Handbook of Enterprise Architecture

Chapter 16

Enterprise Modelling

Michael Gruninger
National Institute of Standards and Technology
E-Mail: gruning@nist.gov

Abstract: An enterprise model is a computational representation of the structure, activities, processes, information, people, behaviour, goals and constraints of a business, government or other enterprise. An enterprise model can be both descriptive and definitional and it may cover both 'what is' and 'what should be'. The role of an enterprise model is to achieve model-driven enterprise design, analysis, control[1] and evaluation. However, in order to effectively support the use of enterprise models in practice, it has become evident that more formal approaches to enterprise modelling need to be employed. This chapter considers the application of ontologies to support enterprise modelling. It begins with an overview of the applications of enterprise modelling industrial practice and identifies various formal requirements for enterprise models from these applications. Current research in enterprise modelling ontologies and the languages used to specify these ontologies are reviewed, and a set of challenge problems for future research is proposed

1. Applications of Enterprise Modelling

An enterprise model is a computational representation of the structure, activities, processes, information, people, behaviour, goals and constraints of a business, government or other enterprise. An enterprise model can be both descriptive and definitional and it may \ cover both 'what is' and 'what should be'. The role of an enterprise model is to achieve model-driven enterprise design, analysis, control[2] and evaluation. We will begin by considering these applications of enterprise modelling and the requirements that they impose on any formal approach to the representation and specification of enterprise models.

0.1. Enterprise Integration.

Interoperability among enterprise applications is often hindered because the applications use different terminology and representations of the domain [Ciociou et al. 2001]. These problems arise most acutely for systems that must manage the heterogeneity inherent in various domains and integrate models of different domains into coherent frameworks ().

For example, consider concurrent engineering. A design engineer creates a product specification using a CAD system; this design must be integrated with the company's product data management (PDM) system, which will represent not only the product's features and geometry, but may also include notions such as design intent, additional product requirements and a bill-of-materials (BOM) decomposition. This design must be shared by the engineer with the process design team (which uses the BOM to specify a set of manufacturing processes whose final output may be a product with the desired features). Any version of the

[1] using executable models (e.g. such as the ones CIMOSA aims to produce)

[2] using executable models (e.g. such as the ones CIMOSA aims to produce)

process design may be shared with the process planning team, which specifies the various machines, tools, and materials that will be required by the manufacturing processes. If the process design team identifies any problems with these processes, they must be communicated to the product designer, who may need to modify the design to guarantee manufacturability. The production planning team will need to share the process plan, since it must be included within the production plan, together with the process plans of other products. Schedulers take the production plan and add further constraints on the occurrence of various processes. If either the production planner or scheduler discover a problem (such as unanticipated bottleneck resources), the underlying process plan or production plan may need to be revised upstream (by earlier teams).

Such integration occurs, for example, in business process re-engineering, where enterprise models integrate processes, organizations, goals and customers. Even when the applications involved use the same terminology, they often associate different semantics (meaning) with the terms. This clash over the meaning of the terms prevents the seamless exchange of information among application programs. Typically, point-to-point translation programs are written to enable pair-wise communication between specific applications . However, as the number of applications used in enterprises has increased and the information has become more complex, it has been more difficult for software developers to provide translators between every pair of applications that must cooperate. What is needed is some way of explicitly specifying the terminology of the applications in an unambiguous fashion, so that every application could refer to such common meanings – whether directly or indirectly.

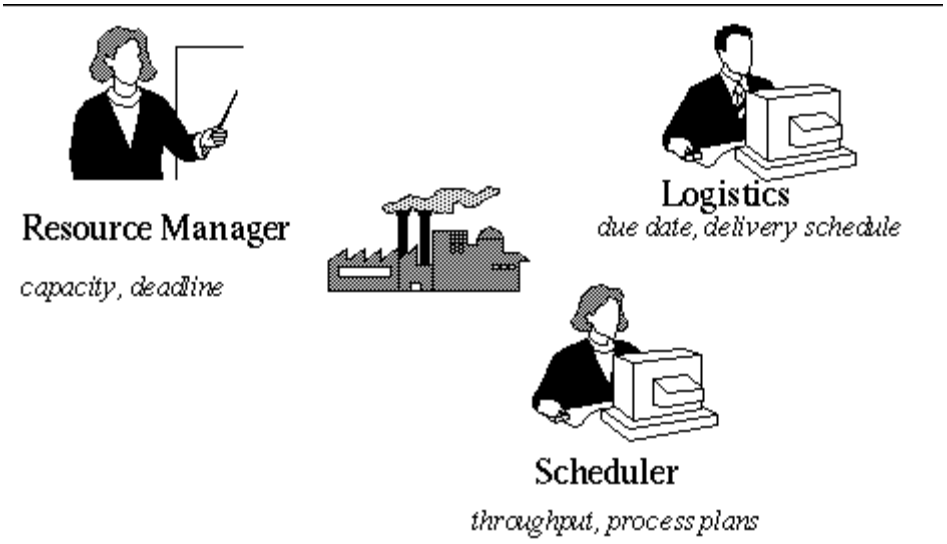

Figure : The challenge of interoperability.

0.1. Reusability.

Knowledge bases that capture the domain knowledge of engineering applications are often tailored to specific tasks and projects. When the application is deployed in a different

domain, it does not perform as expected, often because assumptions are implicitly made about the concepts in the tailored application, and these assumptions are not generic across domains. For example, machine models are often designed for a particular set of properties specific to particular (types of) machines, rather than characterizing generic properties of machines, such as concurrency constraints, setup activities, and operating conditions. Reusability can be achieved through shared understanding of generic concepts that span across multiple projects, tasks and environments[3].

One of the bottlenecks in enterprise engineering is enterprise model acquisition. Models are often constructed for single projects, with little reuse. The dream is to create new models from a repository of existing (partial or particular) enterprise models. Partial models would then be combined into an integrated model of the entire enterprise, thus supporting the iterative refinement/elaboration of the enterprise model. Existing models could be modified to capture the challenges posed by new situations, while templates for various classes of enterprises would allow rapid modelling through instantiation.

0.1. Enterprise Analysis.

An integrated enterprise model provides the language used to specify an explicit definition of an enterprise. The easiest application of such an enterprise model is in checking the consistency of the enterprise model with respect to additional constraints. This may be an *internal* consistency check (in which the enterprise model itself is tested to identify enterprise design problems) or it may be an *external* consistency check (which compares the intended behaviour of the enterprise as expressed in the model with the actual behaviour of the enterprise).

For re-engineering, enterprise engineers need to explore alternative models in the design of enterprises, spanning organisational structure and behaviour. In order to reason about alternative designs for enterprises, they need to reason about different possible sets of constraints for enterprises within the language. Such reasoning can often be expressed by queries whose answers can be deduced from the enterprise model – e.g.: Can a process be performed in a different way, or can the enterprise achieve some goal in a different way? Can the constraints in the enterprise be relaxed, such that we can improve performance or achieve new goals?

