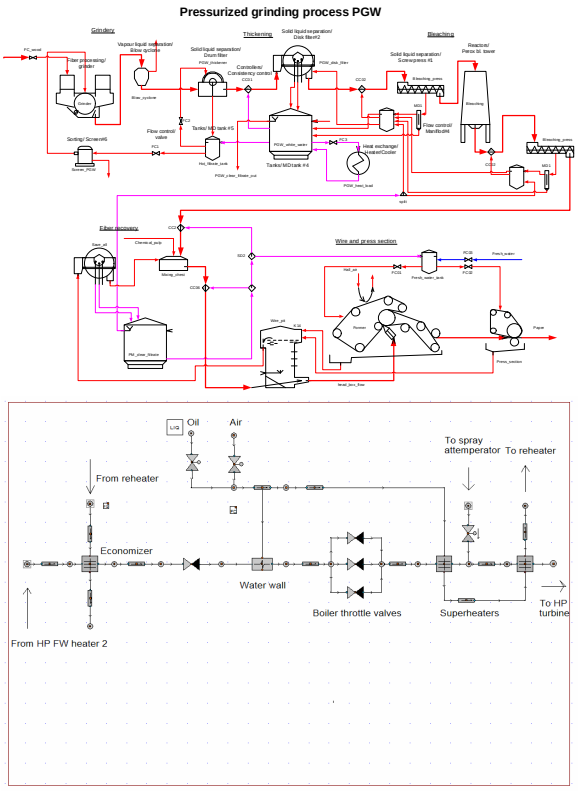

Figure 4: Modeling with BALAS steady-state process simulator [VTb].
Figure 5: Modeling with APROS dynamic process simulator [VTa].

Process simulation can be conducted on different levels of detail. The basic schemes are in the increasing order of detail

Steady-state simulation, which involves calculation of the balance, conditions in a one-dimensional process flow network

Dynamic simulation, which models time-dependent behavior in a one-dimensional process flow network

Computational Fluid Dynamics (CFD) which models time-dependent fluid flow in three-dimensional process equipment. The CFD solution is based on the Navier-Stokes differential equations in discretized 3d-volumes. The dominant solution method for CFD has been the finite volume method (FVM), but packages for the finite element method (FEM) have also appeared.

The most used one-dimensional solution approaches can further be defined as [BG02]

Sequential-modular approach. In this approach, each unit operation is represented by a set of equations grouped into a block (or module) and the whole flowsheet is solved on a module-by-module basis (in a sequential way). This approach is most popular among steady-state solvers.

Equation-oriented approach. The main idea of this approach is to collect all the equations and solve them as a large system of non-linear algebraic equations. Such an approach is popular in dynamic solvers.

Steady-state simulation has been widely used in the conceptual design phase of process plant life cycle. Typical applications include [VTb]

Calculation of mass and energy balances. The outputs and intermediate values of a simulation model are calculated given some inputs and parameters.

Analysis of heat integration and heat recovery. Different possibilities of heat reuse in a process can be explored and simulated.

"What if" analysis. With a simulation model different abnormal or interesting situations can be quickly analyzed.

Process optimization. An optimizer can be used on top of the simulator to optimize given objective function by modifying given parameters.

Dynamic simulation is needed, when transient behavior of the plant and its control system needs to be modeled. Such situations include *power up* and *power down* of processes, *grade changes* of continuous processes and simulation of error conditions such as *breakages*. Important use cases for dynamic simulators include [PAT05][HPK+96][KP06]

Process design especially involving dynamic transient considerations

Control and automation design and testing in which the control system of a process is designed or tested using simulated process and control. [CP04]

Operator training and support in which operators of a simulated plant can be trained to handle typical or abnormal situations in plant operation. The simulator can be connected to live plant measurements and the simulator can be used to answer questions about the near future behavior of the plant operation. Such simulators are called *tracking or on-line simulators*. [OHP02]

Safety analyses of transient behaviour during abnormal conditions such as emergency shutdown or equipment breakages. [PKK+01][KP03][KP04]

If the resolution of the one-dimensional approaches is not sufficient, the calculations can be done using CFD. The CFD calculations are many orders of magnitude slower, but provide accurate results when used in

Detailed studies of fluid behavior in equipment. In some equipment with complex geometry or small-scale dynamics the one-dimensional study is not accurate enough.

Turbulence studies. Turbulence is a major factor in fluid flow and can only be accurately modeled with 3d simulation.

In addition to the first-principles equations of *Navier and Stokes*, some experimental correlations and models are needed to express complex behavior of e.g. a grinder in a pulp mill. These can be introduced using black-box modules or in some equation-based systems by adding new custom equations.

The calculation of chemical reactions in such flow networks is not an easy task. Available packages consisting of large databases of material properties contain algorithms to minimize the free energy to obtain thermo-chemical equilibrium [ES95].

Lighter, more qualitative methods also exist. For non-equilibrium reactions such as pulp bleaching codes exist and new ones are constantly developed.

Another important component of such solvers is the calculation of material properties such as partial phase-diagrams of water depicted in Figure 6. Often these properties are calculated from polynomial or exponential fits of large amounts of experimental data [LH99]. Some material properties can also be modeled and simulated using micro-scale molecular models.

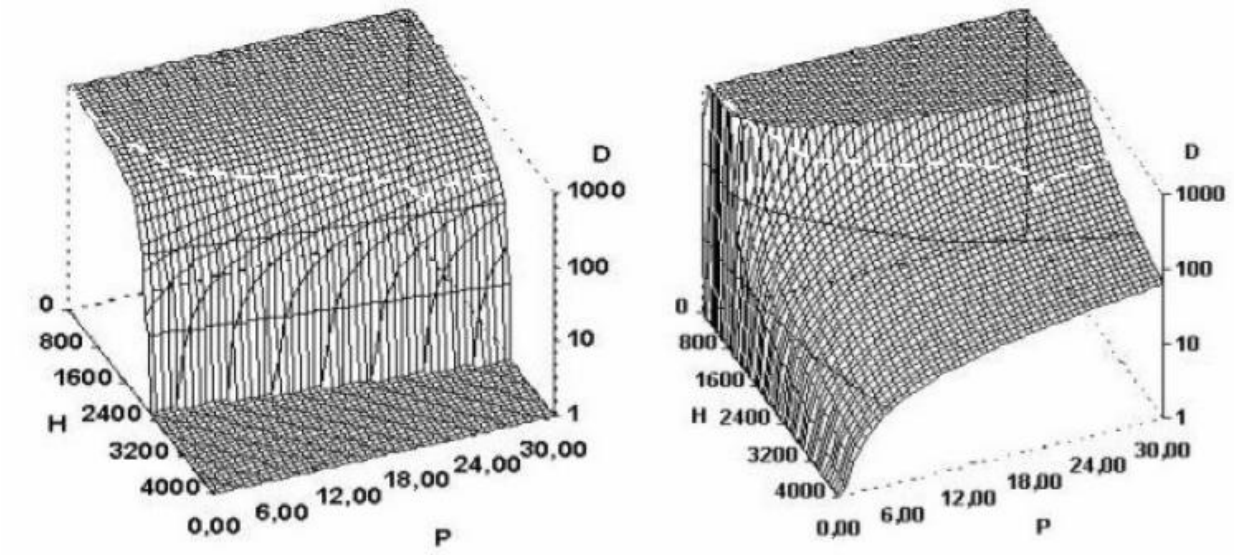

Figure 6: Fittings of density of water as a function of pressure and specific enthalpy [Ju05].

In dynamic process simulators, the control system of the process can also be simulated. The control system can be modeled using the flowsheet methodology by reading inputs from unit operations e.g. tanks or flows and writing control values into appropriate unit operations e.g. valves. The control system hardware or virtual control system running at PC (DCS/PLC) can also be connected straight to the simulator in order to carry out e.g. control system testing or operator training [LTK+99].

2.7.4 Stakeholders in process modeling and simulation

A quick overview of stakeholders in process modeling and simulation further clarifies the application area of this work.

Equipment manufacturers provide the equipment for the plants. Modeling of such equipment requires collaboration with the manufacturers on many levels.

Engineering and consulting companies offer services for design of process plants. Consulting companies use process modeling tools regularly but process simulation is more seldom.

Universities and research institutes develop methods for design and simulation of processes. These groups often do simulation studies.

System providers provide complete subsystems for the plant such as the automation system.

Facility owners control the investment projects and specify the requirements related to modeling and simulation in design and operations

Operators control the operation of a process plant using various decision support systems.

Maintenance companies perform maintenance of the plant equipment. The process of maintenance can benefit greatly from designs and models of the plant.

3 Technology review

3.1 Introduction

A proper way to initiate the study of design challenges of a simulation and modeling platform is to review existing technologies. A technology review is useful in answering questions such as

What can be done with existing modeling and simulation technologies?
What could be improved in existing modeling and simulation technologies?
What are the services and features required of such an environment?
What existing technologies could be used to achieve the requirements?

These questions will be discussed at the end of this chapter. The scope of this study does not allow for a comprehensive treatment of technologies related to modeling and simulation. The structure of the review is based on the following categorization into

Fundamental technologies constituting the foundation of the modeling and simulation environment. These technologies address the issues of architecture and knowledge representation.
Supporting technologies enabling efficient modeling and simulation in the environment. These technologies address the issues of user interface and decision support.
Integration technologies needed to perform integration of existing information and tools. These technologies address standard means for integrating systems and information.
Identified systems and tools to be integrated into the environment. The project group at VTT has made a strategic selection of key systems and tools, which will be considered

for integration into the modeling and simulation environment.

The presented technologies have been carefully selected to present only the most relevant technologies with respect to this work. The selection criteria includes

The vision and strategy of modeling and simulation developed at VTT based on accumulated experience on customer needs and global trends of SBES.
The state of the art and global trends perceived in the different areas of interest with respect to this work

The scope of the selected technologies ranges from fundamental architectural technologies to domain-specific standards and tools. This reflects the delicate balance required between the generic environment approach and the specific goals and needs of the engineering users of the system. The selection of technologies also ranges from immediate requirements to more advanced strategic technologies for it is important to both address the current needs of the users and to plan. The scopes and timeframes of selected technologies are listed in Table 1.

Technology	Scope	Timeframe
Architectural technologies	Fundamental architecture	Required initially
Persistent data storage technologies	Fundamental architecture	Required initially
Knowledge representation technologies	Fundamental representation	Required initially
Upper ontologies	Fundamental representation	Required initially if applicable
Version control	Fundamental services	Required eventually
Security technologies	Fundamental services	Required eventually
Existing ontology development environments	Fundamental architecture	Required initially if applicable
User interfaces	Fundamental to supportive services	Frameworks required initially

Knowledge management (KM) and artificial intelligence (AI)	Supportive services	Strategic
Verification and Validation (V&V)	Fundamental or supportive services	Frameworks required initially
Collaboration	Supportive services	Strategic medium-term
Integration architectures	Simulation service architecture	Required initially for reference. Strategic adoption.
Ontologies and standards	Industrial, mostly process modeling	Strategic short-term
Mapping	Fundamental services	Frameworks required initially
Communication	Fundamental external connectivity	Strategic short-term
MATLAB/Simulink	General modeling	Strategic medium-term
Modelica	General modeling	Strategic short-term
Companion model	General physical modeling	Strategic medium-term
System Dynamics	Business modeling	Strategic medium-term
Phase field methods	Micro-scale modeling	Strategic long-term
Finite elements and CFD	Process modeling	Strategic short-term
Community technology	Domain-specific	Strategic medium-term
APROS	Process modeling	Strategic short-term
BALAS	Process modeling	Strategic short-term
Thermo-chemical simulators	Process modeling	Strategic medium-term
Plant design systems and integration platforms	Process modeling	Strategic medium-term

Table 1: Scope and time frame for the technology review.

The selected categorization is based on the architectural structure of the modelling and simulation environment instead of technology categories. Therefore some technologies can be relevant for many categories. E.g. knowledge management technologies are needed for the fundamental data model and also in many extensions built upon the fundamental architecture.

The first section of the review is a brief introduction to semantic modeling with ontologies.

3.2 Semantic modeling with ontologies

> *"The quality of life in an "information society" will not depend on the quality of its technical networking facilities but on its ability to use and adapt these facilities to the real social needs. Not the amount of information available by such networks will be decisive but its quality, its relevance to the urgent questions of human beings and its usefulness for solving inherent problems of individuals, organisations and human societies" [Fa98]*

The topic of this work is an *ontology*-based modeling and simulation environment. Thus, the use of the word ontology requires definition. The original context of the word is philosophy where it stands for the *study of existence*. This philosophical background has created confusion and controversy since the profound issues of being are not currently manageable in computer science. Some philosophical background and its implications to modern knowledge management can be found in [Sm03]. The knowledge management view of ontology is more pragmatic the most often cited definition being

> *An ontology is an explicit specification of a conceptualization [Gr93]*

This broad definition covers many different approaches such as [UG96][Gr93] *vocabularies*, which define a common language with possibility for agreed semantics, *data model schemas*, which specify and constrain the structure on the data model, *taxonomies and classifications*, which can range from a simple hierarchy of concept names to systems with means for performing complex classification of individual data items.

Often-cited papers by Uschold [UG96] and Roche [Ro03] suggest important benefits of using ontologies. Ontologies enable *communication* between *people* with different needs and viewpoints based on *shared understanding*. Analogically a shared conceptualization enables *interoperability* among *systems* by translating between different modeling methods, paradigms, languages and software tools. *System engineering benefits* include *reusability* based on agreed and formal specification, *reliability* based on automatic inferences such as consistency checking and *specification* aided by the common understanding especially when the requirements involve different groups using different terminology in the same domain, or multiple domains. The specification of conceptualization also facilitates *knowledge management* services such as information retrieval and processing. The seminal paper by Gruber [Gr93] points out the *specification* of the conceptualization based on *formal axioms* is essential in defining the possible interpretations of the defined concepts. While informally agreed conceptualizations such as many international engineering standards are sufficient for humans, only formal specifications can be utilized by knowledge management systems.

Uschold [UG96] gives a useful classification of ontologies based on their *formality* of expression. *Highly informal* ontologies are expressed loosely in natural language while *semi-informal* ontologies are expressed in some restricted and structured form of natural language. *Semi-formal* ontologies are expressed in an artificial formally defined language while *rigorously formal* ontologies include also formal semantics, theorems and proofs of such properties as soundness and completeness. There is a balance between informality and formality needed to address the needs of humans and computers. Increasing formality increases clarity, reduces ambiguity and makes automated reasoning possible. It also reduces the expressive power so that specifications that are more complex quickly become large and inaccessible for human users. Many informally simple assertions cannot be specified formally at all. In many ways informal specifications deal with *information* while the formal specifications deal with *data*.

Another relevant classification is based on the scope of the ontology given by Roche [Ro03]. *Meta-* (or representation) *ontologies* specify the knowledge representation principles used to define concepts of domain and generic ontologies. Meta ontological

concepts include *classes*, *relations*, and other sufficiently formal constructs which can also be layered [HL05b]. A *generic* (or upper) *ontology* specifies general concepts defined independently of a domain of application and which can be used in different application domains. *Time*, *space* and *mathematics* are examples of such general concepts. A *domain ontology* is dedicated to a particular domain but remains generic for that domain such that it can be used and reused for particular tasks in the domain. An *application ontology* gathers more specialized expert knowledge about a particular application or task. Application ontologies are, in general, not reusable.

The design of such semantic ontologies requires expert *knowledge* if not *wisdom*, which cannot be easily captured in a textbook. Gruber [Gr95] suggests some design criteria for ontologies. The criterion of *clarity* requires that an ontology should effectively communicate the intended meaning of the defined terms. Definitions should be *objective* and independent of social or computational context. Gruber suggests formalism such as mathematical logic with necessary and sufficient conditions as a means to this end. All definitions should also be documented with natural language. The *coherence* criterion states that ontology should sanction inferences that are consistent with the definitions. At the least, the defining axioms should be formally consistent. Coherence should also apply to the concepts that are defined informally. The criterion of *extensibility* asserts that the ontology should provide a conceptual foundation for a range of anticipated tasks, and the representation should be crafted so that one can extend and specialize the ontology (logically) monotonically. The ontology should also not depend on a particular symbol-level encoding. The criterion of *minimal ontological commitment* requires an ontology to only provide definitions sufficient to support the intended knowledge sharing activities i.e. to make as few claims as possible about the world being modeled. The ontological approach can also be applied to the design of ontologies. The OntoClean methodology [GW04] is a collection of central ontological concepts and associated best practices encountered in ontology designs. Other methodologies for ontology development are surveyed in [CFG03]. A comprehensive introduction to ontology with recommendations for ontology design can also be found in [Sp06]. In many cases the design of ontology is closely related object-oriented information modeling which is covered in numerous textbooks.

Another important aspect of modeling with ontologies is ontology evolution. The evolution of ontologies is close to database schema evolution, which has been studied extensively. The evolution of ontologies can be categorized as tracked and untracked. For a tracked evolution process a log of operations is available, which can be used to analyse the effects on dependencies such as instances and related ontologies. In untracked evolution no such information exist and some automatic or semi-automatic means are needed to discover the differences between versions. [NK04]

The engineering applications of ontologies relevant to this work include interoperability between systems based on shared conceptualizations and information model *mappings.* Such mappings can be specified between the conceptualizations or between some models based on the conceptualizations. The mapping of models can be manually specified in the ontologies or some techniques for automatic discovery can be used. The mapping mechanisms are further discussed later in the review. The semantics attached to the ontologies can also be utilized in automated decision support. *Identification* of conditions can be used in higher-level analysis and reporting. The conditions can range from simple pattern matching (e.g. pipes with pressures higher than design limits) to high level business parameters (e.g. estimated process interruptions due to maintenance breaks is high with the given design). *Model construction* can be based on semantic modeling knowledge. A simple case of model construction is a template approach, where a model is generated according to given parameters. More advanced services would include automatic design of models based on given specifications. Such synthesis could combine reasoning with simulation and optimization. *Model and ontology verification and validation* can be based on semantic formal specifications such as conditions and rules. A simple validator checks some syntactic constraints imposed on a model (e.g. model of a given type of a car includes exactly four wheels) or an ontology (e.g. constraints defined in the ontology are not contradictory). More high-level validation targets include design constraints e.g. rules and equations. The process of verification through automatic testing can also benefit from semantic configurations.

An active field of ontology research is concentrated around the Semantic Web [BHL01], which is a proposed successor to the World Wide Web (WWW). The purpose of the Semantic Web is to extend the current Web such that the overwhelming

sea of information could be accessed in a semantic machine-interpretable form. The Semantic Web has taken an ontology-based approach with the main standard OWL-DL (OWL), which has robust model theoretic semantics with capability for effective reasoning based on *description logic*. The Semantic Web technologies have been pioneered in medical research, but many other domains are showing increasing interest. The applications of Semantic Web are currenty at the research phase, but significant efforts such as the national FinnOnto [Hy06] are on the way to gather a critical mass needed for wide-spread adoption.

3.3 Fundamental technologies

For an extensible software platform, the most fundamental architectural and technological choices will play a dominant role in determining the applicability of the software to different use cases. Fundamental technologies related to architecture and knowledge representation are selected to address most important strategic issues related to the *vision* of this work including

Multiple users working in a distributed manner including engineers using graphical user interfaces, query services and intelligent agents

Shared semantic information store with powerful knowledge representation and management technologies

Extensible application platform providing means for introducing and integrating new tools and services for SBES and CAE

Other strategic technologies introduced in the vision are later discussed as supporting technologies including

Advanced user interface with context-dependence and visualization

External connectivity to other possibly real time information systems

Decision support technologies such as knowledge management and verification and validation

3.3.1 Architectural technologies

Bosch [Bo00] gives guidelines for architectural design for large software systems. The scope of this study will be on the initial steps of *defining the system context* and identifying the *archetypes* i.e. core abstractions. A powerful tool for architectural design is *patterns*. Basic patterns for architectural design are defined in [BMR+96]. From the application point of view, large parts of the modeling and simulation infrastructure can be seen as *middleware*. Useful patterns for designing middleware can be found in [SSR+00].

The requirement of multiple distributed users was introduced in the vision. Distribution can be applied to the three basic resources in software development: *code*, *data*, and *hardware*. The selected distribution scheme will have its effect on *management* and *availability* of these resources. Fundamental approaches for distribution include *centralization*, which corresponds to the *client-server* architecture, which is one of the most successful architectures in computer science and the traditional way of implementing large multi-user software systems. The *de-centralized* approach corresponds to *peer-to-peer* technologies. A large distributed system will most likely exhibit features from both approaches along with optimizations such as *caching* and *replication* [CDK05].

From the middleware viewpoint several mature technologies exist. Szyperski [SGM+02] outlines *software component technology*

> *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."*

The key words in this description are *composition*, *interfaces* and *deployment*. The basic idea of software components is to establish a *marketplace of components*. Software component technology usually offers middleware for implementation of distributed components under the ownership of a single system. *Service-Oriented Architecture* (SOA) provides means for building complex heterogeneous systems. *The*

Organization for the advancement of Structured Information Standards (OASIS) defines SOA as

> *"Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains." [OA06]*

SOA provides uniform means to offer, discover and interact with such capabilities to produce desired effects consistent with measurable preconditions and expectations. A popular implementation of SOA is *Web services*, which provides capabilities described in the OASIS definition in the context of the *World Wide Web* [Er05]. Another approach for SOA is given in OSGi specifications

> *"The OSGi specifications define a standardized, component oriented, computing environment for networked services that is the foundation of an enhanced service oriented architecture." [OS07]*

The OSGi specifications implement SOA in the context of a single *Java Virtual Machine* (JVM). The *Eclipse* platform builds of the OSGi approach adding layers of functionality

> *"Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle." [Ec]*

Essentially the Eclipse platform includes strong frameworks for Graphical User Interface (GUI).