Enterprise engineers also need to be able to determine the impact of changes on all parts of the enterprise, which involves *hypothetical* reasoning. For example, if one of the policies is relaxed, how will this affect the quality of products or services provided by the enterprise? If a new kind of machine is purchased, how will this affect the activities that are performed? If the enterprise changes the activities that are performed, how will this change resource consumption?

A related problem is the use of benchmarking in the re-engineering process. In benchmarking, performance is compared between enterprises and then processes and

[3] NB Models with various degrees of reusablity may also be achieved by using *a mix* of generic and particular properties. Such models are considered *partial* - i.e., where only some properties are particularized, the rest being generic.

practices are adopted from enterprises that are the best performers. However, not all practices can be adopted from other enterprises; the key is to realize that one must identify opportunities for improvement by analyzing the successes and failures of similar enterprises. Herein lies the problem -- what is a *similar* enterprise? What is compared among enterprises when using the benchmarking approach? Goals and activities cannot be compared among enterprises unless all constraints and assumptions about the enterprise and its environment are made explicit.

0.1. Model-driven Enterprise Operation

In many Business-to-Business (B2B) E-commerce applications, enterprises face the same integration problems among enterprise systems, although it often involves the semantic integration of intelligent agents that are intended to automate business processes. Emergence of the 'Semantic Web' technologies carries the promise of new advances in the area of inter-enterprise and B2B interoperability, as it has a potential to provide a shared basis for capturing semantics of enterprise-level activities and concepts. However, there is currently a gap between the business processes and organizational structures that are defined within an enterprise model and the enterprise applications that either automate or support these business processes. The behaviour of agents within the enterprise often does not conform to the constraints defined within the enterprise model.

1. Ontologies

To support these applications of enterprise models, there has been an increasing interest in Generic Enterprise Models (GEM) [Fox and Gruninger 1998]. A GEM is an object library that defines the classes of objects that are generic across a type of enterprise, such as manufacturing or banking, and can be employed (through instantiation) in defining a specific enterprise. The benefit of such an enterprise model is that the object library supports reusability and integration through a common conceptualization. The problem is that different representations of the same information may be based on different assumptions about the world, and use differing concepts and terminology -- and conversely, the same terms may be used in different contexts to mean different things. Often, the loosely defined natural-language definitions associated with the terms will be too ambiguous to make the differences evident, or will not provide enough information to resolve the differences.

To address these challenges, various groups within industry, academia, and government have been developing sharable and reusable models known as *ontologies*. The most commonly quoted definition of an ontology is "*a formal, explicit specification of a shared conceptualization*" [Gruber 1993]. In this context, a *conceptualization* refers to an abstract model of how people think about things in the world, usually restricted to a particular subject area. An *explicit specification* means that the concepts and relationships of the abstract model are given explicit names and definitions. The name is a term, while the definition is a specification of what the term means. *Formal* means that the specification

is encoded in a language whose formal properties are well understood – in practice, this almost always means logic-based languages. Formality is an important way to remove ambiguity that is prevalent in natural language; it also opens the door for automated processing of semantics.

It is common to refer to taxonomies, thesauri, data dictionaries, data models and other representations as ontologies, despite their lack of formality. Nevertheless, there is a core essence that is common to virtually all uses of the term 'ontology'. This core is that there are two essential components of any ontology :

- a vocabulary of terms, and
- some specification of meaning for the terms.

What distinguishes the many types of things that people refer to as ontologies is the degree and manner of specifying meaning. This gives rise to a kind of continuum of *kinds of* ontology ([Gruninger and Uschold 2002]). At one extreme, we have very lightweight ontologies that may consist of terms only, with little or no specification of meaning (the degenerate case of an ontology). At the other end of the spectrum we have rigorously formalized logical theories which comprise the ontologies (see). Moving to the right along the continuum, the amount of meaning specified increases (thus reducing ambiguity), the degree of formality increases, and there is increasing support for automated reasoning.

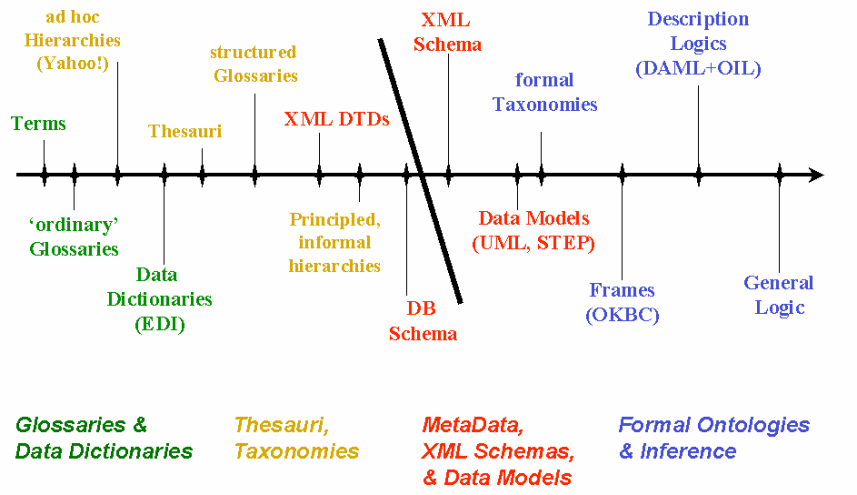

Figure : Kinds of Ontologies – *There are many kinds of things that people call ontologies. Moving to the right there is reduced ambiguity and increased amount of meaning, formality and support for automated reasoning. Not everyone agrees on what is, or what is not an ontology. A line has been arbitrarily drawn, to the right of which there is quite broad agreement that they are ontologies. To the left of the line, the agreement on ontologies is more controversial.*

In the simplest case, the semantics are *implicit* only. Meaning is conveyed based on a shared understanding derived from human consensus. A common example of this case is the typical use of XML tags, such as *price*, *address*, or *delivery date*. Nowhere in the XML document, nor anywhere else, does it say what these tags mean [Cover 98]. However, if there is an implicit shared consensus about what the terms mean, then people can embed this implicit

semantics in screen-scrapers and wrappers. Online travel agents and booksellers routinely do this to find the best deals. From the perspective of mature commercial applications on the Web, this is the current state of the art. The disadvantage of implicit semantics is that they are rife with ambiguity because people often *do* disagree about the meaning of a term.

At the next point on the continuum, the semantics are explicit and are expressed in an *informal* manner, often as text in a specification document. Until the natural language processing problem is solved, only humans can make direct use of informally expressed semantics. Examples of informal semantics that are expressed in text specification documents are: 1) the meaning of tags in HTML, for example, <h2>, which means second level header; 2) the meaning of the `subClassOf` relationship in the RDF Schema language; and 3) the meaning of expressions in programming languages, such as Java.

The main disadvantage of implicit semantics is that there is still much scope for ambiguity. This decreases one's confidence that two different implementations (say of RDF Schema or Java) will be consistent and compatible. Each Java implementation may be different in subtle ways. Users may notice 'features' and start depending on them. This can result in problems if interoperability is required, or if implementations change.