3.3.2 Persistent data storage technologies

The de facto persistent data storage solution is the *relational database* with a *Simple Query Language* (SQL) interface and an associated relational database management system (DBMS). Relational database technology is highly mature and modern implementations can handle huge data sets, answer queries quickly and guarantee

transactional *Atomicity, Consistency, Isolation and Durability* (ACID). The concept of a transaction has focal significance in persistent data management and the design of the modeling and simulation environment needs to address the issues of transaction implementation and long-lived and nested transactions [Gr81]. Relational databases typically do not support versioning or temporality. The requirements of industrial simulation and measurement data including acquisition and retrieval of time series data are also not well suited for popular relational databases. For such requirements, specialized implementations are available [WKL+00]. For complex associative data the tabular representation of relational databases is not optimal. In such cases the *navigational* access model of *object-oriented databases* is more efficient. This work focuses on information models motivated by the W3C Semantic Web standards. For such data specialized *triplet stores* are available [Fr][Ge][Or05]. Such stores usually provide specialized query features such as SPARQL but are not as robust as the mature relational databases.

Many engineering information solutions [KKL04][TH] are at least partly based on storing standard formatted files using the local file system of the data server [GUW02][SKS02].

3.3.3 Knowledge representation technologies

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 semistructured data, the structure-describing part is often called *metadata*. Forms of *unstructured data* are also needed in many cases. For example, plain text and pictures can contain lots of information for the system user. [Bu97] 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 [Ab97]. The W3C SPARQL [Wo07] query language is based on the ideas of SQL. The Open Knowledge Base Connectivity (OKBC) [CFF+98] 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*. Constrained FOLs such as Datalog [SKS02] have been used for queries in relational databases. The popular programming language Prolog [SS94] has been widely used in *expert systems* and is based on a Turing-complete subset of FOL. F-logic [KLW95] is an object-oriented syntax, which is translated into FOL for reasoning. Recent W3C standard OWL [Wo04a] uses description logics (DL) [BCM+03] which are a subset of first-order logic with effective reasoning mechanisms. General first-order logical representations include KIF [Ge91] and Common Logic (CL) [In07]. The proprietary CycL [Cy] uses extensions of first-order logic. The logical approaches can be categorized based on the treatment of unstated information. 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 [Wo04b] and OMG OCL [Ob03b], which is used in Unified Modeling Language (UML) -related standards. The Information Flow Framework (IFF) [Ke06b] uses the logic of information flow [BGH95] and category theory, which are even more general than FOL and set theory. The IFF also utilizes Formal Concept Analysis (FCA) [Pr05], which defines concepts based on attributes. The ISO STEP representation language EXPRESS [In94] 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. Many FOL-based representations use LISP-based syntax [HM01][Ge91][In07][MCW+06]. Other text serializations [Wo06][SS94][KLW95][In94] can be parsed using EBNF parsers. Processing of such data requires an internal representation, which usually is tree-structured such as LISP or XML or graph-structured such as OWL, Conceptual Graphs [So79] and Topic Maps [In02]. These structures can be represented in the relational way using statements about the edges of the graph. For such data, models additional fields for implementing fundamental features such as access control, data organization, temporality and

uncertainty have been suggested in [MK03]. The OMG specifications Meta-Object Facility (MOF) [Ob03a] and Ontology Definition Metamodel (ODM) [Ob06] can be used to represent ontologies accessible through popular UML tools.

3.3.4 Upper ontologies

The model-theoretic semantics are most commonly used to specify ontologies based on e.g. classifications. This ontological mechanism is then used to build layers of ontologies with increasingly specific scopes. For many domain or application ontologies, the informal semantics based on human information become more important than the model-theoretic semantics. Upper ontologies aim to bridge this semantic gap by defining some very general concepts with at least semi-formal specifications, which can be used to build domain and application ontologies. The communication, interoperability and system engineering benefits associated to ontologies are also maximized because of the wide scope of the upper ontology. Common constructs in upper ontologies include partition into *concrete* and *abstract*, *mathematical constructs, temporal concepts* and *aggregation* from parts. The most high-level ontologies such as BFO [Gr03], GFO (successor of GOL) [HHB+06], DOLCE [MBG+03] and ISO 15926 Part 2 [In03] contain only these highly abstract concepts but most ontologies such as Cyc [MCW+06], ISO 15926 Part 2 and SUO [NP01] have also more domain-specific extensions. Some upper ontologies such as ISO 15926 are specifically designed to meet the needs of some industrial modeling domain. The most ambitious of these ontologies is Cyc, which aspires to model all of human consensus knowledge i.e. common sense.

3.3.5 Version control

Version control is a mechanism to manage information by recording all changes to the data. Key services provided by popular version control systems are support for *concurrent work* and *persistent data storage* with backup, change management and auditing. The use of version control systems is a de facto best practice in software engineering. A client of a version control system can *check out* content from a *repository* for local processing. The client modifies the content by *committing* a *set of changes* into the loaded content. Multiple clients can potentially perform *concurrent modifications*, which is a potential source of *conflicts*. The prevailing methods for dealing with conflicts are *locking* and *merging*. Locking enforces mutual exclusion on checked out content and merging tries to algorithmically combine concurrent

modifications. Usual merging algorithms are not always successful and the remaining conflicts need to be resolved by the user.

3.3.6 Security technologies

Security technologies are needed to provide *trust* in business collaboration. Key issues of security include *confidentiality, privacy, authentication*, *authorization*, *integrity* and *accountability*. In the context of the modeling and simulation environment, the technologies for *secure data transfer* between actors and for *access control* into information models are essential. Technologies for *authentication* are well established with the *Public Key Infrastructure* (PKI) standards such as ITU-T X.509 [In05c]. Technologies for *encryption* are also well established with several *symmetric key* algorithms. Access control in semantic information models is a relatively new research area with no established standards. A basic goal of *access control* is to provide authentication and authorization services. Most traditional solutions utilize *hierarchical constructs* for managing the authorization specifications. In distributed semantic information models, such models are not sufficient since the models do not necessarily form hierarchies. The issues of semantic access control have been addressed in recent works of [Ka07][Na07].

3.3.7 Existing ontology development environments

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 Protégé [St], KAON2 [Ina], Jena [Je] exist for reading and writing ontologies, building ontologies, performing DL and SWRL [HPB+04] reasoning and SPARQL queries. The popular Protégé includes generic user interfaces for building models and an extensible plug-in architecture for adding features. The KAON2 platform also provides a stand-alone server providing access to ontologies in a distributed manner using Java Remote Method Invocation (RMI).

3.4 Supporting technologies

The following section reviews essential technologies identified from the vision and strategy of this work. The technologies are categorized as supporting because they will not be needed in all use cases of the system. The technologies related to user interface and communication are absolutely essential for the success of the environment. In the area of knowledge management and verification and validation there are strategic technologies which are currently not state of the art in engineering systems. The following categories are addressed by the review

User interface technologies used for manipulation and representation of information in the system.
Communication technologies used to make connections to external systems
Knowledge management and artificial intelligence (AI) technologies used to provide services for decision support
Verification and validation technologies needed to provide fast and reliable modeling and simulation.

3.4.1 User interfaces

Much of research and development around modeling and simulation still requires writing program code with a text editor or a graphical user interface. The quality of the code development environments is highly language-dependent. Modern languages such as Java or C# have powerful development environments such as Eclipse while tools available for the more traditional modeling and simulation languages such as FORTRAN and C/C++ are less powerful. For special-purpose languages aimed at engineering problems such as Matlab [HL05a] or Modelica [EM97] the development environments provide some additional features such as model building using graphical flowsheets and visualization of results. The user interfaces of simulator tools aimed at engineering use include flowsheeting and physical 3d modeling. The relevant aspects of such 2d and 3d graphics for this work have been covered in [Le07] and [Lu07] respectively. The main visualizations for industrial models include plotting, trending and animations. Trending involves graphical visualization of time series data from *simulations* and operational *measurements*. Animations can communicate important

model parameters to the user by associations to positioning, colors and dimensions [Ke04]. The overall state of the art in user interfaces and graphics can be found in modern user interfaces such as Windows Vista [Mic] and Mac OS X Leopard [Apc] and in latest computer games. In most cases, the user interfaces of modeling and simulation software are based on old technology. The issues of temporal models in user interfaces have been addressed in many media-related applications. A usual approach is to provide a normal non-temporal user interface augmented with a timeline for modifying the position in time the user interface is presenting [ACG+06]. The lifetimes of modeled objects can be manipulated in the timeline. A strong trend in user interfaces is the introduction of multimodal user interfaces such as the Apple IPhone [Apb] or Microsoft Surface [Mib]. User interfaces for mobile clients of the system are required when a dedicated user interface client is not available e.g. while traveling or when using mobile devices for maintenance. Many systems such as [TH] provide Web interfaces for such use cases. The applicability of augmented reality (AR) into industrial user interfaces has also been demonstrated in research projects [SKW+07]. For some applications existing standards-based user interface tools are available. An example in process industry is the OPC real time data standards for which powerful data management and visualization tools such as Webmon [KL05] can be used. Tools are also increasingly becoming available for manipulating OWL.

3.4.2 Knowledge management (KM) and artificial intelligence (AI)

The added information of the ontological approach makes available many technologies from *knowledge management* such as automatic reasoning or *artificial intelligence* (AI). At the time of writing large software suppliers such as Microsoft are investing heavily on AI technologies to produce better user experiences with intentional user interfaces. A strong programming paradigm associated to AI is *software agent technology*

> An agent is anything that can be viewed as perceiving its *environment* through *sensors* and acting upon that environment through *actuators*. [RN03]

Distinctive features of software agents are issues of *learning*, *autonomy* and *communication.* An agent can modify its behavior through learning from its percepts. A

common design principle for agents is *rationality*, which is a function of a performance measure, prior knowledge, percepts and possible actions [RN03]. Many Web technologies such as crawlers and bots are software agents. Agent systems have a close relation to SOA since they offer services and communicate with other service providers using formal protocols.

A fundamental technology of AI is a search in a graph of states. In most practical applications, the search is guided by some *heuristics*, which can be extracted from given constraints or approximations. Planning systems perform search on explicit propositional (or first-order) representations of states and actions for which effective heuristics and algorithms can be developed. Partial order planning can be used to produce plans without committing to a totally ordered sequence of actions. Planning schemes also involve scheduling i.e. planning in time, conditional planning based on execution-time information, incorrect information about the outcomes of actions, refinement of plans in execution time and collaborative planning. Engineering applications of search and planning in semantic data models include *optimization***,** which can be used to find better models from existing models and *model generation*, which can be used to produce a model given some constraints. The most discussed use of semantic data models includes *automated logical reasoning*. Reasoning can be done based on different logics such as *Horn logic* (if-then-else rules and Datalog, SWRL, RuleML [Bo]), *description logics* (OWL-DL), *first-order predicate logic* (CL), *modal logics* with operators for handling *modalities* such as possibility and time, *second-order logics* with quantification over predicates and *fuzzy logics* dealing with a continuous *degree of truth*. Reasoning services include computation of truth-values for a given sentence and search for variable bindings such that a query sentence becomes true. The traditional methods for treating logical uncertainty include *fuzzy logic*, *Bayesian networks* and *Dempster-Shafer theory*. Fuzzy logic has been especially popular in engineering systems. Recent developments also suggest possibilities for combining logical and probabilistic reasoning [La06][Ha05]. Logical treatment of temporality can be included in *Dynamic Bayesian networks* and *modal logics*.

Simulation, reasoning, searching and planning are ways to acquire answers to questions about a model. A *decision analyst* can use these methods for enumerating possible actions and their outcomes. In a decision support system, the *decision maker* is needed

to give preferences to outcomes. Automated decision-making is usually based on assignment of *utilities* to different decision outcome states. The decision maker seeks to maximize expected utility. Automated decision-making process must also address information retrieval actions for which utilities can be assigned. For continuous decision-making, optimal sequences of actions can be calculated but in practice problems are too large for such analysis. Often the outcomes of actions are not deterministic and *belief states* need to be considered. Agents can also co-operate to optimize some global utility or compete to maximize individual utilities. [RN03].

Learning can be used to increase the performance of decision support systems. Learning can be applied to parametric mappings, probabilistic mappings and logical sentences. Basic scenarios for learning are supervised learning from examples and unsupervised learning of structure in input data. The real-world case of reinforcement learning occurs when only indirect feedback is available for rating decisions. Selection of suitable learning mechanisms is difficult and involves a trade-off between accuracy and computational feasibility. Most learning methods require means for acquiring percepts or feedback. For live systems, this requires data from sensors.

For many engineering problems, a solution cannot be found by using a single formal method, but rather by using multiple heuristic methods capable of dealing with imprecision, uncertainty, partial truth and approximation. Such *soft computing* technologies include fuzzy computing, neural networks, evolutionary computing, machine learning and probabilistic reasoning. A Tekes technology program A Tekes technology programme report [BS00] considered soft computing methods to be an important source of competitiveness for the Finnish industry.

3.4.3 Verification and validation (V&V)

Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. Verification is the process of evaluating a system or component to determine whether the products of a given phase satisfy the conditions imposed at the start of that phase [IE90]. With these definitions validation can be seen as high-level and verification as low-level checking of model *quality*. In many engineering domains, the requirements of model validation are addressed by means of verification e.g. testing. While the

definition of validation covers complex subjective requirements, it can also be applied to objective formal specifications. In the context of the modeling and simulation environment relevant areas for V & V include

Development of ontologies according to principles outlined in the introduction to semantic modeling
Configuration of models of high quality using e.g. schematic validation
Verification of decision support algorithms e.g. simulation codes based on results and experimental or analytical data
Verification of modeling experiment results based on design constraints on inputs and internal states

The nature of faults in knowledge-based systems follows familiar patterns. Faults not detected in early phases become progressively more expensive to rectify in later phases [TVZ99]. A general framework for V & V consists of goal-based determination of a test suite consisting of static and dynamic methods. For software development in modeling and simulation comprehensive quality processes such as the CMMI [CKS03] by Software Engineering Institute (SEI) have been developed. The CMMI is a holistic model concerning *processes*, *people* and *technology*. Popular process models for V & V include the V-model [Fe]. Assuring total quality in a modeling and simulation effort involves the measurement and assessment of a variety of quality characteristics such as accuracy, execution efficiency, maintainability, portability, reusability, and usability (human-computer interface). In the context of ontologies, the schematic validation techniques are especially relevant. In schematic validation, a data model is checked against a *schema* or specification. Exemplar uses of schematic validation include database schemas (ER), structured documents (XML), logical schemas (OWL) and formal specification languages (Z [Sp89]). For many mathematical models, the modeling assumptions can be expressed with equations [HC01a]. According to Balchi [Ba97] the informal (i.e. subjective and heuristic) techniques for V&V are most commonly used for conventional simulation models. In addition to static analysis and dynamic testing, some formal techniques such as mathematical proof of correctness exist.

3.4.4 Collaboration

Efficient teamwork and collaboration requires communication methods between the human users of the system. Popular means for direct communication between computer systems include voice, video, desktop sharing, instant messaging and chat. Some of these technologies are available as standards such as IRC [OR93] but mostly the popular solutions are proprietary. Indirect communication methods include e-mail, discussion channels, forums, Wiki and blogs. Most of these technologies are based on standard or free software solutions. A strong trend in distributed communications is the *social media* phenomenon [KTB07]. A distributed information model can also be used as a media for communication as suggested by issue-based information systems (IBIS) [KR70], where problem topics, issues, questions of fact, positions, arguments and model

3.5 Integration and interoperability technologies

The vision of the modeling and simulation environment places emphasis on integration and interoperability between existing systems. Therefore a selection of technologies needed to perform such services are presented including

Architectures designed for integration of simulators.

Ontologies and standards which can provide a common information model for exchanging information between tools and systems

Mapping technologies used to specify and implement synchronization between ontologies and models

Communication technologies needed to integrate systems with different deployments.

3.5.1 Architectures

The selected architectures present the state of the art in simulation integration for engineering/military and process industry domains. Building support for these architectures may prove relevant in the future and architectural insights gained from analyzing them can be used to organize integration of tools in the modeling and simulation environment.

3.5.1.1 IEEE Standard 1516 (High-Level Architecture, HLA)

HLA [De] 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 *run-time infrastructure* (RTI). The HLA approach separates the data model and the architecture's 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). [DFW98]

3.5.1.2 CAPE OPEN

CAPE-OPEN [CAP] is a standard to enable the re-use of process simulator components. CAPE-OPEN was developed in various EU projects in the late 1990's. The architecture of CAPE-OPEN 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. CAPE-OPEN uses standard middleware namely Microsoft COM [Mia] and OMG CORBA [Ob04] for component implementation. CAPE-OPEN 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 *non-sequential modular* and *modular hierarchical* variations, which are not commonly implemented. Many existing simulation vendors offer CAPE-OPEN components.

3.5.2 Ontologies and standards

An important set of industrial standards are the ISO TC184/SC4 [Ind] 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 [Na99]. 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* [In05a], AP 227 for *Plant Spatial Configuration* [In05b]. 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. The ISO 10628 [In97] is an important process plant flow diagramming standard. For the Finnish process industry, the standards of PSK are relevant. PSK has produced a series of standards for process design data in structural form [PS06a][PS06b]. For the construction industry, the IAI Industry Foundation Classes (IFC) [Inc] standard is relevant. IFC is widely supported by construction CAD systems but many of its features have not been widely adopted due to the rapid evolution of the standard. For manufacturing industry the MIMOSA OSA-EAI [MI06] standard offers an integration model.

Academic and industrial stakeholders have expressed increasing interest in the W3C Semantic Web standard OWL for representing industrial data [Tu06]. The ontological approach with formal semantics has also been adopted in the Process Specification Language (PSL) [SGT+99]. The ISO 13584 PLIB is an ontological approach intended to support user interfaces and automatic integration [In01][Pi04]. The ISO 15926 includes a formal upper ontology and a Reference Data Library (RDL) with comprehensive set of product data for oil and gas industry. The Wilmington Agreement between large multinational companies suggests widespread adoption for ISO 15926.

3.5.3 Mapping

The modeling and simulation environment needs to be able to store and process models of different systems in different domains. In many cases the models are expressed using different conceptualizations but with some semantic similarities. For example, a pressure in a CFD model represents the same thing as a pressure in a larger scale dynamic simulator. Establishing this connection between these different ontologies is a key requirement for achieving such business goals as multi-scale and multi-domain modeling ability.

Such problems have been studied for a long time. The work in schema integration studies ways to automatically integrate database schemas into a combined schema [BLN86]. The problem is hard (AI-Complete) and a multitude of approaches has been developed to integrate data sources [DH05]. The area of performing *data integration* e.g. queries over heterogeneous data sources has received much attention [HRO06]. Important approaches include the central mediated model with semantic mappings to information sources and the peer-to-peer approach with local mappings between information models. Both can be used in the context of the modeling and simulation environment. The technologies of the Semantic Web namely description logics can be used to assist in semiautomatic creation of the mappings. Comprehensive surveys of such approaches include [KHR+05][No04]. Most research has been about discovering simple one-to-one mappings and usually some user intervention is needed. Providing integrating queries for heterogeneous data sources does not require communication between the sources but providing interoperability between data models requires synchronization between *instances* of structured data in different sources. Such features have been presented in [MMS+02][Le05][QZ06]. Popular representations for the mappings include description logic and rule-based approaches.

3.5.4 Communication

Integration and interoperability between systems involves communication of configuration and real time data. The communication can take place in process, inter-process or in a distributed manner. The technologies presented here address the issues of *machine-to-machine* communications as compared to *human-to-machine* communications presented earlier. For such interfaces the issues of performance, ease of development and maintainability are essential. The W3C standard Web services are

designed to support interoperable machine-to-machine interaction over a network. Web services provide for transfer, service specification and discovery using W3C standard SOAP, WSDL and UDDI respectively. Web services are strongly endorsed in Microsoft Windows Communication Framework in .NET Framework, which encourages their adoption. Many current industrial applications communicate using software component technology, mostly Microsoft COM/DCOM/DDE. These technologies are currently being replaced by newer technologies such as Web services. Many enterprise information systems connected to industrial processes use Enterprise JavaBeans (EJB) solutions. For basic information exchange, many current systems use files and standard file transfer protocols such as FTP and SSH. For transferring bulk data the file-based approach is highly relevant. The needs for real time data transfer differ somewhat from model configuration exchange. In the domain of process industry, the popular OPC [OP] standards address real time communication needs between automation devices and systems [LPK+03]. Services covered by OPC standards include real time data acquisition, events and alarms, historical data and server-side data exchange. The OPC specifications developed since 1996 have been widely adopted in process industry. A new OPC Unified Architecture standard was released in 2006 and is based on a graph data representation model. Real time data exchange requires additional technologies such as data reconciliation for robust integration. *Data reconciliation* technologies address the errors of data acquisition due to lack of precision, noisy measurements, faulty instruments and calibration problems. General approaches for data reconciliation are model-based filtering or inclusion of a measure of uncertainty. In many cases the real time data acquired from external systems needs to be fitted into the outputs of some model. This process of *parameter estimation* can be used for offline or online analysis of measurements. The access of offline data is important for many applications and standards such as OPC include specifications for historical data, which is used to access large sets of acquired time series data.

3.6 Identified systems and tools to be integrated

The integration of modeling and simulation tools requires strategic alliances between actors inside VTT, with international research partners and with the Finnish industry. Initial requirements of the modeling and simulation environment are strongly based on the systems and tools identified for integration based on such partnerships.

Simulation tools are considered for integration based on simulation needs at VTT. Open Source solvers and solvers developed at VTT are emphasized.
Process simulation tools developed at VTT are introduced as primary targets for integration.
Design systems and integration platforms are considered for supplying design data into the system

3.6.1 Simulation tools

3.6.1.1 MATLAB/Simulink

The MathWorks MATLAB suite of mathematical modeling tools is based on an interpreted modeling language, which is optimized for linear algebra calculations with high-dimensional data. In addition to the convenient syntax, the MATLAB suite provides powerful and comprehensive libraries for common numerical mathematics tasks and many domain-specific libraries called *toolboxes*. MATLAB also provides 2d and 3d-tools for visualization of results. MATLAB Simulink is a graphical block diagramming tool for modeling and simulation of dynamic systems. Simulink supports hierarchical blocks and an API for creating custom blocks using MATLAB language or C/C++, Ada or FORTRAN. Many engineering research codes are written in MATLAB since MATLAB is well suited for rapid prototyping and application development. MATLAB models can be managed in textual form and the models can be deployed in standalone executables capable of interacting with the modeling and simulation environment.

3.6.1.2 Modelica

Modelica is a modeling language primarily based on equations instead of assignment statements. This permits acausal modeling that gives better reuse of classes since equations do not specify a certain data flow direction. Modelica has multi-domain modeling capability, meaning that model components corresponding to physical objects from several different domains can be described and connected. Modelica is an object-oriented language based on general classes. Modelica also has constructs for creating and connecting classes as components [Fr04]. Implementations of Modelica

include an open source Open Modelica [Prb] and a commercial Dymola [Dy]. Modelica models can also be managed in textual form and the models can be deployed either in standalone executables or executed in a Modelica server. The connection mechanism in Modelica can be used to provide a flowsheet-style user interface along with textual code. Other somewhat similar developments include ASCEND [WAR+97] and gPROMS [Pra].

3.6.1.3 Companion model

The companion model approach described by Juslin [Ju05] offers a general framework for modelling and simulation of industrial processes. The approach is based on a generic flow network model, which can be used to simulate a wide range of physical phenomena in different domains. The framework provides means for hierarchical decomposition and construction of simulation unit operations by configuring and combining different physical mechanisms (represented by differential-algebraic equations). Fragments of the flow graph can be packaged into modules for reuse. Such models can be constructed using flowsheet-style tools with hierarchical capability.

3.6.1.4 System dynamics

System dynamics (SD) deals with modeling and simulation of macro-scale systems. Systems described by SD models are typically large, complex, nonlinear sets of differential equations. The nonlinear models cannot be analysed using theoretical system analysis. SD models include flows of information and matter with delays and feedback loops. The assumptions used in the models are heuristic and statistical and thus the simulations cannot provide fully quantitative results but rather qualitative ones. Importantly the different feedback loops and turnaround points of system behavior can be analysed and exploited [St00]. SD provides important modeling capability for business processes. The configuration of SD models is usually done using graphical models close to flowsheets as shown in Figure 7.

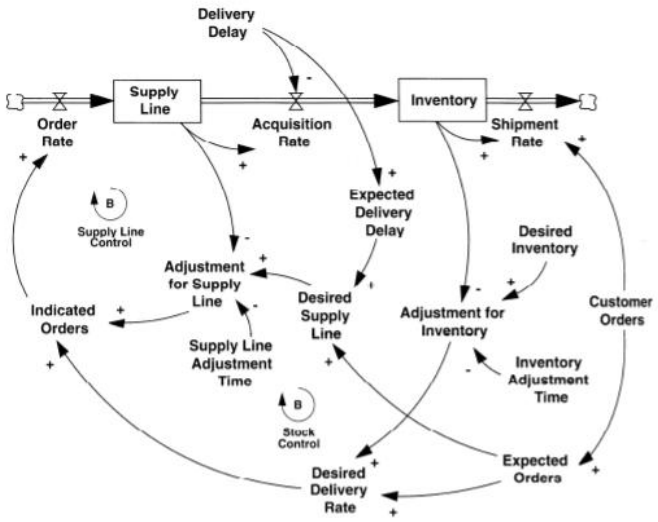

Figure 7: Graphical representation of a System Dynamics model [He].

3.6.1.5 Phase field methods

Phase field modeling [Ch02] is currently considered the "state of the art" approach to modeling phase transformation problems, especially solidification. Phase-field methods are still in the research phase but applications for metal solidification in continuous casting and multiphase flow in pipe networks have been suggested. The main advantages of the approach are numerical modeling of complicated multi-phase flow and phase transformation dynamics without the need for tracking the position of the phase boundaries and a robust physical motivation from thermodynamic free energy without arbitrary or heuristic assumptions and simplifications. The phase-field methods can be used to calculate parameters for meso-scale models using micro-scale models. The phase-field methods involve complex 3d-simulations with large amounts of data. A suitable parametric or tabular representation of the meso-scale parameters can be precalculated in most cases.

3.6.1.6 Finite elements and CFD

Elmer [CS] is a finite element (FE) software package for the solution of partial differential equations (PDE) developed at Finnish IT center for science (CSC). An FE solver is an essential tool for solving two- and three-dimensional PDEs arising in most physical modeling domains. For CFD calculations, *Open Field Operation and Manipulation* (OpenFOAM) [Opb] is an open source CFD solver library developed by OpenCFD ltd. The PDE models usually consist of a textual model representation along

with *initial and boundary conditions* and a *mesh*. Currently the meshes can consist of millions of elements, which is a huge set of data.

3.6.1.7 Community technology

Community technology research at VTT has included development of simulation software. A district heating simulator based on MATLAB is used in large-scale design of communities. *VTT Talo* is a simulation tool for evaluation and development of component-level or holistic structural and equipment systems for detached/semi-detached houses. The simulator provides results for inside conditions, energy, life cycle, environmental and feasibility studies. The models of community technology need to address the life cycle of a community from conception to demolition. The configuration of district heating and house simulation involves two- and three-dimensional graphical flow models.

3.6.2 Process simulation tools

3.6.2.1 APROS

Advanced PROcess Simulator (APROS) [VTa] is a dynamic process simulator developed by VTT and Imatran Voima Oy (currently Fortum Oyj) since 1985. APROS uses equation-based models for solving e.g. pressures, temperatures and mass fractions for multiphase flow in process equipment such as pipes and tanks. Control systems can also be modeled in APROS. Successful APROS applications include training simulators, automation testing and control design, optimization and safety analysis [YPK+05]. APROS models are currently built using a graphical user interface Grades, which uses a flowsheet approach. APROS can also be accessed using a command interpreter. APROS is the primary target for integration in this work as people involved in the development of the modeling and simulation environment are developing it.

3.6.2.2 BALAS

BALAS [VTb] is a steady-state simulation package for chemical processes with emphasis on pulp and paper. The software has been developed at VTT over the last 20 years and several Finnish paper mills, engineering companies and equipment manufacturers currently use it. BALAS has been extensively developed in close co-operation with Finnish forest industry and Tekes. BALAS uses a sequential-modular approach for solving e.g. pressures, temperatures and mass fractions for multiphase flow in process equipment. BALAS models are currently built using a graphical user

interface *Flosheet* or more recently using *Microsoft Visio*. BALAS has also integration with *Microsoft Excel*. The integration of BALAS and interoperability between APROS and BALAS are primary targets for the modeling and simulation environment.

3.6.2.3 Thermo-chemical simulators

A widely used simulator code for thermo-chemical multi-phase equilibrium, ChemApp [ES95], has been developed since 1975 in European research projects. ChemApp is used in programs from such areas as process simulation and modeling, computational fluid dynamics (CFD), solidification and casting simulation, crystal growth, kinetically controlled production processes, etc. VTT has developed a ChemApp and Microsoft Excel – based simulation tool ChemSheet for performing simulations. Calculation codes such as ChemApp require thermo-chemical data of chemical compounds, which are available in large databases such as *Facility for the Analysis of Chemical Thermodynamics* (FACT) databases. ChemApp has been used to create chemical models for APROS and BALAS, which makes its integration relevant. ChemApp modeling could also benefit from graphical user interfaces. The integration of FACT databases could enable important capabilities for calculation of material properties in the environment.

3.6.3 Plant design systems and integration platforms

A recent development in Finnish industry has been the creation of an association of distributed industrial knowledge management (THTH) [TH]. THTH advocates electronic and distributed knowledge management in Finnish industry using a platform called *Prindex*. Prindex is an information management system developed in multiple projects involving Tekes, PSK and Finnish engineering consultants. System characteristics include a generic object-oriented information model, validity services, multi-user information access, information visualization, versioning and access control and information security.

Intergraph supplies modeling tools for process industries. The main tool *Smart Plant 3D* is used for making a 3d model of a process facility. The 3d model can then be used to generate PI&D drawings, component lists and other important documents. The Intergraph suite of tools uses a data warehouse called *SmartPlant Foundation* to store the design data. Main capabilities of *SmartPlant Foundation* are a common integration architecture utilizing an electronic library of all plant information stored against the

familiar plant structure with support for auditing, change management and information access interfaces [Inb].

The integration of popular plant design systems could prove essential in producing the data needed to build simulation and decision support models in industrial investment projects. Such integration requires complex data transfer and mapping between the systems. Existing technologies for such data transfer include Xmplant [No].

3.7 Analysis

What can be done with existing modeling and simulation technologies? The state of the art in modeling and simulation is specialized and efficient tools for dealing with domain-specific tasks. From the architectural viewpoint, the technologies can be divided into custom codes, monolithic engineering systems and integration solutions. The custom codes usually do not provide good external interfaces but such connections can be implemented. The monolithic engineering systems usually provide some interfaces, often COM/DCOM or custom TCP protocols, but the interfaces are usually not comprehensive enough for straightforward integration. Modern integration solutions do provide interfaces for integration, but the software can be costly and the interfaces proprietary. Only the most modern systems provide support for distributed teamwork and associated technologies of versioning and access control. The state of the art in user interfaces ranges from command line programs and code editors to custom graphical tools such as flowsheets and 3d drawings. Most of the graphical solutions are not utilizing modern technologies. Results are most often obtained in textual form and custom solutions for visualization, trending and plotting are available. Technologies from knowledge management are usually not utilized in a large scale. Most programs provide some custom analysis and validation tools.

What could be improved in existing modeling and simulation technologies? There is wide potential in combining capabilities of different yet in some ways related modeling and simulation tools. Especially in engineering the combination of *different domains* is needed to perform holistic analysis of complex systems e.g. vehicles. The combination of *different scale* models would enable meso and macro scale analysis based on robust first principles models. The construction and reuse of existing models could also be improved by moving into a more component-based and layered approach of modeling.

The global economy requires distributed teamwork and collaboration tools since in many modeling and simulation projects the different stakeholders are geographically dispersed. The introduction of knowledge management technologies such as verification and validation, optimization, model generation and decision support systems can greatly reduce the costs of modeling and simulation projects. The user interfaces could be modernized in many cases and graphical model configuration could be more utilized. Especially the visualization means could be significantly improved using modern graphics and animation technologies.

What are the services and features required of such an environment? The requirements for distributed teamwork and collaboration and means for performing mappings between models and communicating with external systems have been motivated in the vision chapter. For an extensible platform the management of components and services is essential. Many simulation and live systems produce large sets of data, which need to be processed, transferred and stored efficiently. The data models of different tools and systems vary greatly from simple but large data structures to complex specifications with embedded logic such as EXPRESS. Many engineering models include aspects of life cycle management, which requires environment-wide treatment of temporality. In addition to efficient real time primitive data many knowledge management technologies require efficient means for processing e.g. searching through different configurations produced algorithmically. The user interface needs include development of program code in different languages, configuration and visualization of various models in 2d and 3d and visualization of calculation results using different means.

What existing technologies could be used to achieve the requirements? For architectural design, many standard solutions exist for realizing the distributed collaboration model and associated requirements of security and versioning. The framework for logical knowledge representation is also strong and should be exploited. The strong user interface frameworks of Eclipse can be utilized along with dedicated frameworks such as Batik [Apa], OpenCascade [Opa] and OpenGL for producing 2d and 3d. The research on ontology mapping can be used as a reference but comprehensive solutions are not currently available since the research has been concentrating on open world issues of the Semantic Web. For machine-to-machine and machine-to-human communications, strong standard solutions are available and should be exploited.

4 Problem statement

4.1 Overview

Recent developments in research of modeling and simulation in process industry have lead to a strategic effort at VTT for developing an integration platform for simulation and modeling. The groundwork for this effort has included research of different aspects of industrial data management, software architectures, visualization technologies and recent information modeling technologies. The research activities have been carried out in close co-operation with the Finnish industry. *The purpose of this study is to identify and analyze the most important requirements and design challenges involved in the creation of such an environment.* The scope of this study does not allow a detailed treatment of all requirements and design challenges. The reduction of the scope has been done by keeping the level of discussion in fundamental issues. The option of selecting some subset of the full problem was not taken because the research work done for this work has been about holistic system-level design of the architecture. The issues identified in this study are not restricted to issues with technological difficulties. Some issues are taken into consideration because their successful solution is critical to the success of the environment. The nature of the system in consideration is elaborated in the following chapters.

4.2 Hypotheses and approaches

To make the treatment of relevant issues more straightforward in this work some propositions are treated as hypotheses such that they are not further scrutinized in the following discussion. Some rationale for these assumptions has already been given and some are further elaborated in Table 2.

Hypothesis/approach	Rationale
Integration of existing modeling and simulation systems is needed.	Existing research and reports suggest benefits. The interest in the example case of integrating APROS, BALAS and CFD has been demonstrated in previous projects at VTT.
Support for teamwork and collaboration needs to be included in the environment.	The global development of network economy is undeniable and has been emphasized by development steering organizations such as Tekes and NIST. The support of teamwork and collaboration in existing

	systems is often poor.
Integration of tools and methods is achieved by mapping information models	Integration using mappings is a mature field in database systems, but its problems have not yet been fully solved. The selection of the mapping approach is subjectively based on the belief that semantic modeling makes such mapping feasible.
Data mappings are achieved using an ontology-based approach.	There is high academic research interest and hope in feasibility of mapping between semantic models. Since the capabilities largely remain undemonstrated, this hypothesis contains risk.
The rapid technological advances in computing will continue.	The hypothesis is widely supported by history and current research developments. The impact of this hypothesis is a forward-looking treatment of issues in this work.
The ontological representation uses a graph-structured data model motivated by W3C RDF and OWL.	The Semantic Web standards have been widely studied and provide a subjectively feasible way of modeling ontology. Adopting a popular standard approach has also many integration benefits.
The user interface is based on the Eclipse Platform.	The use of Eclipse has already been established in previous projects at VTT. Eclipse has a strong and active developer and user base.

Table 2: Hypotheses and approaches for the modeling and simulation environment

4.3 Objectives

The main objectives of the modeling and simulation environment development have been determined in research projects at VTT. Only the most important objectives are listed in Table 3 to provide context for analyzing the design challenges.

Objective	Rationale
Means for tool and method integration	This objective is based on a hypothesis. The initial focus is on simulation tools and models in process industry.
Distributed teamwork and collaboration	This objective is based on a hypothesis. This objective also implies support for the requirements of industrial

	modeling projects including robust security and versioning.
Means for developing multi-scale, multi-physics modeling and simulation	Many integration-related industrial modeling problems are related to multi-scale and multi-physics modeling. Relevant means include integration of old and also development of new modeling approaches.
Verification and validation	Increasing automation is a key factor in increasing productivity, but automation requires robust verification and validation of model configurations and decision support algorithms.
Means for developing decision support	The semantic data model opens up new possibilities for decision support from the field of knowledge management and AI. These new approaches need to be integrated with existing technologies of simulation and optimization.
Productive task-oriented user interfaces	The importance of usability has been clearly demonstrated in the software marketplace. The success of existing graphical modeling and simulation tools (APROS, BALAS, UML) also stresses the importance of productive user interface.
Advanced visualization	The importance of advanced visualization has been stressed in many reports about modeling and simulation. The potential for improvement with e.g. modern hardware is especially large in the visualization field.
Component-based modeling and simulation	Most successful extensible software platforms rely on a powerful component-based model. Such approaches have also been introduced in modeling and simulation by specifications such as HLA and CAPE-OPEN and by languages such as Modelica.

Table 3: Main objectives for the modeling and simulation environment

4.4 Requirements

The contribution of this work includes to some extent the analysis of requirements for the modeling and simulation environment. Most essential requirements have already

been introduced in the previous chapters. The following list of requirements does not offer a comprehensive and detailed list of all requirements, but rather makes a small selection of the use cases and functional requirements. The selection criteria include their impact to overall architecture and their relevance to the vision. The selection of use cases is also biased towards the concrete initial applications. The rough selection serves to keep the treatment of the design challenges relevant and to restrict the scope of the study. These initial use cases and functional requirements are listed in Table 4 and Table 5 respectively.

4.4.1 Initial use cases

Use Case	Rationale
Performing distributed collaborative and concurrent design work.	Addressing these requirements has a strong influence on the overall architecture of the system.
Configuring a large-scale industrial simulation model	The initially most important application of the environment will be the configuration of APROS simulation models.
Managing a set of synchronized industrial simulation models	The hypotheses of integration via mapping needs to be validated at an early phase and the use case needs to be given high priority. Initial applications will be between APROS and BALAS simulation models.
Generating a simulation model from engineering design data	Currently the creation of simulation models is lengthy and costly. Automatic model generation from e.g. plant design system data would be a major breakthrough for simulation.
Managing a simulation experiment with multiple models and simulators	Simulation model integration also requires integration of simulation execution. Such use cases also address the issues of real time data processing. Initial applications will be combined simulation of APROS and BALAS models.
Developing and managing an ontology	The processes and methodologies of developing ontologies need to be clear from the beginning. Ontology development tools should be given high

	priority.
Developing and deploying a software component for decision support.	Tools for developing ontologies need to be augmented by tools for developing software components using the ontologies. Interoperability between such components needs to be supported by a strong architecture designed early.
Developing and managing a model component	Existing component-based modeling approaches (APROS, BALAS, Modelica) have been successful and this success needs to be transferred to other domains as well.

Table 4: Selected use cases for the modeling and simulation environment.

4.4.2 Functional requirements

Functional requirement	Rationale
Security	Security is mandatory for establishing trust in industrial network economy.
Version control	Version control is an essential feature in most document management systems and its success in software industry has been impressive. Most developments in industrial information management include version control.
Industrial strength persistent storage	Industrial strength includes scalability to massive data and prevention of data loss or corruption in any situation. These are essential business requirements.
Interactive performance	The long processing times of ontologies in some research applications are not acceptable in most industrial applications. Interactive performance is a requirement in all but the most demanding analysis services.
Real time and historical data	The processing of real time data is an essential feature of the modeling and simulation environment. The environment needs to be able to process real time simulations of e.g. complete nuclear plant automation systems.
Visualization of model behavior	The requirements of visualization need to be addressed by powerful frameworks, which constitute an integral part of user interface design.

Verification and validation of ontologies, models and simulations	The issues of verification and validation should receive a focal point in the processes and user interfaces of the modeling and simulation environment.
Life cycle modeling	Life cycle modeling has become a requirement in construction and investment project modeling. Robust treatment of temporality needs to be an integral part of the environment design.
Inexact modeling	The importance of inexact modeling has been stressed in reports and the benefits are clear in many decision support applications including simulation. Support for inexact modeling is poor or non-existant in most current modeling solutions and provides thus an opportunity.

Table 5: Selected functional requirements for the modeling and simulation environment.

5 Design challenges and possible solutions

5.1 Introduction

For the treatment of design challenges of the modeling and simulation environment a collection of 14 focal areas are selected based on subjective analysis of importance. These areas are further divided into *immediate* and *imminent*. From each area a few essential challenges are represented and analysed. Solutions for the immediate design challenges are needed for any coherent design of the environment and these solutions are required before any implementation can begin. Solutions for the imminent design challenges are required in order to fulfill the vision of the modeling and simulation environment. Solutions for some of the imminent design challenges can be delayed for some time, but eventually all of them will be required. A perfect design would therefore consider all of the design challenges presented.

The immediate design challenges are presented in Table 6 along with their rationale for inclusion. The imminent design challenges are similarly presented in Table 7. The treatment of the immediate design challenges includes references to a *prototype case design* developed during this work. This prototype design is not thoroughly represented or analysed in this work, but is rather used to provide informative insight through examples.

Immediate design challenge	Rationale
Distributed architecture	The distributed architecture has a profound impact on the user experience and programming interface of the environment.
Knowledge representation	The fundamental knowledge representation decisions need to address a wide range of modeling requirements.
Performance and scalability	The semantic modeling approach is computationally intensive compared with existing solutions. Users will most likely not accept a low-performance solution.
Mappings	The modeling and simulation environment is an integration platform and mappings are the selected approach.

Real time processing	Simulation has very specific real time requirements of a knowledge management system. Fulfilling these requirements will make the environment stand out from existing solutions.

Table 6: Immediate design challenges with rationale.