Finally, there is the possibility of explicit, formally specified semantics that are intended for automated inference. Formally specified ontologies use a logical language, such as DAML+OIL [Hendler & McGuinness 2001] and Knowledge Interchange Format (KIF) ([Genesereth & Fikes 1992], [Hayes & Menzel 2001]).. A formal ontology consists of a set of sentences in this underlying logical language. Within mathematical logic these sentences are also known as *axioms,* so that a formal ontology is also said to be *axiomatized*. The idea is that when new terms are encountered, it is possible to automatically interpret something about their meaning and thus how to use them. In order for an enterprise model to support such inference, it must provide a set of rules of deduction together with a set of axioms. For example, given an axiom stating that the *works-for* relation is transitive:

If X works-for Y and Y works-for Z, then X works for Z

and the facts that Alice *works-for* Bob and *Bob works-for* Carol, an inference system would be able to automatically deduce that Alice *works-for* Carol.

1. Desiderata for Enterprise Modelling Ontologies

Based on the scenarios in the first section, what are the requirements that must be satisfied by ontologies for enterprise modelling both in terms of their formality and their content?

0.1. Verification and Semantic Integration

If we are to support scenarios in which enterprise applications *automatically* share and reuse information, we must provide guarantees that the applications completely understand each other within the context of their shared domains. For example, eusability means that enterprise models for new or re-engineered enterprises can be specified by extending (from generic- and/or partial models) or modifying (from partial- and/or particular models)

concepts defined in an existing ontology. Given an enterprise, the modelling task consists of using the ontology to specify a set of expressions that captures the behaviour of the enterprise. How does the engineer determine that he or she is using the correct set of predefined concepts for modelling the enterprise? The engineer must understand the meaning of the ontology that is being reused in the same way that was intended by the designer of the ontology in question. .

The key notion in both semantic integration and reusability is that one must somehow guarantee that the intended meaning of the terms in an ontology is the only meaning for those terms that is consistent with the content of the ontology in conjunction with the semantics of the ontology representation language. Even in cases where the ontology is not explicit, one can examine the behaviour of the application and consider it to be completely dependent on the inferences it makes, which inferences in turn are completely dependent on the axioms in its ontologies and other internal knowledge. In effect, one may consider an application's behaviour to be constrained by the semantics of its ontology, because any inferences made by the application must conform to those semantics (see). If an application does not behave as expected, or retrieve relevant information, or make correct inferences, then it may be concluded that the application in question does not share its semantics with other applications. In this way, one can observe the application's behaviour and infer information about the semantics the application is using.

One must therefore explore the ramifications of explicit formally specified semantics and the requirements that this approach imposes on both the ontology designer as well as the enterprise application designer. There is a need to characterize the relationship between the intended meaning of an application's terminology and the possible meaning of terms in the application's ontology that must hold to support complete semantic integration and reusability.

In most current ontology research, the languages for formal ontologies are closely related to mathematical logic[4], in which the semantics are based on the notion of an *interpretation*. If a sentence is true in the interpretation, we say that the sentence is *satisfied* by the interpretation. If every axiom in the ontology is satisfied by the interpretation, then the interpretation is called a *model* of the ontology. With a formal ontology, the application's knowledge is specified as a theory, so that a sentence is *consistent* with that theory if there exists a model of the theory that satisfies the sentence; a sentence can be deduced if it is

4 An interpretation consists of three parts:
1. a set of elements (known as the *domain* or universe of discourse),
2. a meaning function that associates symbols in the language with individual elements and sets of elements in the domain (intuitively this specifies what the symbols mean), and
3. a truth function that associates truth values with sentences in the language.

For an excellent introduction to logic, see [Barwise et al 2000].

satisfied by all models of the theory. Therefore, the application's behaviour (and hence the semantics of the application's terminology) can be characterized by this implicit set of models, which will hereafter be called the set of *intended models*.

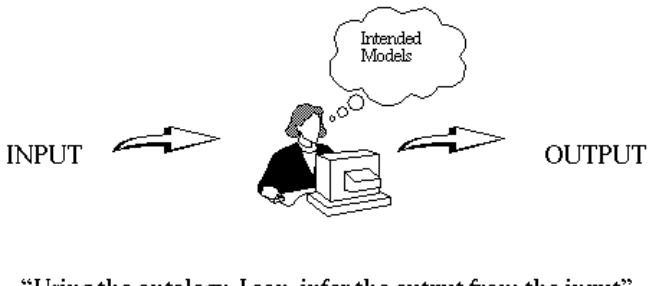

Figure : The Ontological Stance – We can model a software application as if it were an inference system with a formal ontology, and use this ontology to predict the set of sentences that the inference system decides to be satisfiable. This is `the Ontological Stance' [Gruninger & Menzel 2002], and is analogous to the intentional stance ([Dennett 1981]), which is the strategy of interpreting the behaviour of an entity by treating it as if it were a rational agent which performs activities in accordance with some set of intentional constraints, such as goals and obligations.

This characterization may be used to evaluate the adequacy of the application's ontology with respect to these intended models. An ontology is *verified* if and only if the set of models of the axioms of the ontology is equal to the set of intended models for the application's terminology. In a language such as first-order logic, this is equivalent to saying that every sentence that is provable from the axioms of the ontology is also an inference that the application would make, and conversely, every inference that the application makes is provable from the axioms of the ontology.This property of an ontology allows to make the claim that any inferences drawn by the enterprise application using the ontology are faithful to the application's semantics. If an ontology is not verified, then it is possible to find sentences that the application infers based on its intended models which are not provable from the axioms of the ontology. Consequently, automated inference cannot be used to determine that two enterprise applications can in fact be completely semantically integrated, and semantic integration requires human intervention. The discussion in [Gruninger and Uschold 2002] considers in some detail the challenges that must be addressed in achieving semantic integration with unverified ontologies.

We can use an analogy from physics to illustrate the relationship between the ontology, the intended models, and the domain in which the enterprise application is operating. Physicists

use various classes of differential equations to model different phenomena. However, they do not use ordinary linear differential equations to model heat diffusion and they do not use second-order partial differential equations to model the kinematics of springs. If physicists wish to model some phenomena using a class of differential equations, they can use the equations to predict the behaviour of the physical system; if the predictions are falsified by observations, then we have an inappropriate set of equations. Similarly, in this case one may use some class of intended models to predict the inferences that an enterprise application makes; if there is no physical scenario in the domain that corresponds to these inferences then intuitively, the set of intended models is inappropriate

It is important to note that this characterization of semantic integration is independent of the scope of the ontologies. Any given task may require the use of only a portion of the ontologies shared by two enterprise applications. If that relevant portion of the ontologies supports only the intended models, then semantic integration or reusability is not impaired. For example, two applications may have different ontologies for purchasing products, but if the applications are only communicating activities and temporal constraints for scheduling their activities, then any disagreements about purchasing concepts are not relevant. In such a case the applications can still be semantically integrated or reused with respect to the scheduling task even though they have only partially overlapping sets of concepts in their ontologies.