Imminent design challenge	Rationale
Verification and validation	The adoption of verbose and expressive graph data model is mainly justified by projected benefits in the areas of verification and validation and other decision support services.
Extensible decision support services	In developing a platform, the issues of extensibility should be addressed carefully since the impact of the integration platform is determined by its adoption.
Security	The issues of security are required in industrial collaboration and the distributed graph data model makes the challenges nontrivial.
Reuse and redundancy	The issues of reuse and redundancy involve a strategic vision of *modeling through configuration*, which is given special treatment in this work based on perceived success of such modeling and simulation tools.
Life cycle information management	The issues of life cycle information management and inexact modeling are identified as growing trends in modeling and simulation and providing comprehensive support for them in the environment is considered.
Inexact modeling	
User interface	The vision of the modeling and simulation environment has a strong emphasis on user interface. This critical aspect in productive engineering work has been somewhat neglected in many existing modeling and simulation tools. The wide range of projected use cases makes design of the user interface more challenging.
Documentation and tacit	The inclusion of documentation and tacit information

information management	management has been selected since it is of significant economic importance in engineering work and the semantic approach provides significant new possibilities.
Future proof design	The discussion on future-proof design points out the need to acknowledge and address change in the fast-moving field of technology.

Table 7: Imminent design challenges with rationale.

5.2 Prototype design case

The prototype design case study was about steady-state simulation of the bleaching process in a pulp mill. The design included a versioned knowledge base and a user interface with tools such as model browsers, flowsheet editor and a three-dimensional editor. The simulation model of a bleaching line was configured using flowsheet or three-dimensional editors and simulated using a custom simulator developed for the case. The design included support for teamwork, which was not used in the case study demonstration. Different aspects of the case have been analysed and reported in the master's theses of Toni Kalajainen [Ka07], Tuukka Lehtonen [Le07] and Marko Luukkainen [Lu07]. The distributed architecture of the prototype system was based on a client-server model. A locally deployed persistent server was connected to a remotely deployed server with versioning support. The Eclipse-based user interface was connected to the local server. Software components, such as simulators, could be executed at the servers or at the client.

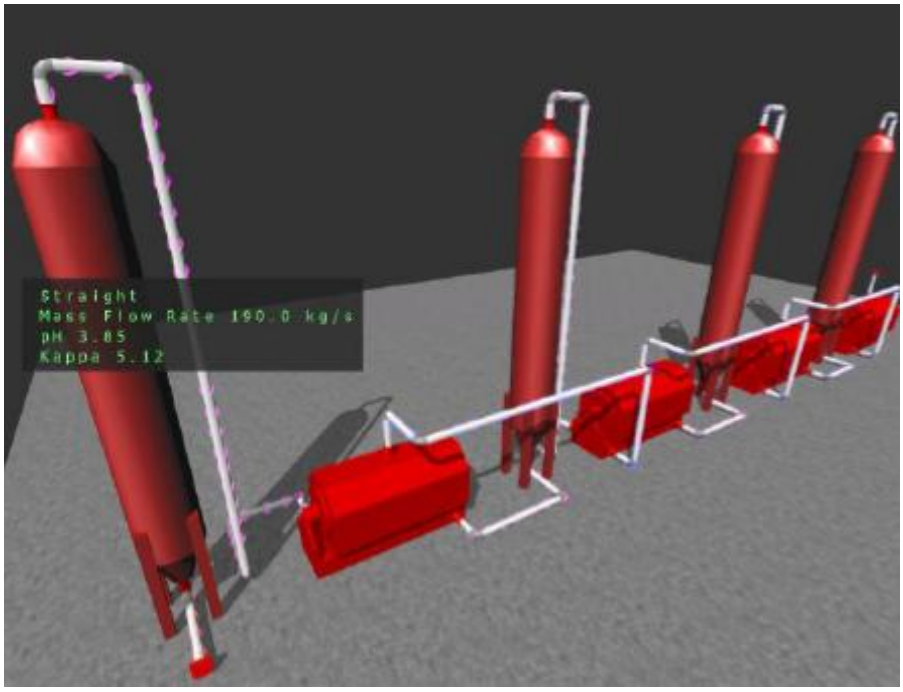

Figure 8: A three-dimensional model of a bleaching line with visualization of simulation results. [Lu07]

5.3 Immediate design challenges

5.3.1 Distributed architecture

The design of a distributed multi-user architecture is always a challenge. The inclusion of a semantic data model with no predefined structure and strong real time capabilities makes many existing solutions directly inapplicable.

5.3.1.1 Supporting relevant collaboration topologies

The projected use cases for the modeling and simulation environment range from local workstation deployments to collaboration between globally dispersed project consortiums. In local deployments, the user has no needs to perform complex access control and synchronization with other users. In company-level collaboration cases, a team of engineers uses the environment for concurrent work on a modeling project. For this case, the issues of synchronization become relevant and access control may or may not be needed. The most challenging case is a large investment project with collaboration among a consortium of design companies, suppliers, operators etc. In this case, a fine-grained control of synchronization points and information security is needed. In general, the possibly competing subcontractors will not allow their confidential data to be seen by their competitors. This implies that the system needs to support persistent information stores with independent management for each

collaborating party. The synchronization model also needs to support the traditional method of producing predefined sets of deliverables at predefined times. The natural distributed topology for corporate deployments is the hierarchical client-server whereas the collaboration between companies would imply peer-to-peer topologies. Issues of such an architecture are presented in Figure 8.

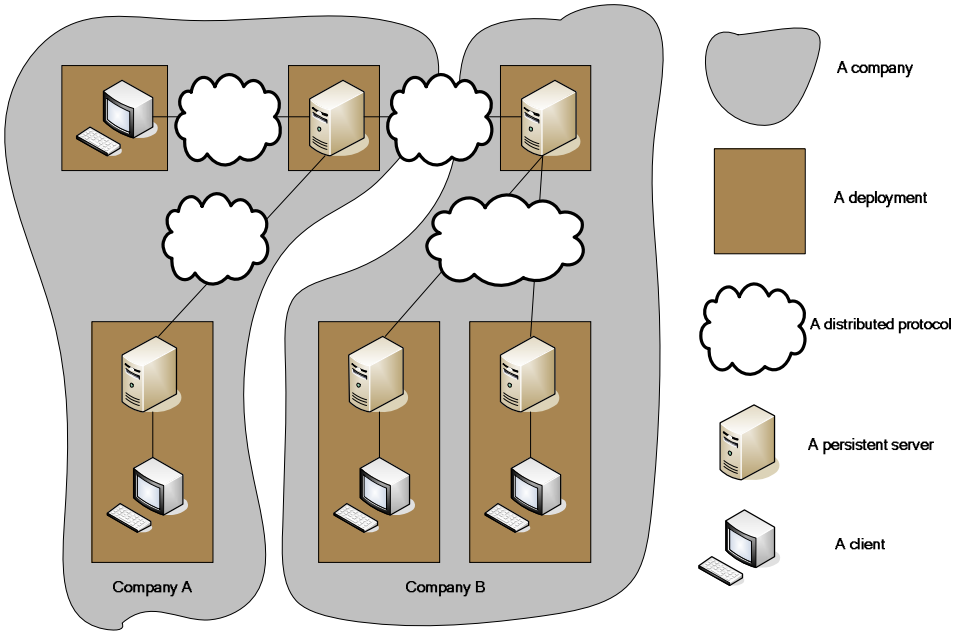

Figure 9: Distributed architecture across companies and deployments with persistent servers and clients.

5.3.1.2 Storing the distributed data

The options for storing a distributed information model range from totally replicated databases to schemes, where each data item is stored only in one persistent location. Replication is easy to implement but major problems include security and performance. If all data was always replicated some information security could be established using strong encryption, but there is the risk that users would not trust such an approach. If only a partial replication is used, the security issues can be dealt with already at the transfer interface. A large modeling project such as a process plant produces huge amounts of data. Replication of such data to many servers is costly if the clients of these servers only use small parts of that data. If the data is not replicated a server needs to be able to acquire it whenever its clients request. In a hierarchical setting, each server has a parent server from which to query the information. This places high performance demands for the central servers and requires that these servers have access to all relevant information. In an industrial investment modeling project, the procurer can operate a trusted root server. In other cases, some other trusted server operator

would be needed. Peer-to-peer technologies do not require a trusted root server, but such approaches require complex management of responsibilities for storing pieces of the information model. Essentially such management becomes costly if the information model needs to be accessed with small granularity. A workstation server can also get disconnected from the network while traveling etc. This requires methods for making relevant information persistently available in a server.

5.3.1.3 Managing changes and versions

The main benefits of existing version control systems are the possibility for concurrent work, management of changes with possibility for rollback and auditing. The version control approach to concurrent work uses a local working copy, which can be modified and committed at a suitable time. Version control systems are often used without global locking with a risk of conflicts due to concurrent modifications. Such an approach is suitable for textual content since possible conflicts are easy to resolve manually. In the semantic graph data model, the resolution of conflicts is significantly harder since the semantic constructs are large and hard to interpret and process manually. A conflict resolution strategy needs to discard one of the conflicting changes and to ensure that the model remains in a meaningful state after the correction. The challenges of implementing such a strategy involve co-operation with verification and validation services and representing the conflicting information model to the user using concepts relevant to the user instead of low-level statements. Because of these difficulties, conflict avoidance should be preferred. Such avoidance is possible with locking but a robust and efficient locking scheme is hard to implement for a large distributed deployment. Some locking will be required to access the working copies in a cooperative fashion. Such needs are encountered in clients with multiple user interface elements requesting read operations and responsive agents such as model mappers requesting write operations. For such cases, the system should provide robust ACID transactions. In any case, the operations with a risk of a conflict should be user-initiated and for automatic operations, success should be guaranteed by locking. Finding a balance between the risk of conflicts and the problems of locking is a challenge. The management of the working copies is another challenge. A working copy requires a repository for reading and committing changes. In large deployments, the working copy itself would benefit from version control. This brings about an interconnected set of dependent version hierarchies. The data in a version hierarchy needs to be stored in

an incremental fashion (change sets) but efficient access to data contained in revisions requires complete snapshots of the data to be built. To complicate things, some data might not be loaded into a server before its clients access it. Efficient design of such a distributed hierarchical version control scheme is a challenge. Many concepts from version control are also useful in a wider context. The concept of a change set can be used whenever modifications to the information model are considered. The concept of a revision can be used to version complete models or to record undoable operations for clients. The concept of a branch can be used to make development sandboxes for projects or to create transient versions of the information model for a client. The generalization of such concepts offers great benefits and poses great threats. The benefits include interfaces with few generic concepts easy to learn and to understand. The threats include use of generalization when it is not completely applicable, which leads to nonfunctional interfaces and suboptimal implementations.

5.3.1.4 Designing the client interface

Clients including user interfaces, long-running calculations and services such as intelligent agents access the distributed persistent information. An important issue is again distribution. In most client installations, a persistent server is deployed in a workstation with user interface and decision support clients accessing it. In such installations, the clients can use efficient IPC protocols for communicating with the server. A distributed client interface on the other hand provides flexibility of deployment, which can prove important for connecting to external systems and supports small e.g. mobile clients. The decision about distribution has far-reaching implications. In particular, a distributed client interface must address the restrictions imposed by bandwidth and latency of distributed protocols. The bottleneck of distributed communications can be reduced by reducing the amount of data transfer (bandwidth) and the amount of operations (latency). The basic approaches to reducing bandwith and latency are caching and server-side queries. In typical use cases such as user interfaces there are tens of clients (editors, views etc.) accessing the same data model. For such cases a heavy client with caching produces best performance with distributed interfaces. Other simple clients such as web reporting etc. would be better served with thin clients and a powerful query language. The design of available queries involves selection between the common database indexing approach and a navigational approach. Indexing provides highly effective queries with global coverage but places

high requirements on initialization and modification times and memory. Navigational queries are local and can be indexed lazily. The use of navigational queries exposes large amounts of intermediate data and requires many primitive operations, but the navigational model is much more versatile and powerful for programming. Providing the navigational model in the client requires caching and means for transferring larger amounts of data from the server at once. Possibilities include partitioning of the graph, which is further discussed in the section of performance and scalability. Most clients also require notification of changes to their content, which is also discussed in the section of performance and scalability. A heavy client needs to synchronize its caches to the persistent server. Such synchronization requires communication when data cached in the client changes in the persistent store. Traditionally a client has initiated such synchronization. While such a pull protocol is easy to implement an active server–initiated push protocol includes notable benefits. The active push protocol reduces the amount of conflicting concurrent changes. Respectively, the implementation of such a push protocol requires that a session with state and a continuous distributed connection be established between the server and the client. The issues of distribution, caching, querying, notification and synchronization are all interrelated and finding an efficient solution, which satisfies all user needs is a major challenge.

5.3.1.5 Deploying and managing component services

The modeling and simulation environment can be seen as a multi-agent system, with passive (reading and listening) agents such as user interface views and active (writing) agents such as user action handlers, model mappers and simulators. The service agents interface with the modeling and simulation environment through the client interface, which can be provided at different servers in the distributed topology. With a distributed client interface, the clients themselves could be deployed freely. In a design office, each designer usually would host a local workstation server with work periodically committed to a shared server. Most services would then preferably be deployed in the local workstation for performance reasons. Issues of management or needs for multiple users could require some services to be deployed in the shared server. The management of such distributed services should use proven solutions from SOA with capabilities for discovery and specification. For deploying services in the user interface client the selected Eclipse platform offers the OSGi framework for dynamic deployment of service components. The semantic graph data model also

provides powerful means for service discovery, specification and deployment. OSGi provides for communications between services inside a single JVM but for distributed services other means for issuing requests are needed. The graph data model is a natural way of communicating between the services. If the services are connected to different servers with separate working copies of the data model, the changes made to the graph will not be automatically transferred between the services. Providing practical SOA in the context of the distributed versioned data model and OSGi is a major design challenge.

5.3.1.6 Prototype case considerations

Even the simple demonstration case highlighted the requirements and problems of collaboration in modeling and simulation. In the case three somewhat distinct users could be identified. The process engineer draws a flowsheet of the process with the intention of designing a process using a graphical tool. Another engineer then wishes to make a 3d model of the flowsheet for demonstration purposes. A simulation engineer later uses the flowsheet design to create a simulation model, which can be visualized in the graphical editors. In a real-world situation, different persons even in different organizations can perform these activities. The case demonstration was performed in a single workstation, but even the small case model produced considerable amounts of data and required a modern workstation to run. Even though the implemented design was in many ways inefficient the case implied that in a distributed teamwork setting, a full replication of all data to all clients would be a performance challenge. A hierarchical versioning knowledge base design was tested during the development (even though not much utilized in the demonstration case). Main difficulties included high-performance service of multiple clients in different revisions and persistent operation with transient performance. It also became evident that the update operations require some means to visualize and categorize the pending changes (no evident solution was found in this work). The prototype implementation included no robust transaction handling and relied on the fact that a single thread was used for the clients to perform access. It became evident that a robust multi-threaded access to the information model would require meticulous analysis of the ordering of read, write and notification operations. During the prototype implementation, simple server-side queries and navigational client-side queries were tested. The server-side queries reduced the amount of data transfer to the clients, but made client-side changes difficult

to handle. Since the resolution of changes is small (add/remove statement) some client-side processing greatly reduces the amount of communication with the server during model modifications. With local queries, modifications were fast, but more data needed to be transferred to the clients for processing. In the prototypes, simulation components were deployed both locally and remotely. For the user interface client, the data access interface was implemented in Java and for the server clients in C++. This made it impossible to deploy the same component in different locations. The extension point mechanism in Eclipse was widely used to communicate between services implemented in the user interface client.

5.3.2 Knowledge representation

The selection of a semi-structured graph data model based on W3C standards RDF and OWL is a hypothesis in this work. Because of the wide range of requirements, these web-oriented standards are not adopted as is and the considered modifications are discussed in this section.

5.3.2.1 Selecting navigability for the graph data model

The graph data model consists of *statements* representing a relation between two nodes of the graph. The usual interpretation of such relations is directed such that directed relations e.g. *whole-part* or *Instance Of* can be represented. Many navigational algorithms can be implemented more efficiently if such relations can be traversed in both directions. The bi-directional navigability raises issues if only a fragment of the graph has been loaded for processing. The design must then ensure that all statements about loaded nodes (outgoing or incoming) are available. A simple solution for the problem is the explicit modeling of statements for both directions, but such an approach doubles the amount of statements. With a predefined partition of the graph the inverse relations can be implicit inside partitions, but not over the boundaries. The requirement of bi-directional navigability also brings forth another question about the management of graph nodes with a large degree. In a large modeling project there can be thousands or millions of nodes with an *Instance Of* relation to a popular node e.g. the primitive data type *Double* and finding all instances of *Double* is not interesting. Still it is hard to design a navigational query such that processing of the *Double* does not involve processing of the uninteresting relations.

5.3.2.2 Specifying the status of statements in the graph data model

The statement model selected by OWL includes two nodes connected by named relations called properties. The model is well justified by interpretation of statements as facts about binary predicates in first order logic. In practical engineering applications, some other requirements can also be seen. A major issue in the OWL model is that the statements themselves are not first-class objects. The implications are that the statements cannot be referred to and they cannot be further described by attaching data. There are, of cource, ways to circumvent this restriction. The *reification* (i.e. refer to all three parts of the statement) approach in RDF produces large amounts of data and is not suitable for wide spread use. Another approach is to model a relation as a node in the graph. In many cases, it can be argumented that the model becomes less intuitive if such constructs are used instead of simple relations. Some ways of making the statements first-class objects include making the predicates instances of a relation and including a fourth statement identifier field to the statement. Modeling implications of first class statements are shown in Figure 9.

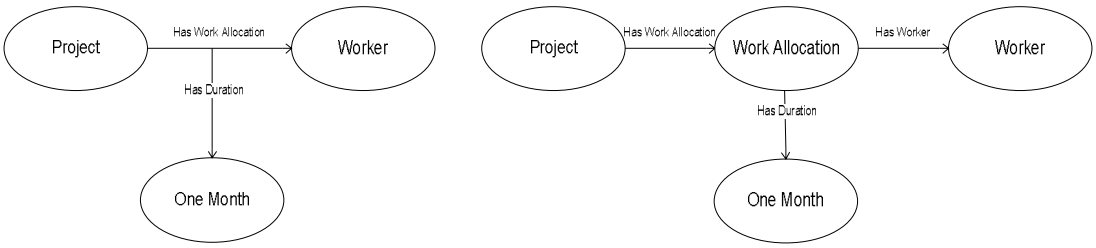

Figure 10: The difference in modeling with first class statements.

5.3.2.3 Specifying extensions to the statement model

There are also other engineering considerations regarding OWL representation. The requirements of the representation include fundamental issues such as temporality, uncertainty and access control. These features can be modeled on top of the statement model, but doing so has significant implications on performance. In many cases, significant performance gains could be attained by attaching some additional data to the statement level to provide low-level solutions to these engineering requirements. Such solutions can enable some critical operations to be implemented already in the interfaces and data structures of the persistent storage. These possibilities are elaborated in later sections. There are two basic ways to augment the statements. New nodes can be added to the statements and data fields can be added. The addition of new nodes makes the graph analogy somewhat questionable, but the extra nodes can be seen

as further specifications of the statement defined by the primary nodes. An important type of augmentation is *contexts*, which can be used to create identified sets of statements. Contexts are applicable whenever something needs to be said about a large collection of information. Augmenting data fields could be used to assign e.g. lifetime, fuzziness, or other uncertainty measure to the statements. The selection between node references and data fields is a careful trade-off between performance and expressive power. Node references make arbitrarily expressive definitions possible, but a fixed data field enables optimization at the low level processing protocols.

5.3.2.4 Representing primitive data

The treatment of primitive data in OWL is not designed for industrial real time data needs. Major issues include the treatment of complex data types, representation of real time values and representation of multiple value configurations. The graph representation is well suited for specification and referring to complex primitive data, but the straightforward encoding produces too much data as determined in experiments [Le07]. Many users of large amounts of primitive data such as simulators also benefit from a controlled serialization of primitive data structures into tabular forms. The specification and serialization of primitive data structures is straightforward but retaining the important ability to refer (with statements) to constructs in the structural data is a challenge. The challenges of real time processing of primitive data are discussed later in the section of real time requirements. Another common requirement for engineering models is the ability to process and store different configurations of primitive values for a model. Examples include different interesting states (pressures, flows, controls etc.) of a large-scale process simulation model. Such states can contain large amounts of data and their representation using semantic statements would be too costly.

5.3.2.5 Specifying the semantics of the data

For a comprehensive modeling and simulation environment with services for advanced decision support it becomes obvious that a single semantic approach will not be able to deliver all needs. A layered approach is needed instead to reach higher levels of abstraction. Within the ontological approach, the purpose of the first layer is to define semantics such that meaningful ontologies can be constructed to serve as the basic building block for higher-level specifications. The OWL specification provides a starting point with effective reasoning support for schematic validation and inference

based on the mathematical model of a description logic. The primary domain of description logics is to establish modeling of ontologically valid structures. The basic ontological mechanism of OWL is based on classifications according to set theory. In most engineering applications, the semantic models need to be offered in simple and practical terms, which a common engineering user can easily understand. Other developments in industrial data management have been driving forward the issues of *object orientation*. In the object-oriented approach, some fundamental issues are *inheritance* and *interfaces*. Both are in a way included in the OWL as *subclasses* and *requirements*. The unfamiliar features in OWL are the construction of classes from other classes using set operations, which is stronger than a traditional object-oriented inheritance. It can be argued that a model with explicit inheritance is easier to introduce to the engineering user. Another issue is that traditionally object-oriented classes can be *instantiated* to build models but in OWL the classes do not generally support this feature (especially the *union* and *not* operations are problematic). The *open world assumption* inherent in OWL is also not fully justifiable for a unified information model. The closed world assumption gives the reasoning engine possibility to make more inferences as expected by the user. Embracing the object-oriented approach also involves attaching interface operations to the models. In most real-world cases, the theoretically computationally tractable specifications such as OWL are not enough and more powerful constructs e.g. recursion and procedural algorithms are needed. In such cases, the ontological level semantics need to be augmented by e.g. rules, equations, logical programs, procedural programs and informal specifications. The integration of these separate mechanisms for specifying semantics for the graph data model is a major design challenge.