A. Inference.

Enterprise analysis requires inference on enterprise models, including deduction, consistency checking, and abduction (hypothetical reasoning). If the semantics of the ontology is informal, then this inference must be done by an engineer or consultant and cannot be automated. However, if this inference is to be automated, then the ontology must have formal semantics. Even if the analysis is not automated, it cannot be guaranteed that two engineers will agree on the verification or validation of an enterprise model, unless the ontology has a formal semantics.

A. Implementing Enterprise Models.

The dream of model-driven enterprise operations rests on the implementation of an enterprise ontology within the enterprise's information systems. The challenge of evaluating such an implementation leads to the verification of ontology implementation, which determines whether the enterprise application is correct with respect to the ontology. Without a formal semantics for the ontology, the verification of an implemented enterprise model is difficult to evaluate. However, with a formally axiomatized ontology, one can use the Ontological Stance () as the basis for such verification. If an enterprise application is described as an inference system, then any sentence that is satisfiable by the ontology must be decided to be satisfiable by the application.

A. What is an Enterprise?

The discussion to this point has focused on the verification of an ontology, but if one carries the analogy to software engineering, then one must also address the *validation* of the ontology. In particular, does an enterprise modelling ontology have sufficient expressiveness to axiomatize the necessary enterprise modelling constructs? What range of concepts is required for enterprise modelling?

To identify the scope of ontologies required for enterprise modelling, the first need is for a characterization of an enterprise. If an enterprise is defined as a set of sentences whose lexicon consists of terminology whose semantics are specified by the ontologies, then the following classes of sentences would intuitively need to be definable in any validated enterprise ontology:

- Definitions of activities performed within the enterprise;
- Resource constraints for the enterprise;
- Organizational constraints for the enterprise, including constraints among organizational roles, positions, and agents within the enterprise, such as commitments, obligations, and responsibilities;
- Enterprise goals and policies (e.g. "all deliveries must be made within 24 hours of placing the order" and "when an order is made, a copy is sent to the regional office");
- Product constraints for the enterprise, including product design requirements, quality constraints, and product standards;
- Service constraints for the enterprise;
- External constraints on the enterprise defining the external environment of the enterprise, dealing with customers, markets, suppliers, and competitors. External constraints also include the definitions of the activities performed by agents external to the enterprise (e.g. suppliers, subcontractors), but whose effects are required by the activities within the enterprise.

In addition to these classes of sentences, there are several basic relationships that need to be captured:

- Customers have goals and requirements that they assign to the enterprise to achieve;
- There are two primary classes of goals, related to products and services;
- Enterprise goals are generated from the commitment to achieve customer goals and satisfy customer requirements;
- There must be ways of assigning goals of the entire enterprise to agents within the enterprise;
- If the enterprise cannot achieve a particular goal, external agents (such as suppliers or partners) would have to be assigned to achieve that goal;
- The enterprise must be able to perform those activities whose effects achieve the enterprise's goals;
- Resources are required by enterprise activities.

Of course, a truly comprehensive characterization would be provided by an enterprise ontology itself, but this intuitive characterization will later be used to evaluate existing enterprise ontologies.

1. Languages for Enterprise Modelling

There are many languages for expressing ontologies – and their semantic properties vary in important ways. They may be based on different underlying paradigms, they support different levels of expressiveness and also their formal properties may differ. Among the variety of

languages being used today to specify ontologies, the four described in this section are the most widely used.

0.1. UML.

The primary application of UML [Fowler & Scott 1999] for ontology design is in the specification of class diagrams for object-oriented software. However, UML does not have a clearly specified declarative semantics, so that it is not possible to determine whether an ontology is consistent, or to determine the correctness of an implementation of the ontology. Semantic integration in such cases becomes a subjective exercise, validated only by the opinions of the human designers involved in the integration effort. More recently, UML has been supplemented with the Object Constraint Language (OCL) [Warmer & Kleppe 1999] that is closer to offering a semantics similar to a restriction of first-order logic, and there is some research [Cranefield & Purvis 1999] on the suitability of OCL for more rigorous ontology specification.

An additional drawback of UML as an ontology specification language is that it contains several implicit ontological commitments, particularly in respect to activity concepts. This makes it difficult to use UML to integrate different process-related enterprise applications (which may require the use of activity terminology whose semantics may not be equivalent to the intended semantics of the corresponding UML concepts).

On the other hand, UML is closer than more logic-oriented approaches to the programming languages in which enterprise applications are implemented. If there is agreement on the informal semantics of the UML-based ontology, then the verification of the implementation with respect to the ontology may be easier.

0.1. EXPRESS.

EXPRESS ([Schenk & Wilson 1994]) was initially designed to support information modeling, particularly the information required to design, build, and maintain products. Although EXPRESS generalizes earlier approaches such as IDEF1X [Menzel 1997], the major drawback for specifying ontologies for semantic integration is that EXPRESS does not have a clear declarative semantics. This makes it difficult to verify ontologies that use EXPRESS, and also makes it difficult to determine the consistency of semantic mappings between ontologies. There are also no automated inference tools capable of reasoning with EXPRESS beyond checking for data integrity constraints.

The EXPRESS language has been accepted as an international standard [ISO 10303] and is widely used by other ISO standards, particularly the STEP standard for product data exchange.

0.1. DAML+OIL.

The Darpa Agent Markup Language (DAML) [Hendler & McGuinness 2001] is based on description logic [McGuinness & Patel-Schneider 1998; Broekstra *et al*. 2000], which is a specialized language that originated in the KL-ONE system of [Brachman & Schmolze 1981]. Description logic is a variation of first-order logic that arises from restrictions to support reasoning within class hierarchies; they also restrict first-order logic by omitting constructs that lead to undecidability of inference within the language. The Ontology Inference Layer

(OIL) [Fensel *et al*. 2000] is a language that extends previous frame-based languages (such as OKBC [Chaudri *et al.* 1998a]) with a richer set of modeling primitives. These two efforts have been merged to create DAML+OIL [DAML 2001]

What distinguishes DAML+OIL from the other ontology specification languages is that it has been primarily designed for the Semantic Web [Broekstra *et al.* 2000], and is intended to be compatible with emerging web standards such as RDFS [Brickley & Guha 2000] in order to make it easier to use ontologies consistently across the web. It is currently being proposed as a standard within W3C.

Ontologies to support the semantic web are being developed using the Darpa Agent Markup Language (DAML). A library of approximately 160 ontologies are available www.daml.org/ontologies/. The most important of these ontologies is DAML-S [McIlraith *et al*. 2001], which is an upper ontology for services that includes concepts for profiles, processes, and time. In this context, 'services' refer to Web sites that do not merely provide static information but allow one to effect some action or change in the world, such as the sale of a product or the control of a physical device. Thus the DAML-S ontology must support automatic web service discovery, invocation, composition, and interoperation.