5.3.2.6 Structuring the data

The semantic graph is a huge interconnected collection of data, which needs to be structured to enable meaningful operations for engineering users. The first natural structuring becomes from the semi-structured nature of the data model. Arguably, there is a distinction between the *specification* and *instance* data in such semi-structured information models. A straightforward engineering definition would be to denote the structure as *ontology* and the instances as *models.* Such distinction is not easy though. The structure part can be seen in many cases as *static* while the instance data can be highly *dynamic*. In many cases, the static specifications can contain large amounts of

static instance data such as templates, default values and other specifications. On the other hand, in many practical modeling cases sets of sets are needed such that the specifications themselves are instances of higher-order specifications. Other essential mechanism for establishing structure is *hierarchies*, which can be used to organize and lay on information. A relevant categorization in engineering work is *projects*. The need to identify the set of statements belonging to a given concept or structure is encountered in many places such as import, export, access control and change management. The classifications of description logic can be used to extract concepts, but many times a more general *view* needs to be specified to collect statements according to e.g. a rule-based search. Other possibilities include *contexts* as described earlier and (possible hierarchical) manual or automatic partition of the graph based on some structural criteria. The issues of structuring and fragment identification are regularly encountered in applications and a concise solution for them needs to be included in the design.

5.3.2.7 Defining the standard library

Most programming languages can be seen to consist of the basic language constructs, the standard library and the third-party libraries. In many cases the quality of the standard library and the availability of third-party libraries are a critical factor in determining success. For the modeling and simulation environment, the graph data model with model-theoretic semantics forms the basic language. The standard library is the first layer above the ontological constructs, which can be seen as an upper or domain ontology for modeling and simulation. The concepts selected in this ontology are essential in defining the capabilities of the modeling and simulation environment. The available upper ontologies such as DOLCE, GFO or Cyc supply basic elements for creating such an ontology. A basic problems in these upper ontologies is that the concepts are too abstract for practical use in engineering applications. Possible concepts for the standard library have been identified in research and prototypes associated to this work. *Hierarchies* are useful for structuring the information in e.g. *projects* and *libraries*. *Views* enable to access the graph as a search tree. Data structures such as *lists* and *trees* are required in most modeling tasks. For engineering applications, the concepts of *units* and *quantities* are essential. The vision of the modeling and simulation environment requires that the concepts of *mapping, verification and validation* and *documentation* be comprehensively supported. For

simulation, specification of an *experiment* is needed. Fundamental modeling constructs include *spatio-temporality*, *inexact modeling* and *component-based modeling*. Concepts relevant for extensibility include specifications for *services* and *methods* attached to concepts. Some concepts of user interface and the Eclipse approach can also be included to enable modeling of user interfaces. The selection of a necessary and sufficient standard library with accurate and meaningful semantics is a major design challenge.

5.3.2.8 Prototype case considerations

The issues of navigability were encountered often in the prototype development and both approaches were tested. When bi-directional navigability was introduced, it could be used in numerous algorithms. It was also commonly used in situations, where it was not appropriate. In many cases the inverse of *Instance Of* was used to find certain instances to be presented in user interfaces. In the context of the whole information model, this worked, but when the search needed to be conducted in a context (e.g. a certain project) this approach was not useful since filtering was difficult. The support of bi-directional navigability also included many useless statements. If only some inverses were modeled, most generic algorithms could not be used. A first-class object status was also tested for statements. It was useful in some modeling situations, but made the interfaces and the conceptual model more difficult to people familiar with the OWL model. Testing for statement extensions included statement identifier and time. The statement identifier was very useful in the implementation (it was not exposed in the interface though). When a time period was embedded in the statements, some temporal operations could be implemented directly in the data access interface. Both of these approaches increased the size of the data model though. The prototype implementation used a fully semantic treatment of primitive data structures, which was determined inacceptable as the calculation in [Le07] showed that in many models the primitive data structures accounted for 80-90% of the overall data. Another remark in primitive data handling was that in many cases it was easier to manipulate data decoded into strings or primitive arrays than to use semantic modeling. These problems could be alleviated by generating code for common operations. The foundational semantics of the prototype knowledge representation were close to OWL with a closed world bias. In many cases, it was discovered that engineers found the use of set operations awkward and preferred traditional object-oriented inheritance. The OWL

classification approach was seen to be good for validation (and classification), but not so good for model building. In practical cases the expressive power of description logic was often not enough and in many cases custom codes were needed to perform operations. As expected such codes were in many cases difficult to manage and nearly impossible to introduce to engineering users. In the prototype implementation, the data could be mostly structured in hierarchies of ontologies and models. The problems of a graph model were often encountered when e.g. exporting an ontology or model since there was complex (e.g. mapping) connections between different models. It became evident that some higher-level mechanism is needed to keep the structure of the information model manageable. In the prototype design, no stable standard library could be developed since new requirements were constantly introduced. Consequently, this made modeling difficult since things were constantly changing. The specification of a sufficient and concise standard library is indeed a major design challenge.

5.3.3 Performance and scalability

5.3.3.1 Low-constant performance

The modeling and simulation environment is targeted for industrial applications. Such applications involve a large amount of large models and real time data transfer between the system and some external systems. The issues of performance are closely related to issues of scalability. In many cases, the deciding factor of performance is the scalability of some required operation and the solutions for the challenges of scalability will be required to attain acceptable performance in large modeling scenarios. Nevertheless, in some cases the *problem statement* is inherently complex and requires large data sets to be processed. Examples include real time processing with massive data transfer needs, global analyses such as data mining and statistical analyses. In these cases, even a scalable solution will not provide acceptable performance without efficient and optimized data structures and algorithms with low complexity constants. The selection of a rich and expressive data model has been selected at the expense of performance and this selection introduces significant design challenges in optimizing the system for low-constant performance.

5.3.3.2 Optimization of verbose data

The semantic graph data model is highly verbose. Realistic industrial modeling cases involve millions of statements and the processing of such data masses places high

strain on the memory system. A major design challenge of performance is to internally process the data in some effective form while retaining the original verbose interface. The representation of data needs to be considered in persistent servers, distributed protocols and local client caches. Many compression schemes are based on some form of locality. The graph data model has local connectivity but a coarse-grained partition would also be useful in compression. Possibilities include manual arrangement into files, automatic arrangement into clusters and heuristic arrangement based on modeled concepts. Manual arrangement can prove to be highly tedious work for the user. An automatic partition according to some heuristics or statistics is possible, but there is no straightforward solution available. The use of modeling concepts is problematic since the concepts are often hierarchical and overlapping. An important case for compression is primitive data structures as already discussed related to knowledge representation.

5.3.3.3 Scalability based on used data

A major scalability challenge is to reduce the computational complexity of data access operations such that they are effectively independent of the overall size of the knowledge base. The complexity of operations should only depend on the amount of data needed to perform the operation and the complexity of continuous real time operations should not depend on the selected representation of the data in the knowledge base. The traditional database indexing approach almost fulfils the complexity requirement with the expense of memory and time required to build the indices. In a distributed environment, such an approach also requires a server-side query language such that the transfer of the database can be avoided. An operation truly based on needed data can be established using on-demand lazy indexing or retrieval of data. Such indexing requires a careful selection of indexed information and policies for maintaining the indices.

5.3.3.4 Responding to changes

Most clients need to respond to changes in the data model. Typical clients use the graph data model to build other models (e.g. user interfaces, reasoning services, search services, etc.). If such models are large, an efficient strategy for responding to changes in the graph data model is needed. Possible strategies include notification of all available changes (as statements or higher-level concepts), notification of changes based on used data and notification based on query results. If the client receives notification from all changes occurring in the system, the processing overhead for

determining the relevant changes can be prohibitive. The filtering of changes can be based on the popular *listener* approach such that the client registers notifications or on automatic tracking of a *working set*. The processing and management of such listeners and working sets can be centralized for many clients to ease the processing burden. A problem in these approaches is still to correctly determine how the changes should be reacted to. Notification based on query results is robust but difficult to implement especially for composite navigational queries. Existing implementations (e.g. Rete) of such query tracking show prohibitive memory usage even for a moderate amount of data and queries. Providing efficient and correct means for responding to changes is a major design challenge.

5.3.3.5 Prototype case considerations

The small case study of the bleaching line clearly showed that a naïve straightforward prototype design would not be able to deliver industrial-strength performance needed in large-scale modeling of e.g. nuclear plants. An unoptimized representation of the data model consumed too much space and processing even relatively small models became intractable whenever simple, but inefficient algorithms were used. The implication is that the data structures and algorithms of the modeling and simulation environment need to be carefully designed and implemented with little leeway for compromises or bad design. The challenge of the verbose data model was clearly demonstrated. As was discussed before, in the prototype case the different activities required different parts of the data, thus making a lazy indexing approach attractive. In the prototype design, client-side queries with lazy data retrieval were implemented. A statement-level granularity of data retrieval from the server was obviously found out to be inacceptable due to round-trip delays and ways to enlarge the granularity were much discussed, but no suitable candidate could be implemented for the prototype. The issues of responding to changes were encountered often in the prototype. Most user interface and simulation codes needed to track changes in the graph model and different schemes were implemented. As was described in the challenge, a complete query dependency tracking of Rete consumed too much memory even for small query sets and the listener approach was hard to use correctly. Responding to changes was a major burden of implementation for all client applications and needs to receive special attention in the design.

5.3.4 Mappings

5.3.4.1 Specifying an ontology and a methodology of mapping

The goal of the mappings in the system is to exploit the similarities between related models such that changes in one model can be reflected to related models when applicable. This enables e.g. the automatic generation of a simulation model based on design data or the synchronization of the results of a steady-state process simulator into a dynamic simulator. Such mappings can always be implemented in an ad hoc manner, but as the mappings are a focal concept in the vision of the modeling and simulation environment, a concise methodology is needed for specifying the mappings. The first thing to note about mappings is the distinction between *mapping specification* between concepts and the established *correspondences* between specific instances of these mapped concepts. The mapping specification specifies in which way the concepts are related to each other and the correspondences are used to indicate the application of the mapping. The most usual mapping is a one-to-one mapping between two concepts, which is also easiest to manage. The specification and use of the mappings becomes more difficult if the mappings are defined between sets of concepts. The direction of the mapping is also important. The mappings can be either unidirectional or bidirectional. An invalid unidirectional mapping is easier to make valid than a bidirectional one. For example, a (bijective) functional mapping between values can be made valid by modifying either end. If the mapping is not bijective, a solution might not exist altogether. Correction of bidirectional mappings usually requires knowledge about actions leading to the invalid state, but for constrained mappings even this might not be enough. When a mapping between corresponding models becomes invalid due to changes, the mapping needs to specify means to correct the situation. Possible specifications include the *algorithmic* approach and the *declarative* approach. The algorithmic approach defines a corrective algorithm to perform given changes. The declarative approach is based on rules such as equations. In many cases the declarative definition provides an easy way to correctly *specify* bidirectional mappings. A natural declarative specification takes the graphs as input and produces a Boolean *indicator* of the validity of the conditions of the correspondence. Such a definition does not necessarily offer means for making invalid correspondences valid and sometimes some algorithms might be needed. An important aspect of the mapping method is the discovery of correspondences. In most situations, correspondences between models are

dependent on the existence of other correspondences e.g. the correspondence between pipes P_1 and P_2 in a static and dynamic process simulation models implies that there also exists a correspondence between flows f_1 and f_2 of pipes P_1 and P_2 respectively. The design challenge involves formulating a necessary and sufficient ontology and a clear and concise methodology for performing mapping in the modeling and simulation environment.

5.3.4.2 Enforcing the mappings

Initially the enforcing of mappings means keeping all established correspondences valid as changes are introduced. This requirement is made slightly more complex by several other considerations. In many cases, automatic enforcement of mappings is forbidden. This is especially the case when the corresponding models are both created and managed by the user. In such cases, the correspondences should be identified as in need of repair. Similar situations occur when there is no automatic way to correct a correspondence. In some cases, only the user can correct an invalid correspondence. Enforcing the mappings is not cheap and processing of mappings in e.g. large amounts of read-only models is not feasible. Thus, a mechanism for activation and deactivation of mapping correspondences with relevant granularity might be needed. There is also a trade-off between the performance given by explicit modeling of correspondences and the space saved by implicit (reasoning) of correspondences. In many cases the mapping specifications would allow for almost completely implicit correspondences between large models, but reaction to changes in such situations can be difficult. The possibilities range from fully persistent explicit modeling of correspondences to fully implicit runtime resolution of correspondences. The mechanism required to convert information about changes to actions enforcing the mappings also needs to be consistent with mechanisms for responding to change in other situations.

5.3.4.3 Usability

The usability of the mappings is essential for acceptable user experience. For the end user of the modeling and simulation environment the workings of the mappings should not be visible whenever automation can be used. The activation of mappings should be tied to other user actions such as opening and closing of a model. If the mappings cannot be performed automatically, the user interface needs to present the possible user actions clearly and intuitively. The definition of the mappings is a lower level task, but should be based of declarative data-driven implementations whenever possible because

in many cases building robust enforcement strategies is a tedious task. These data-driven mapping implementations can be presented to the mapping designer using a graphical user interface. The mechanisms for establishing correspondences between instances of mapped models can be generalized to address also more general *issues* concerning models with some attached strategies for resolving them. Such issues include *formal validation* with correction strategies and markers with strategies for improving the model in some way. The issues can be established by decision support services or by human users and the strategies can involve automatic or semi-automatic means or informal instructions etc. Like the mapping correspondences, such issues can depend on each other. Such generalization could simplify the user experience by reducing the amount of mechanisms used for similar tasks, but as generalization is always a double-edged sword its application needs to be designed with extra care.

5.3.4.4 Prototype case considerations

During the prototype design and implementation, two methods for performing mappings were evaluated. The first method included a Rete rule engine with an attached Prolog interpreter and some associated mapping rules. Performing Rete at statement-level granularity was found out to be too expensive and a larger granularity was closer to simple resource modification listeners. The approach was successfully used in small-scale 3d modeling though. The second approach used a custom Java code, which handled reaction to changes in the model and performing the mappings according to specifications in the graph. The approach was used in the prototype case with success, but in other developments it was shown that developing and maintaining such mapping codes was too difficult and time-consuming.

5.3.5 Real time processing

In the context of the modeling and simulation environment real time processing means manipulation of fast-changing data produced by e.g. simulators and consumed by e.g. graphical visualizations, loggers etc. Although the environment can have connections to live industrial systems, such systems are treated as *soft real time.*

5.3.5.1 Representing real time calculation data

Modern simulator tools are capable of calculating thousands of values many times a second. The system needs to be able to store and process and to transfer such values between different simulators. The requirements include full-scale simulation of the

control system of a nuclear power plant including tens of thousands of signals in real time or faster. In operator support applications, the system also needs to be connected to measurement systems including sensors and process equipment. The first design challenge is the representation of such real time values in the semantic graph. In many cases the treatment of real time values in simulation experiments is not tightly connected to the graph data model and the real time requirements for transactions with real time values are stricter than for the semantic graph, where delays are more acceptable. The version control features of the semantic information model are also not relevant for the real time values. Nevertheless, the real time data should be seamlessly accessible from the graph data model. In many modeling cases, it is also important to be able to store multiple configurations of values corresponding to different interesting states of the model. Many calculation codes prefer to access the data in a tabular form with no structural overhead. These challenges can be addressed by providing the real time primitive data stored in the system with *identity* such that values *specified* in the semantic graph can be also stored and processed in other forms. Essential requirements include several static *configurations* and several dynamic *states* for the values. In many cases the nature of a value (static or dynamic) can not be specified before simulation time, which makes the management of such values harder. The storing and accessing of historical data should also be seamlessly supported for relevant values.

5.3.5.2 Transferring real time calculation data

The wide range of functionality of the modeling and simulation environment creates a wide range of requirements for transferring real time calculation data. The typical requirements of data transfer between calculation components involve transferring large predefined sets of values periodically between components. The transfer is usually done in a synchronous way. User interfaces require data transfer for monitoring and visualization and the selection of transferred values can vary from single value to a large collection and can also dynamically change during simulation. The real time requirements for user interfaces can vary greatly. Some can tolerate dropouts, and some require buffering. Usually asynchronous operation is enough. The usual requirements for data loggers include a fixed set of values asynchronously with provision for time-stamps and no acceptance for dropouts. The semantic *identity* for values enables them to be processed and transferred outside the graph data model, but the synchronization of these containers and the graph data model need to be handled in a robust manner.

Existing standards such as OPC DA [OP] have been successfully used in many industrial modeling projects, but the performance of such COM/DCOM or XML-based systems has been a challenge. The newer OPC UA [OP] specification could provide some answers to distributed data transfer. For local deployments, the most effective implementation could involve shared memory between calculation components, but designing an interface suitable for distributed and shared-memory use is challenging.

5.3.5.3 Real time modification of the graph data model

In most applications encountered during this work, the semantic graph data model can be seen as a slowly changing model *configuration*. Such an assumption can be utilized in the design of the environment. There is also another class of applications e.g. simulation, optimization, search, which would want to make fast real time modifications to the models. At least two reasonable alternatives to this problem can be considered. The initially most straightforward solution is to replicate the semantic graph model into a fast runtime data structure in a suitable programming language and perform the operations using these structures. In the general case, a two-way synchronization of changes between the semantic graph and the runtime structures is needed. Manual implementation of such structures is tedious work but some automatic support could be possible to implement. Clearly, such a custom solution would provide best performance for algorithms in most cases. Another approach is to provide a custom real time version of the semantic graph model with some performance-enabling constraints. The graph interface would provide a familiar programming environment and at least some of the services built upon the semantic graph could be utilized.

5.3.5.4 Prototype case considerations

The real time requirements of the prototype case were limited to infrequent calculation of process variables and tracking their state in the user interface. The prototype design included ways to describe sets of real time variables in the semantic model and to process such sets with dedicated interfaces. Some issues encountered during prototyping included the difficulties of specifying and managing the sets – especially what to include in the sets, how to present them to the user and how to deal with the versioning of configurations. In accessing and transferring the simulation results the difficulties of different use cases were encountered with multiple user interfaces requesting small sets of variables with best effort delivery and a data logger requesting a predefined set of time-stamped values whenever changes were made. The

straightforward approach of making all variables available at all times worked for such a small model, but the scalability of the approach was not solid.

5.4 Imminent design challenges

5.4.1 Verification and validation (V&V)

5.4.1.1 Specifying an ontology and a methodology for verification and validation

The projected range of validation needs was already discussed in the technology review. The significance of V&V in the adoption of simulation in engineering needs to be stressed as most models of reality are only applicable given some very specific conditions. Thus, traditionally accurate simulation of complex and complicated phenomena requires a combination of wide theoretical and domain-specific expertise. People with such expertise are not available in large enough quantities for widespread adoption of state-of-the-art simulation in industry to happen. The only way to bring simulation into the mainstream of engineering work involves more automation in production and V&V of simulation models such that the engineer can still trust the results given by the models. The semantic models offer considerable theoretical possibilities for automatic validation of model configurations using formal specifications. Various methods of model verification such as testing agains measurements can also be supported by semantic specifications and configurable experiments. Essentially, the realization of such services requires a clear and concise ontology and methodology of V&V, which needs to be comprehensively supported by all processes in the modeling and simulation environment. The issues of formal validation are closely related to issues of mapping and such synergies should be exploited in their design. A general formulation of validation consists of a set of conditions, which the models need to satisfy. Such conditions can range from declarative and algorithmic specifications to informal textual specifications, but formal specifications (equations, logic etc.) should be preferred whenever possible and their semantics exposed for utilization in other decision support services. The benefits of using formal and declarative languages include a clear and concise notation, which aligns well to the graph representation model, formal semantics and availability of existing reasoning tools. On the downside, logical formulations cannot well capture many relevant issues e.g. related to algebra and thus some general programming language might be needed to cover such cases. Another essential requirement is that

when a formal validator detects an issue it should be able to provide some *explanation* about the causes of the issue. If applicable, the validator should also be able to suggest some *correction* strategies to deal with the issue. As the range of V&V specifications is large (from informal text to test runs of complex experiments) the ontology and methodology must be highly versatile and extensible while providing a generic foundation enabling user interfaces and decision support services to interact with the V&V process.

5.4.1.2 Enforcing verification and validation

Enforcing V&V is again closely related to enforcing the mappings between models. Relevant issues include efficient reaction to changes in models and the supporting semantic structures and control over automatic and manual V&V with relevant model-level granularity. The cases for manual V&V are greater than for manual mapping since continuous timely execution of V&V is not as critical as timely mapping in many cases. On the other hand, the results of V&V are significantly more interesting for other users than the workings of mappings. The results of V&V can be actively used to prevent unwanted situations such as publishing or simulation of broken models. The user interfaces should also be able to visualize the V&V issues in a meaningful way. For verification support, the automatic means are more incomplete than for formal validation tasks. Some verification can be obtained using automated testing, which requires strong framework and user interface support. An important aspect of high-level V&V is informal issue management (e.g. design rationale), which also needs strong support. The informal and formal issue management has again strong synergy with tacit information management discussed later.

5.4.2 Extensibility and evolution

5.4.2.1 Specifying a methodology for extending the environment

The basic deployment of modeling and simulation environment can only provide basic means for building meaningful engineering applications. The services provided by the system can be augmented by defining new ontologies, decision support services and user interfaces. The environment should provide powerful generic tools for creating new ontologies. In domain-specific cases, also some specific tools are needed to create catalogues and other data-intensive ontologies. The creation of an ontology is only the first step in producing a meaningful extension to the environment. The use of the

ontology must be enabled through services for simulation, verification and validation, decision support and through user interfaces. The requirements for writing code should be minimized to maximize the productivity and quality of the extensions. Writing code can be averted by using existing service interfaces and components and by using automatic code generation [KJB+04] based on the semantic specifications. The packaging of related ontologies and codes should be consistent with the OSGi plugin model used in the Eclipse platform. A main challenge of the extensible decision support is to create a decision support services architecture with clear interfaces and possibilities for reuse. The relationship between ontology and services should also be exploited so that the application codes can be semantically integrated into the information model.

5.4.2.2 Specifying ways to publish and acquire services

The Eclipse platform provides means for publishing and acquiring components via *update sites*. The mechanism can be used to load new application plug-ins but does not address the deployment of associated ontologies. This brings about a design challenge since such existing solutions should usually be preferred. Another supply channel for plugins and ontologies is the distributed semantic graph model where components can be centrally installed into the information model so that they are available for all users of the environment. Essential use cases also include local deployments of extensions so that they will not be shared with other users of the information model. Both cases can be served in the modeling and simulation architecture by setting up component servers from which the components could be copied. This approach would also enable access to the components without loading them into the local information model.

5.4.2.3 Dealing with evolution

The semantic models and associated software components will inevitably evolve as requirements change and issues are fixed. The changes introduced to components need to be reflected to their dependencies including other model configurations, software components and user interfaces. The design of the modeling and simulation environment needs to acknowledge this change by including a change management process with a well-defined ontology and methodology. The challenges include a robust and usable way of resolving the issues encountered when dependent components are changed. In most cases, the changes can be done in a tracked manner. Additionally, in some modeling projects multiple versions of a component might be needed because

of complex dependencies to other older or newer components. The verification and validation services available in the environment should be designed so that they can be used to automatically detect and correct issues related to ontology and model evolution. Such services are especially suited to dealing with declarative and data-driven modeling. For software components, change management can benefit from automatic code generation and good compilers.

5.4.2.4 Specifying a framework for simulation

A fundamental goal and purpose of the modeling and simulation environment is to provide better tools for decision support in computer-aided engineering. Simulation is an essential tool for decision support but other fields such as reasoning, optimization, and data analyses offer major possibilities. The challenges of simulation support include the challenges of real time data and integration of simulation models but also a framework for specifying *experiments* involving different models in different states and possibly multiple calculation codes and live systems with complex data exchange, synchronization, and measurements. A key element of such an environment is the management of the life cycle of the calculation components and the communication between the different systems involved. The approach adopted in HLA and CAPE-OPEN is to use a coordinating executive, which orchestrates the execution of the experiment. Such an approach simplifies the interface specification of the calculation components so that components only need to be able to talk to the executive. When the experiment includes significant data transfer and messaging the routing of the data through the coordinator might not be effective. The design of the modeling and simulation environment should include an ontology and a methodology for supporting simulation experiments. The specification of experiments should be highly accessible for engineering users. Possible models include flowsheets, state machines and Petri nets, which can be represented in graphical form.

5.4.3 Security

The requirement for information security is essential in establishing trust when multiple possibly competing companies are involved in a modeling project using a centralized information model. Security involves extra management and incurs some performance losses.

5.4.3.1 Choosing an access control model

The main design challenges include the selection of a proper access right model for the graph data structure and the implementation of security between servers in the system. Most contemporary access control schemes rely on some hierarchical structure of the data. In a graph data model there can be alternative paths to a given location and blocking one path will not ensure that the location will not be accessible using some other path. Generally, the propagation of rights in a hierarchical manner can lead to conflicting rights and management of such conflicts is difficult. The amount of access specifications is also potentially large due to the homogeneousness of the graph data model. Traditionally rights have been attached to large entities such as files and thus the amount of access right definitions has been manageable. A solution for such subgraph identification could be highly useful also in other areas as discussed. An efficient solution to the access control model could also benefit from low-level extensions to the statement model such as *contexts*. In contrary to usual file system etc. access control, the performance of access control in such semantic networks is a challenge because of the lack of a natural coarse-grained structure. The concepts of access control e.g. users and roles should also be designed so that they can be used to establish intuitive task-oriented user interfaces.

5.4.3.2 Enforcing security

In the distributed architecture, the security needs to be handled in the data transfer interfaces. A server should not be able to receive any data, which its users should not be able to access. Different means for providing this functionality include encryption and filtering of data based on access rights. The encryption model can be made virtually secure, but with sufficient processing power the encryption might be broken and even the availability of encrypted data contains lots of information. A filtering approach is a performance challenge and involves design of associated data structures and algorithms. The issues of encryption and identity between servers can be dealt with standard solutions such as Public Key Infrastructure (PKI), which requires consideration about a trusted authority. The client interface needs to deal with similar issues as the distributed servers. If robust security is required, the client must only be able to perform operations according to the credentials it will supply to the server. If a client uses a local server the client interface does not need to care about security at all since the local server is accessible by the user in any case. Still in many cases support

for such soft security implemented in the client side could be useful for example in administrative duties for testing the access rights. The issues of trust become important, when the deployment of services provided by the environment is not completely controlled by the user. Such cases can occur when connecting to external systems or when some remotely deployed environment services are used. In such cases some validation of identity should be employed. The technologies of PKI should be preferred.

5.4.4 Reuse and redundancy

5.4.4.1 Specifying a methodology for reusable model configurations

Reuse is a persistent challenge in all software and engineering systems. Most engineering work is based on some form of reuse of previous designs. Paper-based documentation reuse has involved tedious copying of old documents. Computer-aided design has potential to streamline this work significantly but in many areas the reuse aspect has been neglected and copying old designs is laborious. Model reuse is an essential enabler for widespread adoption of computer-aided simulation. A major design challenge of reuse involves the design of a methodology specifying mechanisms to encapsulate and offer existing models so that the designers can use a simple building blocks approach to build models. This kind of reuse is both time efficient and reduces modeling errors. Traditional approaches for supporting software reuse include the *object-oriented* approach and *software components*. Both approaches are based on a more profound principle of design based on interfaces. Useful interfaces for information models include a parametric interface and a connectional interface. Such approaches have been used in the information model of ProServ and in the modeling language Modelica. The interface parameters can be used to calculate values for hidden parameters included in model structures and some interface parameters can be calculated as model outputs. A parametric interface can also be used to generate substructures for a model. In a graph data model the connectional interface approach can be used to construct large graph models by connecting smaller graphs. In such connection approach, a single subgraph model could be used many times in a larger model or models. If the models only refer to their subgraphs, some significant space savings can be achieved. In such cases, modifications to the subgraphs become difficult. If the subgraphs are directly modified the modifications will be reflected to all models based on these subgraphs. If such side effects are not desirable, a clone of the

subgraph can be used. In that case the connection to the original subgraph and its modifications will be lost if the connection is not explicitly saved and maintained. Another approach is to model modification operations to a subgraph and to include such operations wherever subgraphs are referenced. This approach is used in Modelica.

5.4.4.2 Discovery and management of reusable model configurations

A significant design challenge concerning model reuse is the management of the reusable models. Some means are needed to create, publish and discover such models. Preferably, the modeling process should automatically produce reusable models with interfaces. Traditional approaches for publishing such models include component libraries or catalogues. In the graph data model, reusable subcomponents of models could also be directly referenced but the dependencies created between models could prove problematic. In many cases, the discovery of suitable reusable components can be more effective if some descriptive data is included with the component. In many cases, these problems can be addressed by suitable processes and conventions built on top of simple catalog services. Still it is essential that such processes are part of the design of the modeling and simulation environment so that the users have a clear methodology supported by the environment.

5.4.4.3 Design for building reusable software components

The concept of reuse is also highly relevant for services and operations defined for the semantic models. In most modeling cases a clear separation between models and algorithms increases the reusability of both. Such separation again requires well-defined *interfaces* between the models and the algorithms. These interfaces can then again be attached to the semantic models using an object-oriented method approach. Such a connection makes the the operations applicable to the models accessible. Such operations can range from simple data processing methods to complex autonomous services. Attaching software component implementations to the interfaces requires uniform means for discovery such as supplied in OSGi or other SOA schemes. The methods can also provide capabilities for user interface. The design of user interfaces can benefit from reuse by using a small set of generic user interface means such as tree browsers and flow diagrams. Another possibility of user interface reuse includes attaching simple user interfaces such as dialogs and wizards to the information model as methods.

5.4.4.4 Design for reduced redundancy in reuse of model configurations

Although semantic modeling techniques can introduce larger amounts of raw data than more primitive representations the semantic information can in some cases be used to make significant savings by reducing redundancy. Utilization of copy-based reuse approaches generates large amounts of redundant data, which could be eliminated using template-based approaches. Similarly redundancy can be reduced using models generated from parameters if the generation of models can be deferred until the actual use of the model. Such redundancy is also common in application code and can be reduced by utilizing services.

5.4.5 Life cycle information management

Life cycle information management has become a major requirement in important modeling domains such as construction, manufacturing and capital projects industry involving operation and maintenance.

5.4.5.1 Specifying support for spatio-temporality

The vision of the modeling and simulation environment calls for temporality on three separate dimensions. The first dimension is the *life cycle of the information model* per se as recorded by the versioning knowledge base. This dimension does not represent the temporality of the model itself rather than the development of the model. The second dimension is the *life cycle of the modeled system*. This involves representation of the temporal evolution of the modeled system. The third dimension is the *simulation time*. The dimensions are generally orthogonal, but some relationships between the dimensions can be established. The second dimension can be used to model the first dimension so that the development of the model would be contained in the model. Importantly the activity in the first dimension concentrates on the latest model (head revision) and the history is interesting only in special cases. In the second dimension, all parts of the temporal range are equally important. The simulation runs in some configuration of the temporal model. In some cases the model temporality plays an important role in the simulation and the simulation time needs to correspond to the modeled time such that the simulator will respond to the temporal evolution of the modeled system. The concept of spatio-temporal modeling has recently been advertised e.g. in construction industry. Essentially the concept involves three-dimensional modeling of time and space where the spatial model can be a function of time. The most common engineering requirements for temporal modeling include representation

of *events* and *durations*. More generally, the semantics of temporality include treatment of relational issues such as causality and consequence common from natural language [MS98b]. The treatment of time should be able to address the usual needs of applications while maintaining the ability to model issues that are more complicated. From the application point of view, common questions include: *What is the (non-temporal) state of the model at time t? What are the temporal events between times t1 and t2? When was this item added?* Most software would prefer to see the model through a non-temporal snapshot. Most tools in fact do not wish nor need to address temporality at all. With a non-temporal snapshot, the evolution of a model can be processed as sets of changes applied to the snapshot. In the temporal snapshot, the properties of events need to be provided as queries (e.g. *When does this relationship between concepts expire?*). The modification of such as temporal model is much more difficult. Addition and removal in the temporal snapshot is not well defined i.e. a removal operation can remove a statement altogether or only set its lifetime to expire at the current time. Similar questions and operations should also be supported for primitive data. If an application is more knowledgeable of the temporal model the application might also want to process the model in full temporal form such that the full (or partial) temporal evolution of a concept is available at once. Such software includes analysis and reasoning such as modal logics. A temporal graph model is well suited for representing discrete-time spatial models, but for continuous-time spatial models, the spatiality needs to be established with equations.

5.4.5.2 Representation of spatio-temporality

The main temporal representational challenge concerns the second dimension of the modeled system life cycle. The possibilities for representation include at least two major approaches. The low-level approach assigns a period of lifetime for each statement (and primitive value) in the system. Such an approach provides for efficient implementation of low-level operations for handling non-temporal snapshots already in the client interface using optimized algorithms and data structures. The approach models most events implicitly, which can cause modeling problems (e.g. the creation of an item can be inferred from the creation of its defining statements). The high resolution can also cause problems since in most cases the lifetimes of larger concepts are of interest. The second approach widely adopted among ontology research such as DOLCE, GFO or ISO 15926 Part 2 models the temporality at the upper ontology level.

Such models can process all aspects of temporality in full semantic form with the expense of performance (i.e. building the snapshot). Even in the low-level approach, some semantic (graph) constructs are also needed to model the higher-level semantics of temporality. Similar statement-level optimizations are also possible for spatial operations on the model, but the benefits for the modeling and simulation environment have been considered smaller and this option has been discarded from consideration.

5.4.5.3 Supporting spatio-temporality in user interfaces

Temporal modeling requires comprehensive support from the user interface. Essential user operations involve browsing a non-temporal snapshot and moving the snapshot in modeling time, visualizing the temporal events of a model and modifying temporal statements. Major challenges include the handling of creation and deletion operations and the management of temporality for complex structures. The integration of temporality processing with other aspects of modeling is also a challenge since many tools do not process temporality at all and some user interfaces wish to process it completely on their own. Nevertheless, the tools for managing temporal issues should work seamlessly with all other tools in the environment. For spatial models, the usual user interface requirements include cropping of data based on spatial regions, which should be supported generically if the concepts of spatiality are included in the standard library of concepts.

5.4.6 Inexact modeling

There are cases, when relevant information cannot be expressed simply by saying that a relation exists or a variable has some specified value. The principal cases include *uncertainty* and *vagueness*. The major design challenges involving inexact modeling include

5.4.6.1 Specifying means for inexact modeling

The treatment of uncertainty is an area widely neglected by contemporary modeling and simulation tools. The vision of highly automated model configuration and advanced decision support services can not be fully realized without a proper account of uncertainty because many relevant parameters of real world systems are not known exactly. Inaccurate models, inaccurate model parameters and noisy measurements from real time systems introduce uncertainty. Such issues can be captured in probability distributions, but general probability distributions can not be represented in reasonable

space and operations on probability distributions are computationally intensive. A robust treatment also includes the possibility of *ignorance* such that the probability distributions might not be normalized (i.e. we have only measured that the probability of event *A* is 0.5 and have no other knowledge). There are multiple ways to approximate probability distributions and also some more ad hoc methods such as interval arithmetic. Such probabilistic models can be processed using various Bayesian approaches and ignorance can be addressed in e.g. the Dempster-Schafer approach. The issues of vague information can be dealt with fuzzy logic with e.g. a degree of truth assigned to statements. A majority of existing calculation software can only deal with exact values. If the environment would include comprehensive support for some inexact representations there would have to be also methods for translating these representations to exact ones. Such backward compatibility can be a deciding factor in the adoption of new technology. The full benefits of inexact modeling could be realized only if relevant software could be translated to operate on the inexact representations. For primitive value calculations, an inexact representation with straightforward algebra cannot be very expressive generally. On the other hand, any measure of inexactness such as a *confidence interval* would be a major improvement in simulation software. An inexact graph model with easy translation to exact models seems extremely difficult, but some approaches such as “defuzzification” of fuzzy models could be considered. The design challenge involves making selections about support for some of these approaches and establishing ontologies and methodologies for endorsing their use in applications. All of these technologies can also be modeled and applied on higher layers of the environment but committing to some technologies could help to create standard processes for dealing with many issues of inexact models.

5.4.6.2 Designing support for inexact modeling

If the environment made a commitment to some fundamental representation of inexactness, the representation could be included in the statement level (e.g. scalar fuzziness) or in the primitive data representations (e.g. probability measure with n moments). The representational trade-off is again about expressivity against performance. Especially for primitive data, a low-level representation could be more efficiently processed than a semantic representation. On the other hand, the inexact treatment is not needed or applicable in many places even if inexactness were comprehensively supported. If inexact modeling is not supported on the data structure

level, it can still be included in the standard library of concepts and its support can be a requirement for all general tools for the environment. A commitment for inexact modeling obviously requires comprehensive support from the user interface. Uniform and novel means for creation, representation and visualization of inexact models is needed for the representational technologies supported by the modeling and simulation environment. Such means should also be readily available for higher-level functionality implementers.

5.4.7 User interface

5.4.7.1 Establishing an user interface methodology based on user needs and goals

The intended user groups of the system range from simulator programmers and modeling and simulation experts to plant operators and other groups of users who can be considered nonprofessionals in modeling and simulation. In most cases, the user performs tasks with very specific goals and requirements. Addressing the various use cases requires means for *task-orientation* and generally *context-dependent* user interfaces. Task-orientation requires means for building custom yet accessible user interfaces. As the amount of different use cases is large, the customization should be based on some form of configuration or reusable components. In the selected Eclipse environment, the included means for task-orientation include *perspectives* and the *Mylyn* task-focused UI. Any additional mechanisms should consider their relationship to these existing technologies. Context-dependence requires means for customizing the user experience also based on users and their roles. Such customization can be based on user preferences and history of use. Context-dependence should also be available for important modeling concepts such as projects and models.

5.4.7.2 Supplying advanced editing tools

The software industry has a long-standing dispute about the best way of doing user interfaces. Large companies such as Microsoft and Adobe have driven the graphical what-you-see-is-what-you-get (WYSIWYG) approach. Traditional UNIX-platforms have preferred a text and command-based tools. The virtues of graphical user interfaces include accessibility, intuitivity and visualization of results while more advanced users can often feel frustrated by lack of fast and configurable operations. The text and command-based interfaces can be hard to learn, but can provide high productivity for experienced uses. Providing the best from both worlds requires strong frameworks for

building graphical tools, high configurability for keyboard and mouse input along with means for textual specifications and scripting. A major challenge of user interface is to provide *frameworks* for building powerful task-oriented and context-dependent user interfaces with best features from graphical and textual worlds. Such frameworks should enable straightforward support of all modeling features included in the standard library of modeling concepts and they should also be extensible so that new modeling methodologies could also be introduced.

5.4.7.3 Clarity and consistency

A major usability issue with task-oriented and context-dependent interfaces is the clarity and consistency of the provided tools. If the tools associated to different tasks differ greatly from each other the user will lose time in learning the different approaches. The design challenge involves specifying frameworks and methodologies of user interface in the modeling and simulation environment so that there are agreed upon and unambiguous processes for performing similar tasks i.e. the user interface of the environment is *predictable*. Reusable generic components can assist in creating a consistent user experience. Such components include editors for list- and tree-form data and modeling in 2d and 3d. Such generic components should fully cover all aspects of modeling included in the standard library of concepts and be extensible. The design and implementation of such generic components is challenging because of the high requirements of reusability. In many cases these elements will not be sufficient and new components need to be created. The design of the environment should include comprehensive specifications for building such components with same high quality as the standard components.

5.4.8 Documentation and tacit information management

The issues of technology transfer and the so-called tacit information stored only in the thoughts of the engineers are major causes of economical losses for all organizations.

5.4.8.1 Specifying processes of extracting tacit information from the users

The vision of the environment design includes that rationale for all decisions made during a modeling project should be present somewhere in the information system. Even better would be if the rationale could be interpreted by the modeling system and could be used for automatic decision support e.g. model synthesis in later stages of the model life cycle. The challenge of extracting tacit information should thus be addressed

in the overall design of the system. The extraction of textual documentation from users is straightforward to support, but in many cases, the task of documentation will be neglected. A common solution is to require that some documentation be produced. Such an approach without further motivation usually does not provide the best quality documentation. The best motivation to supply documentation comes from social mechanisms and other rewards. If the system promotes teamwork, it also promotes technology transfer. The ability to gain decision support services based on the acquired documentation could also be an important driver. The proposition to acquire information from communications has been around for a while and is usually dismissed due to issues of privacy. More acceptable but similar means include social media and issue-based information systems.

5.4.8.2 Specifying support for making effective use of documentation

A traditional representation of design information has been more or less informal textual documentation attached to pieces of information. The usefulness of such documentation can be increased with clear guidelines about what to include in documentation as e.g. in [Ja06]. Some information is directly related to models and some is about processes such as *best practices* and *faqs*. In the semantic information model, such textual information could utilize techniques similar to hypertext linking to provide semantic connectivity of documentation. Another challenge involves endorsing more formal representations for technical documentation that could be accepted and adopted by the engineers using the environment. Such documentation is closely related to verification and validation, which can be seen as formal documentation of the system requirements. This synergy should be exploited in the design. The production of documentation itself is not enough and the environment needs to provide powerful means for utilizing the collected information. A design challenge involves finding ways to maximize the benefits from supplying the documentation. Possible means include effective means of finding and managing informal documentation and offering means to use formal documentation in verification, validation and other decision support services.

5.4.9 Future-proof design

The only constant in software development is change. This is true now more than ever. This is further emphasized in platform development. Additionally, the life cycle of industrial software is considerably longer than for consumer software. The life cycle of

an industrial plant can range well over 50 years. Designing software to last for such long time spans is not possible with modern rapid advances in technology but the change should be an acknowledged part of the overall design of the environment.

5.4.9.1 Design according to current trends and signals

The design of the environment should take into consideration some emerging megatrends including [RT05]

Rapidly increasing computing and graphics performance
Transition to parallel programming
Ubiquitous computing, connectivity and information services

The pace of the performance increase is currently so great that applications can turn from too computationally demanding to trivial in a matter of five years. It is thus important to design software so that it will be able to use all available processing power at release time and to continue to evolve to meet new user goals as performance increases. Lately the progress of graphics performance has been even greater than computing performance. The general software industry is only recently catching up with hardware-accelerated user interfaces such as the Windows Presentation Foundation and Mac OS X Leopard. The availability of such power should be addressed in user interface design. An emerging megatrend in software engineering is parallel programming. Leading hardware providers have made commitments to multi-core processor technologies, which require a completely new approach to software. Making effective use of parallel processing requires careful overall system design. The increasing connectivity and ubiquitous computing trend has several concequences. It will make distributed communications more attractive and also require flexibility of deployment for clients and services. It also emphasizes capabilities for communicating with external systems. Slightly weaker signals can also be identified in the technology field related to the modeling and simulation environment. The field of user interfaces is experiencing a shakedown with strong cases for *multimodal* user interfaces and *augmented reality* in industrial information systems. The continuous strong investment in AI technologies for user interfaces and decision support should also be accounted for.