0.1. KIF.

The Knowledge Interchange Format (KIF) ([Genesereth & Fikes 1992], [Hayes & Menzel 2001]) and Conceptual Graphs (CG) [Sowa 2000] are languages designed to support the interchange of knowledge among heterogeneous computer systems[5]. KIF includes a core language that has the expressiveness of first-order logic; its syntax and semantics are those of traditional first-order logic. Most recently, this has been extended to include extensions that allow certain formulae of infinite length (known as infinitary logic), sorted formulae for the specification of class hierarchies, and the specification of the meta-theory[6] of KIF within the language itself. Several inference tools are available for reasoning with KIF/CG (such as the SNARK theorem prover [Stickel *et al*. 1994], although these have had limited use outside of the academic community.

1. Ontologies for Enterprise Modelling.

Enterprise modelling ontologies are distinguished by their scope and the central role of integrating multiple ontologies. The ontologies must be able to represent concepts in the domains of activities, time, resources, products, services, organization, goals, and policies. Furthermore, these must be integrated in order to support reasoning that requires the use of

[5] Although defined separately, both KIF and CG have equivalent expressiveness, and are being standardized together within the International Standards Organization

[6] A note to the reader: what is usually referreed to in the Enterprise modelling literature as a 'model' is called in mathematrical logic a 'theory'. Similarly the concept of a meta-model in Enterprise Modelling (and in most engineering disciplines) is mathematically speaking a meta-theory. Considering an enterprise model as a theory allows people and machines to make no inferences from the enterprise model that were not intended (and to be able to make all intended inferences).

multiple ontologies and to support interoperability among tools using different ontologies. For example, the notion of *manufacturability* requires reasoning about the product properties, preconditions and effects of activities and the capabilities of resources.

0.1. Edinburgh Enterprise Ontology.

The Enterprise Project at the University of Edinburgh [Uschold *et al*. 1997] supported an environment for integrating methods and tools for capturing and analyzing key aspects of an enterprise, based on an ontology for enterprise modeling.

The Edinburgh Enterprise Ontology (EEO) has five top-level classes for integrating the various aspects of an enterprise (Activities and Processes, Time, Organization, Strategy and Marketing) for integrating the various aspects of an enterprise.

Activity	Activity Specification, Execute, Executed Activity Specification, T-Begin, T-End, Pre-Conditions, Effect, Doer, Sub-Activity, Authority, Activity Owner, Event, Plan, Sub-Plan, Planning, Process Specification, Capability, Skill, Resource, Resource Allocati
	on, Resource Substitute.
Organization	Person, Machine, Corporation, Partnership, Partner, Legal OrganisationEntity, Organisational Unit, Manage, Delegate, Management Link, Legal Ownership, Non-Legal Ownership, Ownership, Owner, Asset, Stakeholder, Employment Contract, Share, Share Holder.
Strategy	Purpose, Hold Purpose, Intended Purpose, Strategic Purpose, Objective, vision, Mission, Goal, Help Achieve, Strategy, Strategic Planning, Strategic Action, Decision, Assumption, Critical Assumption, Non-Critical Assumption, Influence Factor, Critical Infl
	uence Factor, Non-Critical Influence Factor, Critical Success Factor, Risk.
Marketing	Sale, Potential Sale, For Sale, Sale Offer, Vendor, Actual Customer, Potential Customer, Customer, Reseller, Product, Asking Price, Sale Price, Market, Segmentation Variable, Market Segment, Market Research, Brand Image, Feature, Need, Market Need, Promot
	ion, Competitor.
Time	Time Line, Time Interval, Time Point

Figure : Concepts in the Edinburgh Enterprise Ontology

The Activities and Processes concepts define the activities and resources in the enterprise. The Organization concepts cover the organizational constraints for the enterprise. Goals, policies, and their relationship to the activities performed by the enterprise and its agents are covered by the Strategy concepts. The Marketing concepts cover the constraints that characterize the external environment of the enterprise including the relevant relationships between an enterprise, its customers, suppliers and partners. On the other hand, the Enterprise Ontology lacks a characterization of products and services.

The EEO is semi-formal -- it provides a glossary of terms expressed in a restricted and structured form of natural language supplemented with a few formal axioms using KIF and Ontolingua [Chaudri et al 1998b]. As such, EEO is not a verified ontology.

Lloyd's Register has used the EEO for more effective modeling and re-engineering of business processes for strategic planning. IBM UK intends to exploit the Enterprise Ontology in modeling its own internal organization as well as providing technical input via its BSDM (Business Systems Development Method) business modeling method. The Enterprise Ontology is an ongoing source of inspiration for projects, both academic and commercial that require models of concepts in this domain. To the author's knowledge, the EEO is never imported or translated into a target language in full. Rather, it is perused and picked over for ideas and concepts that may be useful in the new context.

0.1. TOVE

The TOVE (TOronto Virtual Enterprise) project ([Gruninger 1998 and Fox], [Gruninger 1997]) has created an integrated suite of ontologies to support enterprise engineering. Since this suite aims to be a shared terminology for the enterprise that every application can jointly understand and use, the ontologies span knowledge of activity, time, and causality ([Fox *et al*. 1995], [Fadel *et al*. 1994], [Kim & Fox 1994], [Tham *et al*. 1994]).

The TOVE ontologies were developed in cooperation with several companies and have been applied to the design and analysis of enterprise models within supply chain management (SCM), project management, and business process engineering. In particular [Atefi 2000] discusses the application of the TOVE ontologies to the analysis of customer relationship management processes within IBM Canada. In other work, the ontologies were used to model the supply chain of BHP Steel (Australia) and assist in the construction of management scenarios.

shows the suite of TOVE ontologies. The suite is divided into three groups: Core, Derivative, and Enterprise ontologies.

The Core ontologies capture the generic characteristics of enterprises.

The Derivative ontologies are specializations of various classes within some Core ontologies. For example, the concept of “goal” is defined in the (core) Organization Ontology, while different classes of goals (such as purchase orders and deadlines) are defined in the (derivative) Goals Ontology. An ontology may also be derivative of multiple core ontologies. For example, the Scheduling Ontology axiomatizes different classes of plan and

schedule activities, as well as resource and temporal constraints; it is therefore derivative of both the Activity/Time and Resource Ontologies.

There are some problems with the integration of these various ontologies within TOVE, particularly in regards to the Product Ontology. The primary motivation for the TOVE Product Ontology was to support collaborative design. It must therefore be able to represent an evolving and incomplete design for a product, as well as represent the requirements that the product must satisfy. It must also capture the design rationale for various features and parameters of the product. However, TOVE lacks an adequate integration of the Product Ontology with the other ontologies through the following problem: Given a design for a product, how can it be manufactured ? That is, what activities are required to manufacture a product with the properties specified in the design and what resources and organisational constraints are required to support these activities?

The Enterprise ontologies are used to define classes of enterprises. The Enterprise Design Ontology defines the template used to model any enterprise; as such, there is a close relationship to the informal definition of an enterprise. The various Enterprise ontologies define classes of processes, resources, products, services, and organization constraints used to define a particular class of enterprises. For example, the Material Flow ontology axiomatizes the sets of processes and constraints that define supply chain enterprises, the Project ontology captures the constraints of one-of-a-kind manufacturers such as construction and ship-building, and the Business Process ontology addresses service-based enterprises. This approach is intended to support reusability and benchmarking, by identifying those constraints that are shared among different enterprises.