5.5 Analysis

The issues encountered in the discussion about design challenges cover a wide ground. All issues are of practical importance to the design and implementation of the modeling and simulation environment, but some of them can be treated with special consideration. Such issues recurring in the discussion about design challenges include

Managing the distributed information model
Operations provided in the client interface
Real time requirements and transactional needs
Stateful models responding to changes
Declarative management of issues
Extensible component and service architecture
Focus on user interface

The designs selected for the deployment and management of the distributed information model have significant implications for the interfaces provided for the clients of the system. The design of the clients themselves is highly dependent on the specification of operations provided in the client interface. The implications of requirements for both real time and transactional communications are essential in designing the interfaces and data structures needed in the environment. The issues of reacting to changes are encountered whenever information is refined or cached and uniform solutions are instrumental in all client application development. The essential areas of mappings, validations and decision support have similar requirements about declarative management of issues. These issues are also closely related to formal and informal documentation. The pattern of layered components and services is encountered in multiple areas. Well-defined software service architecture is a deciding factor in establishing the system's extensibility and applicability to integration. The need of powerful user interfaces was also a recurring issue. All in all the design challenges of a modeling and simulation environment involve a complex interdependent set of design decisions between different approaches and trade-offs. The decisions include architecture, processes and data structures and algorithms. Even an initial design needs to consider solutions for most of the discussed challenges, which makes the design itself a challenge. This challenge was encountered in protype designs and implementations related to this work. Such a complex task can tempt the designer

to choose overly generic and abstract solutions, which may not fully match the requirements and may be difficult to implement efficiently. In addition to traditional software engineering, the design of such an environment requires comprehensive effort at establishing processes and methodologies specifying the intended use and best practices in this environment.

6 Conclusions

6.1 Contributions

This work is part of a larger strategic effort at VTT to develop a modeling and simulation platform capable of addressing various needs of simulation based engineering at the industry. This work can be seen as an interim statement of an ongoing development started in summer of 2005. Some parts of the material analyzed in the work are direct implications from previous research on modeling, simulation and information systems at VTT. Some parts are based on related research in VTT during these two years in different research projects. Connected to this work many of the issues discussed in this work have been prototyped and further studied. The contributions of this work are shortly discussed in the following sections.

6.1.1 Updated vision of the modeling and simulation environment

The vision of the modeling and simulation environment has been developed and matured during many years of research and application of modeling and simulation techniques at VTT. This work has taken these ideas, added some new ingredients and formulated an updated version of the vision. The vision is by no means a result of this work, but this work has contributed to update and document the vision and to bring it a step closer to reality.

6.1.2 Technology review

The technology review in this work covers a wide ground of modeling and simulation – related technologies. For designing a modeling and simulation environment, much less is not possible. The importance of the review needs to be underlined since in regular development projects such wide literature studies are often not possible when concrete results are wanted. The information collected for the technology review can be further utilized in developing the modeling and simulation environment.

6.1.3 Design challenges

The actual design and implementation of the modeling and simulation environment has already started at the time of writing this document. Sitting down and reflecting the vision, related technologies and completed prototypes has been important to identify the essential from the whole.

6.1.4 Documentation of performed research

During research and prototyping, documentation is easily neglected when aiming at a moving target. This work provides a partial account of the work performed by a research group during the last two years at VTT.

6.1.5 Suggestions for the design

In the architectural design of the modeling and simulation environment the use of proven paradigms and patterns can be beneficial. In this work two architectural patterns can be applied to solve some of the essential design challenges encountered. The first pattern is related to the model-centric approach into complex decision support requirements

> *"The Blackboard architectural pattern is useful for problems for which no deterministic solution strategies are known. In Blackboard several specialized subsystems assemble their knowledge to build a possibility partial or approximate solution."*

In the context of the modeling and simulation environment the Blackboard pattern involves more or less independently deployed software agents connected to the information model. The agents may initiate actions and react to changes in the information model. This solution reduces coupling between the system components and makes no constraints on agent deployment. In a data-centric application of the blackboard pattern standard domain ontologies and information models developed by international organizations should be utilized. The second important pattern is the *layers* pattern. The layered approach can be used to assemble both models and services from simple to complex.

The amount of data in a large industrial modeling project is huge and processing of the semantic graph places high strain on processing power and memory. To reach acceptable performance some techniques are suggested. The first suggestion is to use caching to hide the bottlenecks of distributed processing and the verbose information model. The second suggestion is to use lazy processing such that only necessary data retrieval and operations are performed. The third suggestion is to use event-based

processing so that only the changed information needs to be reprocessed. Such change events can also cover a wide range of different situations such as remote or local changes, undo/redo and temporal evolution. The fourth suggestion involves establishing a locality-based partition for the graph data. Such a partition provides means for efficient representation and distributed transfer of the data.

Many essential services of the modeling and simulation environment require formal specifications from the user. The primary specifications are mappings, validation specifications and parametrizations. The suggestion of this work is to use declarative specifications whenever possible. Such specifications include first-order logic and equations. While algorithmic approaches might be tempting in many situations the concise, formal and robust specifications provided by the declarative means aid the management and quality of the system. Such specifications can also be implemented in a data-driven fashion with known benefits.

The semantic information model is the heart of the modeling and simulation environment but its effective use requires large amounts of program code related to scientific calculation, user interface and decision support services. To make the model more data-centric this work suggests that application code be semantically attached to the information model as much as possible. This involves attaching references to the code into relevant elements in the information model. Such a model provides for discovery of services and also makes possible semantic code generation based on model specifications and user contributions. Such an approach promotes reuse of code and reduces coding errors.

A major contributor behind the success of such engineering tools as APROS and BALAS involve the simple and intuitive approach of graphically configuring models by connecting units. Such an approach can be used in general engineering problems as demonstrated by the connection model in Modelica. Combining such a Lego blocks approach to modeling with template-based reuse provides an intuitive and effective framework for modeling.

The needs of calculation components require separate mechanisms for representation, storage and transfer of real time primitive and graph data. Such mechanisms should be tightly integrated to the semantic information model.

Major trends in computing include increasing mobility and the growth of processing power based on parallel processing. Both of these developments should be addressed in the design of the modeling and simulation environment. The centralized information model provides means for collaborative software agents working according to the blackboard pattern and sending changes to each other. The client interface should be designed with distribution in mind and it should provide for robust synchronization between clients.

The final suggestion is about stressing the importance of documentation and tools for collaboration. Synergies between documentation and collaboration should be exploited and processes for producing formal documentation such as validity conditions should be established.

6.2 Shortcomings

6.2.1 Wide scope

The scope of the work was intentionally left wide. The implications are that the treatment and analysis throughout the work stays at a very high level of abstraction. Obviously, this can cause difficulties for the reader. The offered justification is that the subject of this work has been the architecture of the modeling and simulation environment and for a meaningful treatment no less can be said.

6.2.2 Minimal user and requirements analysis

The discussion about users and requirements was cut to the minimum to keep the size of the work manageable. This is partly due to the wide scope since a wider treatment of specific users and requirements would have introduced unwanted details to the overall discussion. Essentially the high-level users and requirements presented are distilled from a large set of specific use cases from different domains of modeling and simulation.

6.2.3 Variation in the depth of analysis

Some areas of the work received more attention, research, and experimenting while others are more based on purely theoretical consideration and speculation. While the general level of discussion is kept at a high level of abstraction, some areas have more backing than the others. Indicating this in the text is not always easy and could have been done better.

6.2.4 Subjective selection of treated issues

In the technology review and presentation of design challenges, a selection of presented issues was needed. The selection criteria were mostly based on subjective judgements of importance instead of more objective criteria. If the work did not have constraints on size, the treatment would have been wider.

6.3 Open issues and further work

As this work is part of an ongoing research the analysis and discussion presented in this work presents only a selected set of conclusions needed in the design and implementation of the modeling and simulation environment. Research and elaboration on these issues will continue at VTT until the design of the modeling and simulation environment is finished and it is finished when it says it is finished.

7 References

[Ab97] Serge Abiteboul et al. Querying Semi-Structured Data. Proceedings of the 6th International Conference on Database Theory, pages 1–18, Delphi, Greece, 1997.

[ACG+06] Katifori Akrivi, Vassilakis Costas, Lepouras Georgios, Daradimos Ilias, Halatsis Constantin. Visualizing a Temporally – Enhanced Ontology, Proceedings of the Working Conference on Advanced Visual interfaces, Venezia, Italy, 2006.

[Apa] The Apache XML Graphics Project. Batik SVG Toolkit website. Available at: http://xmlgraphics.apache.org/batik/. [Referenced 3.10.2007].

[Apb] Apple Inc. iPhone website. Available at: http://www.apple.com/iphone/. [Referenced 3.10.2007].

[Apc] Apple Inc. Mac OS X Leopard website. Available at: http://www. apple.com/macosx/leopard/. [Referenced 3.10.2007].

[AT06] Eija Alakangas, Pekka Taskinen, editors. MASI Technology Programme 2005-2009. Yearbook 2006. Technology review 191/2006. Tekes, 2006.

[Ba97] Osman Balchi. Verification, Validation and accreditation of simulation models. Proceedings of the 1997 Winter Simulation Conference, Atlanta, USA, 1997.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter Patel-Schneider, editors. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003.

[BD94] Philip A. Bernstein, Umeshwar Dayal. An Overview of Repository Technology. Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994.

[BG02] Bertrand Braunschweig, Rafiqul Gani, editors. Software Architectures and Tools for Computer Aided Process Engineering, 11, volume 11 of Computer-Aided Chemical Engineering. Elsevier, 2002.

[BGH95] Jon Barwise, Dov Gabbay, Chrysafis Hartonas. On the Logic of Information Flow. Bulletin of the IGPL, 3(1):7-49, 1995

[BH74] Alan Baddeley, Graham Hitch, Working memory. In G.H. Bower

(editor), The psychology of learning and motivation: Advances in research and theory, Vol. 8, pages 47-89, Academic Press, 1974.

[BHL01] Tim Berners-Lee, James Hendler, Ora Lassila. The Semantic Web. Scientific American magazine, May 2001.

[BLN86] Carlo Batini, Maurizio Lenzerini, Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, 18(4):323–364, 1986.

[BMM+06] Sebastian C. Brandt, Jan Morbach, Michalis Miatidis, Manfred Theissen, Matthias Jarke, Wolfgang Marquardt. Ontology-Based Information Management in Design Processes. 9th International Symposium on Process Systems Engineering, Garmisch-Partenkirchen, Germany, 2006.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. Pattern-Oriented Software Architecture Volume 1. A System of Patterns. John Wiley & Sons Ltd, 1996.

[Bo] Harold Boley. The Rule Markup Initiative website. Available at: http://www.ruleml.org/. [Referenced 3.10.2007].

[Bo00] Jan Bosch. Design & Use of Software Architectures. Adopting and evolving a product-line approach. Pearson Education Limited, 2000.

[BS00] Piero P. Bonissone, Juhani Sarparanta. Oppivien ja älykkäiden järjestelmien sovellukset 1994-1999. Teknologiaohjelmaraportti 20/2000. Tekes, 2000.

[Bu97] Peter Buneman. Semistructured Data. In PODS ’97: Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pages 117–121, New York, USA, 1997.

[CAP] The CAPE-OPEN Laboratories Network. CAPE-OPEN interface standards website. Available at: http://www.colan.org/index-3.html. [Referenced 3.10.2007].

[CDK05] George Coulouris, Jean Dollimore, Tim Kindberg. Distributed Systems Concepts and Design. Fourth edition. Pearson Education Limited, 2005.

[CFF+98] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, Jamer P. Rice. Open Knowledge Base Connectivity 2.0.3. Specification, Artificial Intelligence Center, SRI International and Knowledge Systems Laboratory, Stanford University, 1998.

[CFG03] Oscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez. Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering, 46(1):41-64, 2003.

[Ch02] Long-Qing Chen. Phase-field models for microstructure evolution. Annual Review of Material Research, 32:113-140, 2002.

[CKS03] Mary Beth Chrissis, Mike Konrad, Sandy Shrum. CMMI®: Guidelines for Process Integration and Product Improvement, second edition. Addison Wesley Professional, 2003

[CP04] Juan Atanasio Carrasco, Matti Paljakka. Simulation Aids in the Automation of Industrial Processes. EUROSIM 2004 Congress. Paris, France, 2004.

[CS] CSC Scientific Computing Ltd. Elmer. Open Source Finite Element Software for Multiphysical Problems. Available at: http://www.csc.fi/elmer/. [Referenced 3.10.2007].

[Cy] Cycorp Inc. The Syntax of CycL website. Available at: http://www.cyc.com/cycdoc/ref/cycl-syntax.html. . [Referenced 3.10.2007].

[De] Defense Modeling and Simulation Office (DMSO). High Level Architecture website. Available at: https://www.dmso.mil/public/public/transition/hla/index_html. [Referenced 3.10.2007].

[DFW98] Judith S. Dahmann, Richard M. Fujimoto, Richard M. Weatherly. The DoD High Level Architecture. An Update. Proceedings of the 30th Winter simulation conference, pages 797 – 804, Washington, USA, 1998.

[DH05] AnHai Doan, Alon Y. Halevy. Semantic integration research in the database community: A brief survey. AI Magazine, 26(1):83–94, 2005.

[Dy] Dynasim AB. Dymola Multi-Engineering Modeling and Simulation website. Available at: http://www.dynasim.com/. [Referenced 3.10.2007].

[Ec] The Eclipse Foundation. Eclipse – an open development platform. Available at: http://www.eclipse.org/. [Referenced 3.10.2007].

[EM97] Hilding Elmqvist, Sven Erik Mattsson. An Introduction to the Physical Modeling Language Modelica. Proceedings of the 9th European Simulation Symposium, Passau, Germany, 1997.

[Er05] Thomas Erl. Service-Oriented Architecture. Concepts, Technology, and Design. Pearson Education Inc., 2005.

[ES95] G. Eriksson, P.J. Spencer. A General Thermodynamic Software Interface, Proceedings of the 2nd Colloquium on Process Simulation, Helsinki University of Technology, 1995.

[Fa98] Eckhard D. Falkenberg et al. FRISCO. A Framework of Information Systems Concepts, IFIP Working Group 8.1.1998. IFIP 1996, 1998.

[Fe] The Federal Ministry of the Interior of the Federal Republic of Germany. The V-Modell XT website. Available at: http://www.v-modell-xt.de/. [Referenced 3.10.2007].

[FI04] FIATECH. Capital Projects Technology Roadmap. Element 9 Tactical Plan, Lifecycle Data Management and Information Integration. 2004.

[Fl03] Luciano Floridi. The Blackwell Guide to the Philosophy of Computing and Information. Blackwell Publishing, 2003.

[Fr] Franz Inc. AllegroGraph 64-bit RDFStore website. Available at: http://www.franz.com/products/allegrograph/. [Referenced 3.10.2007].

[Fr04] Peter Fritzson. Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley-IEEE Press, 2004.

[Ge] Paul Gearon. Mulgara semantic store website. Available at: http://mulgara.org/. [Referenced 3.10.2007].

[Ge91] Michael R. Genesereth. Knowledge Interchange Format. Proceedings of the 2nd International Conference on the Principles of Knowledge Representation and Reasoning, pages 599-600, Cambridge, USA, 1991

[Gr03] Pierre Grenon. BFO in a Nutshell: A Bi-categorial Axiomatization of BFO and Comparison with DOLCE. IFOMIS Report 06/2003, 2003.

[Gr81] Jim Gray. The transaction concept: Virtues and limitations. Proceedings of 7th International Conference of Very Large Data Bases, pages 144–154, Cannes, France, 1981

[Gr93] Thomas R. Gruber. A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2):199-220, 1993.

[Gr95] Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies, 43(5):907–928, 1995.

[GUW02] Hector Garcia-Molina. Jeffrey D. Ullman, Jennifer Widom. Database Systems. The Complete Book. International Edition. Pearson Education International. 2002.

[GW00] Ignacio E. Grossmann, Arthur W. Westerberg. Research Challenges in Process Systems Engineering. AIChE Journal, 46(9):1700-1703, 2000

[GW04] Nicola Guarino, Christopher Welty. An overview of OntoClean. In Steffen Staab and Rudi Studer, editors, Handbook on Ontologies, International Handbooks on Information Systems, pages 151–159. Springer, 2004.

[Ha05] Rolf Haenni. Towards a Unifying Theory of Logical and Probabilistic Reasoning. 4th International Symposium on Imprecise Probabilities and Their Applications, pages 193-202, Pittsburgh, USA, 2005

[HC01a] Katalin Hangos, Ian Cameron. A formal representation of assumptions in process modeling. Computers and Chemical Engineering. 25:237–255, 2001

[HC01b] Katalin Hangos, Ian Cameron. Process Modelling and Model Analysis, Process Systems Engineering Volume 4, Academic Press, 2001.

[He] Helsinki University of Technology Control Engineering Laboratory. AS-74.2400 System Dynamics. Lecture slides, 2006.

[He95] Sandra Heiler. Semantic Interoperability. ACM Computing Surveys. 27(2):271–273, 1995.

[HHB+06] Heinrich Herre, Barbara Heller, Patryk Burek, Robert Hoehndorf, Frank Loebe, Hannes Michalek. General Formal Ontology (GFO). A Foundational Ontology Integrating Objects and Processes. Part I: Basic Principles. Version 1.0. Onto-Med Report No. 8. IMISE. 2006.

[HHJ+01] Heikki Haario, Matti Heiliö, Jari Järvinen, Pekka Neittaanmäki. Matemaattiset menetelmät suomalaisten yritysten t&k – toiminnassa. Teknologiakatsaus 104/2001. Tekes, 2001.

[HJL00] Juha Haataja, Jari Järvinen, Yrjö Leino. Sillanrakennuksesta lääkeainesuunnitteluun. Matemaattinen mallintaminen suomalaisissa yliopistoissa. CSC – Tieteellinen laskenta Oy, 2000.

[HKL95] Rudy Hirschheim, Heinz K. Klein, Kalle Lyytinen. Information Systems Development and Data Modelling: Conceptual and Philosophical

Foundations, Cambridge University Press, 1995.

[HL05a] Duane Hanselman, Bruce Littlefield. Mastering MATLAB 7. Pearson Education Inc., 2005.

[HL05b] Heinrich Herre, Frank Loebe. A Meta-ontological Architecture for Foundational Ontologies. In R. Meersman and Z. Tari (Eds.): Proceedings of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, pages 1398-1415, Agia Napa, Cyprus, 2005.

[HM01] Volker Haarslev, Ralf Möller. RACER System Description. Proceedings of the First International Joint Conference on Automated Reasoning, pages 701-706, Siena, Italy, 2001.

[HP07] HPC in Europe Taskforce. Towards a Sustainable High-Performance Computing Ecosystem through Enabling Petaflop Computing in Europe. 2007.

[HPB+04] Ian Horrocs, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean. SWRL: A Semantic Web Rule Language combining OWL and RuleML, W3C Member Submission, 2004. Available at: http://www.w3.org/Submission/SWRL. [Referenced 3.10.2007].

[HPK+96] Karri Honkoila, Kari Porkholm, Pekka Kokkonen, Timo Kallonen. End-user protégé from Plant Analyzer to Full Scope Training Simulator Development with APROS. 10th European Simulation Multiconference, Budapest, Hungary, 1996.

[HRO06] Alon Halevy, Anand Rajaraman, Joann Ordille. Data Integration: the Teenage Years. Proceedings of the 32nd International Conference on Very Large Data Bases, pages 9–16, Seoul, South Korea, 2006.

[Hy06] Eero Hyvönen. Semantic Web Applications in the Public Sector in Finland – Building the Basis for a National Semantic Web Infrastructure. Norwegian Semantic Days, Stavanger, Norway, 2006.

[IE00] IEEE (Institute of Electrical and Electronics Engineers). IEEE Standard 1471-2000: Recommended Practice for Architectural Description, 2000

[IE90] IEEE (Institute of Electrical and Electronics Engineers). IEEE Standard 610.12-1990: Glossary of Software Engineering Terminology, 1990.

[In00] Integrated Manufacturing Technology Initiative (IMTI). Integrated Manufacturing Technology Roadmapping Project: Roadmap for Modeling & Simulation. IMTI, Inc. 2000.

[In01] International Organization for Standardization (ISO), Geneva, Switzerland. ISO 13584-1:2001: Industrial Automation Systems and Intregration – Parts Library – Part 1: Overview and fundamental principles, 2001

[In02] International Organization for Standardization (ISO), Geneva, Switzerland. ISO/IEC 13250 Topic Maps. Information technology. Document Description and Processing Languages, 2002

[In03] International Organization for Standardization (ISO), Geneva, Switzerland. ISO IS 15926-2:2003: Industrial automation systems and integration — Integration of life-cycle data for oil and gas production facilities — Part 2: Data model, 2003.

[In05a] International Organization for Standardization (ISO), Geneva, Switzerland. ISO DIS 10303-221:2005(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.

[In05b] International Organization for Standardization (ISO), Geneva, Switzerland. ISO DIS 10303-227:2005: Industrial automation systems and integration — Product data representation and exchange Part 227: Application protocol: Plant Spatial Configuration, 2005.

[In05c] International Telecommunication Union (ITU). ITU-T X.509. Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. 2005.