The TOVE ontologies are axiomatized using KIF and implemented using Prolog. Implementations of TOVE ontologies are used to analyze enterprise models in what are referred to as advisors, which are encapsulations of the theories required to reason about alternative enterprise designs [Gruninger and Fox 1994]. Advisors have included activity-based costing, quality, time-based competition, and process integration.

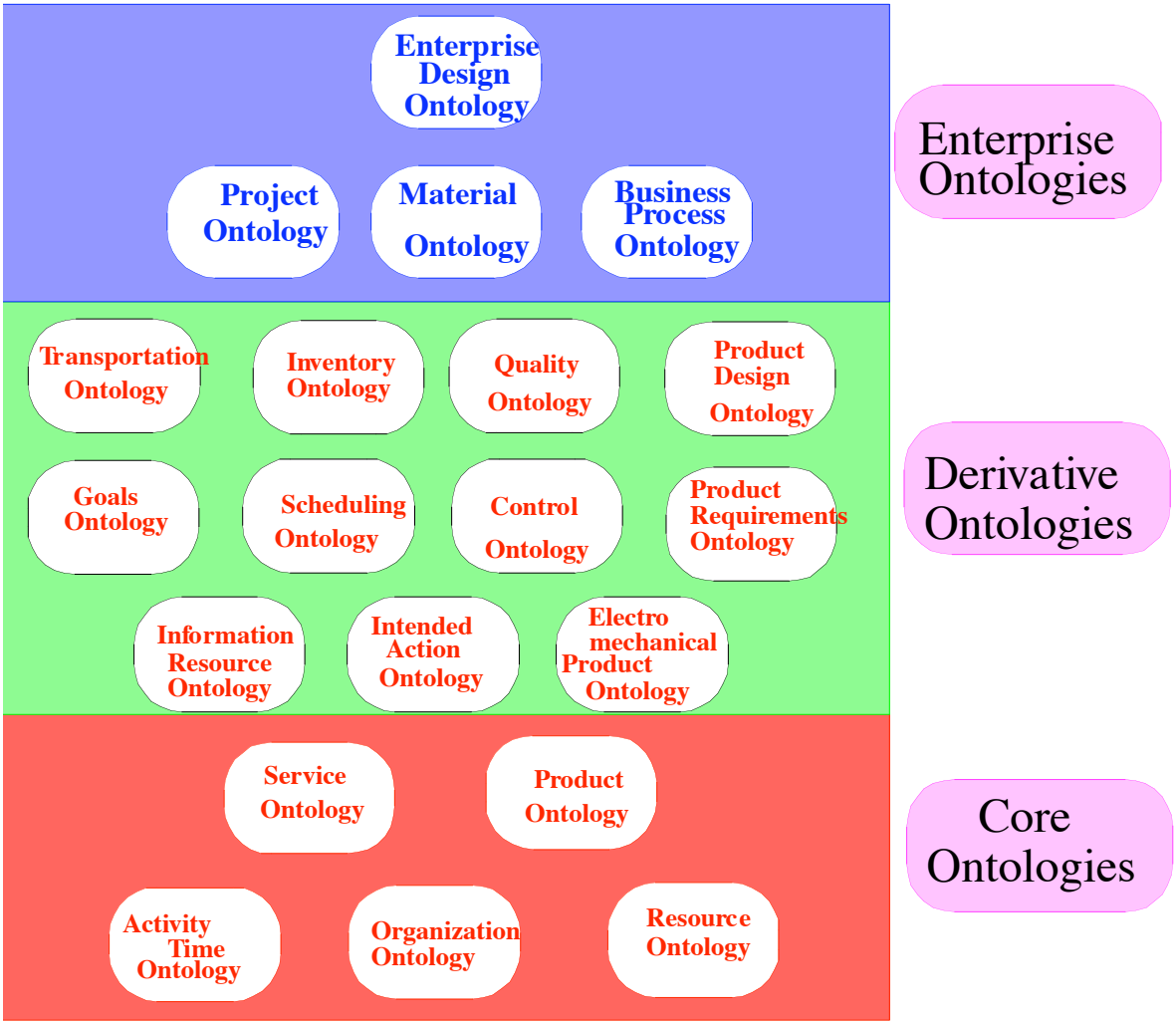

Figure : TOVE Ontologies

0.1. ENV 12204.

ENV12204 [ENV 12204], [Kosanke and Nell 1997]) describes a set of twelve modelling constructs that define the basic language for modelling enterprise operations (). In comparison to the intuitive definition of an enterprise, ENV 12204 provides adequate coverage for enterprise modelling concepts. The primary drawback of ENV 12204 is that it has only an implicit semantics expressed in natural language, and does not have an underlying ontology specification language.

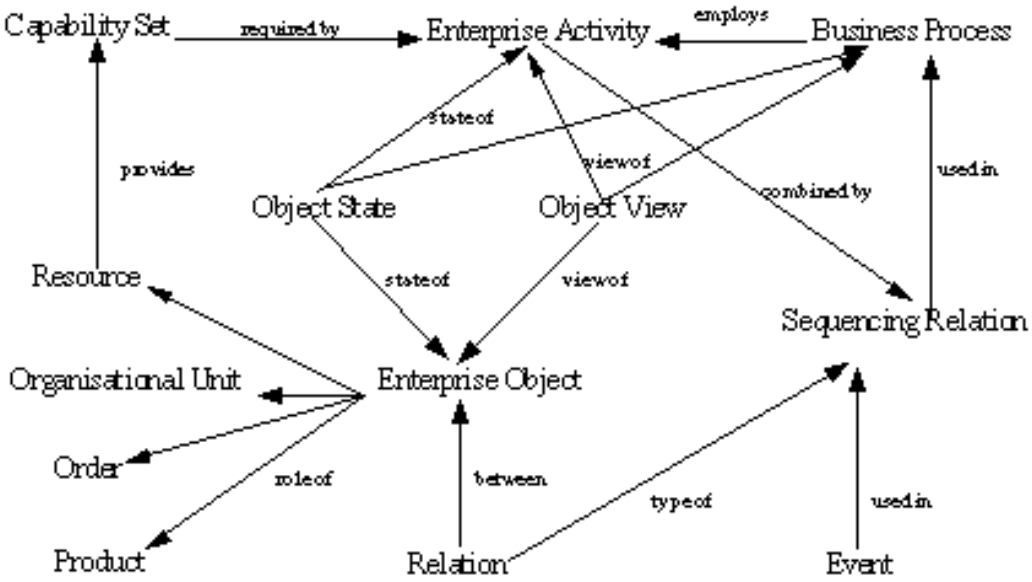

Figure : ENV12204 enterprise modelling constructs.

1. Ontologies for SETS OF Enterprise Modelling Concepts.

Enterprise modelling ontologies explicitly construct an integrated set of smaller modules in order to capture the entire range of enterprise concepts. There are also several efforts underway within academia, industry, and government that are focused on designing ontologies for more restricted sets of enterprise concepts, such as processes, resource, and products.