[In07] International Organization for Standardization (ISO), Geneva, Switzerland. ISO/IEC FDIS 24707: Information technology – Common Logic (CL): A framework for a family of logic-based languages. 2007.

[In94] International Organization for Standardization (ISO), Geneva, Switzerland. ISO IS 10303-11:1994: Industrial automation systems and integration — Product data representation and exchange – Part 11. Description methods: The EXPRESS language reference manual. 1994

[In97] International Organization for Standardization (ISO), Geneva, Switzerland. ISO IS 10628:1997(E): Flow diagrams for process plants — General rules, 1997.

[Ina] 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 website. Available at: http://kaon2.semanticweb.org/. [Referenced 3.10.2007].

[Inb] Intergraph Corporation. Integraph Plant Design and Engineering website. Available at: http://www.intergraph.com/pde/default.asp. [Referenced 3.10.2007].

[Inc] International Alliance for Interoperability (IAI). Industry Foundation Classes IFC2x Edition 3 website. Available at: http://www.iai-international.org/Model/R2x3_final/index.htm. [Referenced 3.10.2007].

[Ind] International Organization for Standardization (ISO) TC184/SC4. TC184/SC4, Setting the Standards for Industrial Data website. Available at: http://www.tc184-sc4.org/. [Referenced 3.10.2007].

[Ja06] Mustafa Jarrar. Position Paper: Towards a notion of gloss, and the adoption of linguistic resources in formal ontology engineering. Proceedings of the 15th international conference on World Wide Web, pages 497-503, Edinburgh, Scotland, 2006.

[Je] The Jena Team. Jena – A Semantic Web Framework for Java. Available at: http://jena.sourceforge.net/. [Referenced 3.10.2007].

[JSS04] Harri Jokinen, Tapio Salonen, Juha Sääski, Technical documentation processes in paper mill life cycle. PaperIXI project report D1.1, 2004.

[Ju05] Kaj Juslin. A Companion Model Approach to Modeling and Simulation of Industrial Processes. VTT Publications 574. VTT Technical Research Centre of Finland, 2005.

[Ka02] Tommi Karhela. A Software Architecture for Configuration and Usage of Process Simulation Models: Software Component Technology and XML-based Approach. VTT Publications 479. VTT Technical Research Centre of Finland, 2002.

[Ka07] Toni Kalajainen. An Access Control Model in Semantic Data Structure: Case Process Modelling of a Bleaching Line. Master's thesis, Helsinki University of Technology, 2007.

[Ke04] I.W. Kerlow. The art of 3D computer animation and effects. 3rd edition. John Wiley & Sons Inc., 2004.

[Ke06a] Heikki Kekäläinen (editor). Elektronisen liiketoiminnan logistiikan teknologiatiekartta. Technology Review 189/2006. Tekes, 2006.

[Ke06b] Robert E. Kent. The Information Flow Framework: New Architecture. International Category Theory Conference (CT06), White Point, Canada, 2006.

[KF95] Thomas M. Koulopoulos, Carl Frappaolo. Electronic Document Management Systems : A Portable Consultant, McGraw-Hill, 1995.

[KHR+05] Yannis Kalfoglou, Bo Hu, Dave Reynolds, and Nigel Shadbolt. Semantic integration technologies survey. Technical Report 10842, Southampton, UK, 2005.

[KJB+04] Aditya Kalyanpur, Daniel Jiménez Pastor, Steve Battle, Julian Padget. Automatic Mapping of OWL Ontologies into Java. Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering, pages 98-103, Banff, Alberta, Canada, 2004.

[KKL04] Kalle Kondelin, Tommi Karhela, Pasi Laakso. Service Framework Specification for Process Plant Lifecycle. VTT Research Notes 2277. VTT Technical Research Centre of Finland, 2004.

[KKM+01] Tommi Karhela, Kalle Kondelin, Teemu Mätäsniemi, Matti Paljakka. Prosessi-integraation tieto- ja malligalleria. Suomen Automaatioseuran julkaisusarja nro 24, Automaatiopäivät 01. pages 55-63, Helsinki, 2001.

[KL05] Toni Kalajainen, Marko Luukkainen. Webmon’s User’s manual. 2005

[KLW95] Michael Kifer, Georg Lausen, James Wu. Logical Foundations of Object-Oriented and Frame-based languages. Journal of the ACM, 42(4):741-843, 1995.

[Ko01] Jan L.A Koolen. Design of Simple and Robust Chemical Plants. Wiley-VCH, 2001.

[KP03] Heikki Kantee, Kari Porkholm. Applications of APROS Simulation Software in Development and Validation of New Emergency Operating

Procedures for Loviisa NPP. The 10th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH10) Seoul, Korea, 2003.

[KP04] Heikki Kantee, Kari Porkholm. Capability of APROS in Design and Validation of I&C Systems. International Meeting on Updates in Best Estimate Methods in Nuclear Installation Safety Analysis, pages 206-214, Washington, USA, 2004.

[KP06] Ari Kettunen, Matti Paljakka. Process Simulation in Power Plant Design. SIMS 2006 Proceedings of the 47th Conference on Simulation and Modeling, pages 176-181, Helsinki 2006.

[KR70] Werner Kunz, Horst. W. J. Rittel. Issues as Elements of Information Systems, Working Paper No. 131, July 1970.

[KT01] Kimmo Klemola, Ilkka Turunen. State of Mathematical Modelling and Simulation in the Finnish Process Industry, Universities and Research Centres. Technology Review 107/2001. Tekes, 2001.

[KTB07] Petteri Kangas, Santtu Toivonen, Asta Bäck (editor). Googlen mainokset ja muita sosiaalisen median liiketoimintamalleja. VTT Research Notes 2369. VTT Technical Research Centre of Finland, 2007.

[La06] Kathryn Blackmond Laskey, MEBN: A Language for First-Order Bayesian Knowledge Bases. Working Paper, SEOR Department, George Mason University, 2006.

[Le05] Yuangui Lei. An Instance Mapping Ontology for the Semantic Web. Third International Conference on Knowledge Capture, Banff, Alberta, Canada. 2005.

[Le07] Tuukka Lehtonen. Ontology-Based Diagram Methods in Process Modelling and Simulation. Master's thesis, Helsinki University of Technology, 2007.

[LH99] Reijo Lilja, Jari J. Hämäläinen. Modeling of Thermodynamic Properties of Substances by Neural Networks. International Joint Conference on Neural Networks (IJCNN'99). Washington, DC, USA, 1999.

[LHV+01] Juha Luomala, Juha Heikkinen, Karri Virkajärvi, Jukka Heikkilä, Anne Karjalainen, Anri Kivimäki, Timo Käkölä, Outi Uusitalo, Hannu Lähdevaara. Digitaalinen verkostotalous. Tietotekniikan mahdollisuudet liiketoiminnan kehittämisessä. Technology Review 110/2001. Tekes,

2001.

[LPK+03] Pasi Laakso, Jyrki Peltoniemi, Tommi Karhela, Matti Paljakka. The use of OPC in simulation systems – experiences and future prospects. The 3rd International Symposium on Open Control Systems 2003, Helsinki, 2003.

[LTK+99] Jari Lappalainen, Sami Tuuri, Tommi Karhela, Janne Hankimäki, Pekka Tervola, Soile Peltonen, Toivo Leinonen, Erkki Karppanen, Jarmo Rinne, Kaj Juslin. Direct Connection of Simulator and DCS Enhances Testing and Operator Training. Proceedings of of TAPPI 1999 Engineering / Process & Product Quality Conference, pages 495–502, Anaheim, USA, 1999.

[Lu07] Marko Luukkainen. Use of 3D graphics for configuration and visualization of large-scale process simulation: ontology-based approach. Master's thesis, Helsinki University of Technology, 2007.

[MBG+03] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, and Luc Schneider. Wonderweb Deliverable D17. Preliminary Report Version 2.1, Laboratory for Applied Ontology – ISTC-CNR, Padova (IT), 2003.

[MCW+06] Cynthia Matuszek, John Cabral, Michael Witbrock, John DeOliveira. An Introduction to the Syntax and Content of Cyc. Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering, Stanford, CA, USA, 2006.

[Me04] Daniel Memmi. Toward Tacit Information Retrieval. 7th International Conference on Computer-Assisted Information Retrieval, Avignon, France, 2004.

[MI06] MIMOSA. MIMOSA OSA-EAI Version 3.1 Release. 2006.

[Mia] Microsoft Corporation. COM: Component Object Model Technologies website. Available at: http://www.microsoft.com/com/default.mspx. [Referenced 3.10.2007].

[Mib] Microsoft Corporation. Microsoft Surface website. Available at: http://www.microsoft.com/surface/. [Referenced 3.10.2007].

[Mic] Microsoft Corporation. Windows Vista website. Available at:

www.microsoft.com/windowsvista/. [Referenced 3.10.2007].

[MK03] Robert MacGregor, In-Young Ko. Representing Contextualized Data using Semantic Web Tools. Practical and Scalable Semantic Systems, Proceedings of the First International Workshop on Practical and Scalable Semantic Systems, Sanibel, Florida, USA, 2003.

[MMS+02] Alexander Maedche, Boris Motik, Nuno Silva, Raphael Volz. MAFRA – a Mapping FRAmework for Distributed Ontologies. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web. pages 235-250, Siguenza, Spain, 2002.

[MN03] Wolfgang Marquardt, Manfred Nagl. Workflow and Information Centered Support of Design Processes, 8th International Symposium on Process Systems Engineering. Kunming, China, 2003.

[MS98a] H. H. Mayer, H. Schoenmakers. Application of CAPE in Industry – Status and Outlook. Computers in Chemical Engineering, 22(1):1061-1069, 1998.

[MS98b] Marc Moens, Mark Steedman. Temporal Ontology and Temporal Reference. Computational Linguistics, 14(2), 1988.

[Na04a] National Institute of Standards and Technology (NIST). Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry, 2004.

[Na04b] National Institute of Standards and Technology (NIST). Economic Impact of Inadequate Infrastructure for Supply Chain Integration, 2004.

[Na06] National Science Foundation (NSF). Simulation-Based Engineering Science. Revolutionizing Engineering Science through Simulation. 2006.

[Na07] Anton Naumenko. Semantics-Based Access Control in Business Networks. Dissertation, University of Jyväskylä, 2007.

[Na99] National Institute of Standards and Technology (NIST). STEP The Grand Experience, 1999.

[NK04] Natalya F. Noy, Michel Klein. Ontology Evolution: Not the Same as Schema Evolution, Knowledge and Information Systems, 6(4):428 – 440, 2004.

[No] Noumenon Consulting Limited Services. XMpLant website. Available at: http://www.noumenon.co.uk/. [Referenced 3.10.2007].

[No04] Natalya F. Noy. Semantic Integration: a Survey of Ontology-Based Approaches. SIGMOD Record, 33(4):65–70, 2004.

[NP01] Ian Niles, Adam Pease. Towards a Standard Upper Ontology. Proceedings of the international conference on Formal Ontology in Information Systems Volume 2001, pages 2-9, Ogunquit, Maine, USA, 2001.

[NT95] Ikujiro Nonaka, Hirotaka Takeuchi. The Knowledge Creating Company. How Japanese Companies Create the Dynamics of Innovation. Oxford University Press, Inc., 1995.

[OA06] OASIS (Organization for the Advancement of Structured Information Standards). Reference Model for Service-Oriented Architecture 1.0 Committee Specification 1, 2006.

[Ob03a] The Object Management Group (OMG). Meta Object Facility (MOF) 2.0 Core Specification version 2.0. 2003.

[Ob03b] The Object Management Group (OMG). UML 2.0 OCL Specification. 2003.

[Ob04] The Object Management Group (OMG). ISO/IEC 19500. Common Object Request Broker Architecture (CORBA) Specification, Version 3.1. 2004.

[Ob06] The Object Management Group (OMG). Ontology Definition Metamodel. Sixth Revised Submission to OMG/ RFP ad/2003-03-40. 2006.

[OHP02] Toni Ollikainen, Ari Heino, Kari Porkholm. Generic Training Simulator for a Combined Cycle Gas Turbine Power Plant. Proceedings of the 43rd Conference on Simulation and Modeling of the Finnish Society of Automation, Oulu, Finland, 2002.

[OP] OPC Foundation. The OPC Foundation website. Available at: http://www.opcfoundation.org/. [Referenced 3.10.2007].

[OP06] OPC Foundation. OPC Unified Architecture Specification Part 2: Security Model, 2006.

[Opa] Open CASCADE. Open CASCADE Technology, 3D modeling & numerical simulation website. Available at: http:// www.opencascade.org/. [Referenced 3.10.2007].

[Opb] OpenCFD Ltd. OpenFOAM: The Open Source CDF Toolbox website. Available at: http://www.opencfd.co.uk/openfoam/. [Referenced 3.10.2007].

[Or05] Oracle Corporation. RDF Support in Oracle. Whitepaper, 2005.

[OR93] Jarkko Oikarinen, Darren P. Reed. Internet Relay Chat Protocol. Network Working Group. Request for Comments 1459. 1993.

[OS07] OSGi Alliance. About the OSGi Service Platform. Technical Whitepaper. 2007.

[PAT05] Kari Porkholm, Aino Ahonen, Olli Tiihonen. Utilization of the Simulators in I&C Renewal Project of Loviisa NPP. Presented at Technical Meeting to Develop a Technical Report on Upgrade and Modernization of NPP Training Simulators, KSG, Essen, Germany, 2005.

[Pi04] Guy Pierra. The PLIB Ontology-Based Approach to Data Integration. Proceedings of the IFIP 18th World Computer Congress, Topical Sessions, pages 13-18, Toulouse, France, 2004.

[PKK+01] Herkko Plit, Harri Kontio, Heikki Kantee, Harri Tuomisto. LBLOCA Analyses with APROS to Improve Safety and Performance of Loviisa NPP. Proceedings of the OECD/CSNI Workshop on Advanced Thermal Hydraulic and Neutronic Codes: Current and Future Applications. Paris, France, 2001.

[Pr05] Uta Priss. Formal Concept Analysis in Information Science. Cronin, Blaise (editor), Annual Review of Information Science and Technology, ASIST, Vol. 40. 2005.

[Pra] Process Systems Limited. gPROMS website. Available at: http://www.psenterprise.com/. [Referenced 3.10.2007].

[Prb] Programming Environment Laboratory, Department of Computer and Information Science, Linköping University. The OpenModelica Project website. Available at: http://www.ida.liu.se/~pelab/modelica/OpenModelica.html. [Referenced 3.10.2007].

[PS04] Antti Pärnänen, Pekka Siltanen, Product data standards, PaperIXI project report D6.3, 2004.

[PS06a] PSK Standards Association. PSK 5962. XML Data Transmission. Base

classes and Data Transfer Model. Helsinki, PSK Standards Association. 2nd Edition, 2006.

[PS06b] PSK Standards Association. PSK 5967. XML Data Transmission. Structure of Piping Information. Helsinki, PSK Standards Association. 2006.

[Pu03] Leila Puska. Moniskaalainen mallinnus eri tiedealoilla. CSC – Tieteellinen laskenta Oy, 2003.

[QZ06] Pengfei Qian, Shensheng Zhang. Ontology Mapping Meta-model Based on Set & Relation Theory. Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences Volume 1 (IMSCCS'06), pages 503-509, Hangzhou, China, 2006.

[RN03] Stuart Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall Series in Artificial Intelligence. 2nd edition. Pearson Education Inc. 2003.

[Ro03] Christophe Roche. Ontology: a survey. Proceedings of the 8th Symposium on Automated Systems Based on Human Skill and Knowledge, IFAC, Gotenburg, Sweden, 2003.

[RST+06] Mervi Rajahonka, Senja Svahn, Markku Tinnilä, Mikko Valtakari. Kohti verkostomaista liiketoimintaa. ELO – teknologiaohjelman loppuarviointi. Technology programme report 15/2006. Tekes, 2006.

[RT05] R. M. Ramanathan, Vince Thomas, editors. Platform 2015: Intel® Processor and Platform Evolution for the Next Decade. White paper. Intel Corporation, 2005.

[SGM+02] Clemens Szyperski, Dominik Gruntz, Stephan Murer. Component Software. Beyond Object-Oriented Programming. Second Edition. Pearson Education Ltd. 2002.

[SGT+99] Craig Schlenoff, Michael Grüninger, Florence Tissot, John Valois, Josh Lubell, Jintae Lee. The Process Specification Language (PSL): Overview and Version 1.0 Specification, NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD, USA, 1999.

[Sh00] Robert W. Shirey. Internet Security Glossary. RFC 2828 (Informational), 2000.

[Sh99] Graeme Shanks. Semiotic Approach to Understanding Representation In

Information Systems. Proceedings of the Information Systems Foundations Workshop Ontology, Semiotics and Practice, Sydney, Australia, 1999

[SHT+05] Peter Szulman, Mark Hefke, Adrian Trifu, Martin Soto, Danilo Assmann, Joerg Doerr, Michael Eisenbarth. Using Ontology-Based Reference Models in Digital Production Engineering Integration. Proceedings of the 16th IFAC WORLD CONGRESS, Prague, Czech Republic, 2005.

[SKS02] Abraham Silberschatz, Henry F. Korth, S. Sudarshan. Database System Concepts. 4th edition. McGraw-Hill, 2002.

[SKW+07] Pekka Siltanen, Tommi Karhela, Charles Woodward, Paula Savioja. Augmented Reality for Plant Lifecycle Management. Proceedings of the 13th International Conference on Concurrent Enterprising. (ICE 2007). Sophia Antipolis, France, 2007.

[Sm03] Barry Smith. Ontology. Preprint version of chapter “Ontology”, in L. Floridi (editor), Blackwell Guide to the Philosophy of Computing and Information, pages 155–166, 2003.

[Sö05] Jarmo Söderman (editor). Prosessi-integraatio 2000-2004. Final Report. Tekes, 2005.

[So79] John F. Sowa. Semantics of Conceptual Graphs. Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics, pages 39-44, San Diego, USA, 1979.

[SP06] Pekka Siltanen, Antti Pärnänen. Comparison of data models for plant lifecycle information management. Proceedings of the 13th ISPE International Conference on Concurrent Engineering: Leading the Web in Concurrent Engineering, pages 346–353, Antibes, France, 2006.

[Sp06] Andrew D. Spear. Ontology for the Twenty First Century: An Introduction with Recommendations. 2006

[Sp89] J. Michael Spivey. The Z notation: a reference manual, Prentice-Hall Inc., 1989.

[SS94] Leon Sterling, Ehud Shapiro. The Art of Prolog. Advanced Programming Techniques. Second Edition. The MIT Press. 1994.

[SSR+00] Douglas Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann.

Pattern-Oriented Software Architecture Volume 2. Patterns for Concurrent and Networked Objects. John Wiley & Sons Ltd. 2000.

[St] Stanford Medical Informatics. The Protégé project website. Available at: http://protege.stanford.edu/. [Referenced 3.10.2007].

[St00] John D. Sterman: Business Dynamics - Systems Thinking and Modeling for a Complex World, McGraw-Hill, 2000.

[TH] THTH Association of Decentralized Information Management for Industry. THTH website. Available at: ththry.net. [Referenced 3.10.2007].

[Tu06] Kimmo Tuukkanen. Representing Industrial Data models in OWLWeb Ontology Language. Master's thesis, Helsinki University of Technology, 2006

[TVZ99] Wei-Tek Tsai, Rama Vishnuvajjala, Du Zhang. Verification and Validation of Knowledge-Based Systems. IEEE Transcations on Knowledge and Data Engineering, 11(1), 1999.

[UG96] Mike Uschold, Michael Grüninger. Ontologies: principles, methods and applications. Knowledge Engineering Review, 11(2):93-155, 1996.

[VTa] VTT Technical Research Centre of Finland. The Advanced Process Simulator Environment (APROS) website. Available at: http://apros.vtt.fi/. [Referenced 3.10.2007].

[VTb] VTT Technical Research Centre of Finland. BALAS Process Simulation Software website. Available at: http://balas.vtt.fi/. [Referenced 3.10.2007].

[WAR+97] Arthur Westerberg, Benjamin Allan, Vicente Rico-Ramirez, Mark Thomas, Kenneth Tyner. ASCEND IV. Advanced System for Computations in Engineering Design. A portable mathematical modelling environment. Release 0.8. 1997.

[WKL+00] Antoni Wolski, Jorma Kuha, Tapio Luukkanen, Antti Pesonen. Design of RapidBase – an Active Measurement Database System, International Database Engineering and Applications Symposium (IDEAS 2000). pages 75-82, Yokohama, Japan, 2000.

[Wo04a] World Wide Web Consortium (W3C). OWL Web Ontology Language Overview, W3C Recommendation, 2004. Available at:

http://www.w3.org/TR/owl-features/. [Referenced 3.10.2007].

[Wo04b] World Wide Web Consortium (W3C). XML Schema Part 0: Primer Second Edition. W3C Recommendation. 2004. Available at: http://www.w3.org/TR/xmlschema-0/. [Referenced 3.10.2007].

[Wo06] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C Recommendation. 2006. Available at: http://www.w3.org/TR/xml/. [Referenced 3.10.2007].

[Wo07] World Wide Web Consortium (W3C). SPARQL Query Language for RDF, W3C Candidate Recommendation, 2007. Available at: http://www.w3.org/TR/rdf-sparql-query/. [Referenced 3.10.2007].

[YPK+05] Jean-Peter Ylén, Matti Paljakka, Tommi Karhela, Jouni Savolainen, Kaj Juslin. Experiences on Utilizing Plant Scale Dynamic Simulation in Process Industry. Proceedings of the 19th European Conference on Modelling and Simulation ECMS 2005. Riga, Latvia, 2005.