0.0.1. PSL.

The Process Specification Language (PSL) ([Menzel & Gruninger 2001], [Schlenof *et al*. 1999], [Cutting-Decelle *et al.* 2000]) has been designed to facilitate correct and complete exchange of process information among manufacturing and business software systems. Included in these applications are scheduling, process modeling, process planning, production planning, simulation, project management, workflow, and business process reengineering.

The PSL Ontology is organized into PSL-Core and a partially ordered set of extensions. All axioms are first-order sentences, and are written in KIF . There are two types of extensions within PSL -- core theories and definitional extensions. Core theories introduce and axiomatise new relations and functions that are primitive, whereas all terminology introduced in a definitional extension have conservative definitions using the terminology of the core theories. Thus, definitional extensions add no new expressive power to PSL-Core.

The purpose of PSL-Core is to axiomatize a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes[7]. Consequently, this characterization of basic processes makes few assumptions about their nature beyond what is needed for describing those processes, and the Core is therefore rather weak in terms of logical expressiveness. In particular, PSL-Core is not strong enough to provide definitions of the many auxiliary notions that become necessary to describe all intuitions about manufacturing processes.

To supplement the concepts of PSL-Core, the ontology includes a set of extensions that introduce new terminology. Any PSL extension provides the logical expressiveness to axiomatize intuitions involving concepts that are not explicitly specified in PSL-Core. All extensions within PSL are consistent extensions of PSL-Core, and may be consistent extensions of other PSL extensions. However, not all extensions within PSL need be mutually consistent. Also, the core theories need not be conservative extensions of other core theories.

The definitional extensions are grouped into parts according to the core theories that are required for their definitions. gives an overview of these groups together with example concepts that are defined in the extensions. The definitional extensions in a group contain definitions that are conservative with respect to the specified core theories; for example, all concepts in the Temporal and State Extensions have conservative definitions with respect to both the Complex Activities and Discrete States theories.

Definitional Extensions	**Core Theories**	**Example Concepts**
Activity Extensions	Complex Activities	deterministic / nondeterministic activities concurrent activities partially ordered activities
Temporal and State Extensions	Complex Activities, Discrete States	preconditions effects conditional activities triggered activities
Activity Ordering and Duration Extensions	Subactivity Occurrence Ordering, Iterated Occurrence Ordering, Duration	complex sequences and branching iterated activities duration-based constraints
Resource Role Extensions	Resource Requirements	reusable, consumable, renewable, deteriorating resources

[7] The axioms of PSL-Core were directly incorporated from earlier work with the Process Interchange Format (PIF) ([Lee *et al*. 1996]).

Figure : Definitional extensions of PSL.

PSL is a project within Joint Working Group 8 of Sub-committee 4 (*Industrial data*) and Sub-committee 5 (*Manufacturing integration*) of Technical committee ISO TC 184, (*Industrial automation systems and integration*). Part 1 of the standard has been accepted as a Committee Draft [ISO18629-1]. All theories within the PSL Ontology that are currently being standardized have been verified with respect to the intended semantics of their terminology.

0.0.1. STEP and MANDATE.

STEP [ISO 10303] has been standardized within the International Standards Organization to support interoperability among manufacturing product software applications (such as CAD systems and process planning software) throughout the entire product life-cycle. STEP provides standard data definitions for geometry (wire frame, surfaces and solid models), product identification, product structure, configuration and change management, materials, finite element analysis data, drafting, visual presentation, tolerances, kinematics, electrical properties and process plans. STEP is currently being implemented in the aerospace, automotive, shipbuilding, building design and electronics industries.

MANDATE [ISO 15531] is also being standardized within Joint Working Group 8 of Sub-committee 4 (*Industrial data*) and Sub-committee 5 (*Manufacturing integration*) of Technical committee ISO TC 184, (*Industrial automation systems and integration*). MANDATE is primarily concerned with manufacturing resource data, including an informal ontology of time.

Both STEP and MANDATE are specified in EXPRESS; consequently, they cannot be verified with respect to their intended semantics.

1. Challenge Problems for Enterprise Modelling.

This chapter concludes with five challenge problems to motivate future research within enterprise modelling.

0.1. Ontologies for Enterprise Modelling.

Two ontologies for enterprise modelling have been constructed in the past decade – TOVE and the Edinburgh Enterprise Ontology. Although there is considerable overlap in the set of concepts in each of these ontologies, no effort has been made to merge or align them. Such an alignment could at least form the basis for a formalization of the concepts informally described in ENV 12204. More ambitiously, an alignment of these two enterprise modelling ontologies could be used as the basis for standardization of a wide range of enterprise concepts.

An initial step in this direction is the Unified Enterprise Modelling Language (UEML), a new project whose goal is to provide a common language suited for enterprise modeling

[Kosanke &Nell 1997]. It is intended to provide business users with a standard interface to software for enterprise modeling, analysis, and simulation. It also aims to provide a neutral language for enterprise model exchange.

0.1. Implementing Ontologies .

There is very little work being done on the implementation of ontologies within enterprise applications. In fact, many of the ontologies for enterprise integration are designed post-hoc by extracting the ontology implicit within existing enterprise applications. However, as ontologies are extended to new domains (particularly for organizational constraints and electronic commerce), new applications will be implemented directly from the ontologies. Thus, a methodology for the evaluation of such implementations is needed.

0.1. Ontology Reuse.

Although ontologies came to prominence within artificial intelligence through the DARPA program for Sharable and Reusable Knowledge Bases ([Neches *et al.* 1991], [Gruber 1995]), there is still limited reuse and sharing of ontologies. It is difficult to determine why this is the case ([Uschold *et al*. 1998], [Pinto 1999], [Goldstein & Esterline 1995]). The Ontolingua ontology library at the Stanford University Knowledge Systems Laboratory contains almost 100 ontologies (http://www.ksl.stanford.edu/software/ontolingua), but there are limited links among most of them. Within the context of semantic integration, this becomes the problem of how enterprise applications determine that they have overlapping sets of concepts and how could they possibly share the semantics of their terminology.

The challenge here is to build an ontology for enterprise modelling that integrates existing ontologies for process, product, resource, and organization. Such ontologies often have overlapping concepts, and these may cause problems with reuse. For example, the Standard for the Exchange of Product data (STEP) [ISO10303] was designed for product modeling and the Process Specification Language (PSL) [Schlenoff *et al.* 1999] was designed for process modeling. However, both ontologies contain the concept *process-plan*, which is the sequence of activities that must be performed to manufacture a product according to its design specifications. Unfortunately, this concept is defined very differently in the two ontologies, preventing easy reuse between them.

Enterprise (life-cycle) architecture frameworks, such as GERAM (Chapter 3) may be used to organise ontological theories and define the scope of ontology development for enterprise modelling purposes. The 'generic enterprise models' column of GERAM may be populated with ontological theories. At the moment this column is populated by so-called meta-models. Mathematically speaking this kind of meta-model is a weak ontological theory represented in the form of a meta-schema, defining the concepts and relations between them, but without expressing inference rules or semantic integrity constraints (with the exception of very simple constraints). Thus a meta-schema can be used to design a database to store enterprise models, but can not be used to perform deductions on these models One way to develop an enterprise ontology is to define a meta-schema and then add inference rules and semantic integrity constraints to the concepts defined in such a meta-schema.

0.1. Ontology Extension.

How could generic ontologies be extended to more domain-specific ones? This problem appears in the distinction between Core and Derivative ontologies in TOVE, but there is no coherent methodology for ontology extension. Many ontologies originate as domain ontologies, within different applications' and scientific disciplines' ontologies ([Ashburner 2000], [Cohn 2001], [Dalianis & Persson 1997], [Smith & Becker 1997]). It may be argued that there are few domain concepts in common between physics and logistics and hence little reuse may exist between ontologies for these domains. However, such domain ontologies often use very similar generic concepts (for example, both ontologies may contain an ontology of time). The challenge of reuse and sharing is then equivalent to the task of identifying the generic concepts that are implicit within a domain ontology. In fact, the goal of the Standard Upper Ontology project [Pease 2001] is to a define generic ontology that more domain-specific ontologies can reuse in this way.

0.1. Enterprise in a Test Tube.

There are many issues within enterprise integration that can only be resolved through empirical approaches. There is a need to establish an academic and industrial testbed (which will be refer to as an 'Enterprise In a Test Tube' (ETT)) that consists of multiple enterprise applications and ontologies. Using this environment, participants would carry out experiments to test, compare, and validate various theories about enterprise design and reengineering.

One problem is that enterprise design knowledge is currently descriptive and ad-hoc. It is a collection of heuristics that are not applicable in all circumstances. Therefore, it is desirable to define a theory of enterprise design by discovering its underlying principles. It has to be understood why different approaches and techniques work for certain enterprises and why they fail for other enterprises. There is a need for a distillation of the principles for enterprise design implicit within the heuristics, and the formalisation of these principles as logical theories. Once this is accomplished, various enterprise design theories could be tested, compared and validated.

The use of integrated ontologies allows the flexible configuration of enterprise models and operating scenarios for problems. For example, operating strategies within an enteprise (such as quality problem response, production strategies and inventory management policies) would be explicitly represented in the enterprise model, supporting a 'plug-and-play' approach to the incorporation and change of constraints in a problem specification. Hypotheses for enterprise design heuristics are expressed as queries that can be deduced from the axioms of the ontologies and theories.

The ETT should also support new ways of building enterprise models, particularly in the acquisition and validation of an enterprise model. It should support the capability of

reconciling different enterprise designs that may arise during the acquisition process. Model acquisition must therefore be able to handle incomplete and inconsistent information, as well as being able to modify or augment a model when things don't work. It should use partial enterprise models, combine these partial models into an integrated model of the entire enterprise, and support the iterative refinement / elaboration and definition of the enterprise model, 'filling in' pieces of incomplete models.

To be effective, the ETT must also support reusability by providing a repository for various enterprise models, including previous problems and their solutions[8]. ETT must provide the capability of dynamically constructing and modifying models, so that new models can be created from existing models by reconfiguring them to adapt to a given problem.

1. References

[Ashburner 2000] Ashburner, M. Gene ontology: Tool for the unification of biology. *Nature Genetics* 25:25-29.

[Atefi 97] Atefi, K., "Formalization of Business Process Re-engineering Heuristics", M.A.Sc. Thesis, Mechanical and Industrial Engineering Dept., University of Toronto, 1997.

[Barwise *et al*. 2000] Barwise, J., Etchemendy, J., Allwein, G., Barker-Plummer, D. *Language, Proof, and Logic*. Seven Bridges Press.

[Brickley & Guha 2000] Brickley, D. and Guha, V.R. *Resource Description Framework Schema Specification 1.0*. W3C Candidate Recommendation. (http://www.w3.org/TR/rdf-schema)

[Broekstra *et al*. 2000] Broekstra, J., Klein, M., Fensel, D., and Horrocks, I. Adding formal semantics to the Web: Building on top of RDF Schema. *Proceedings of the ECDL 2000 Workshop on the Semantic Web.*

[Chaudri et al 1998a] Chaudri, V.K., Farquhar, A.,Fikes, R., Karp, P.D., and Rice, J.P. OKBC: A programmatic foundation for knowledge base interoperability*, Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, Wisoconsin.

[Chaudri et al 1998b] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, & J. P. Rice. Open Knowledge Base Connectivity 2.0. Knowledge Systems Laboratory, 1998.[Ciociou et. al. 2001] Ciocoiu, M., Gruninger M., and Nau, D. Ontologies for integrating engineering applications*, Journal of Computing and Information Science in Engineering*, 1:45-60.

[Cohn 2001] Cohn, A. Formalizing biospatial knowledge, *Formal Ontology in Information Systems* 2001, Ogunquit, Maine.

[Cover 1998] Cover, R. XML and Semantic Transparency, The XML Cover Pages http://www.oasis-open.org/cover/xmlAndSemantics.html

[Cranefield & Purvis 1999] Cranefield, S. and Purvis, M. UML as an ontology modeling language. In *Proceedings of the Workshop on Intelligent Information Integration*, Sixteenth International Joint Conference on Artificial Intelligence.

[Cutting-Decelle *et al*. 2000] Cutting-Decelle, A.F., Anumba, C.J., Baldwin, A.N., Gruninger, M. Towards a unified specification of construction process information: The PSL approach, in *Product and Process Modelling in Building and Construction*, Steiger-Garcao and Scherer (eds), 199-207.

[Dalianis & Persson 1997] Daliannis, H. and Persson, F. Reuse of an ontology in an electrical distribution network domain, *Ontological Engineering*, AAAI-97 Spring Symposium Series, Stanford.

[Dennett 1989] Dennett, D. *The Intentional Stance*. MIT Press.

[ENV 12204] ENV 12204 *Advanced Manufacturing Technology – Systems Architecture – Framework for*

[8] e.g. in the form of enterprise modelling *patterns*.

Enterprise Modelling. CEN TC 310/WG1, 1995.

[Fadel *et al*. 94] Fadel, F., Fox, M.S., and Gruninger, M. A resource ontology for enterprise modelling. Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises, (West Virginia University 1994), pp. 117-128.

[Fensel *et al.* 2001] Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. IEEE Intelligent Systems, 16(2):38--44. http://citeseer.nj.nec.com/fensel01oil.html

[Fowler & Scott 1999] Fowler, M. and Scott, K. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley.

[Fox *et al.* 1995] Fox, M.S., Barbuceanu, M., Gruninger, M., An Organisation Ontology for Enterprise Modelling: Preliminary Concepts for Linking Structure and Behaviour, *Computers in Industry*, Vol. 29, pp. 123-134.

[Genesereth & Fikes 1992] Genesereth, M.R. and Fikes, R. *Knowledge Interchange Format 3.0*.Technical Report KSL-92-01, Knowledge Systems Laboratory, Stanford University.

[Goldstein & Esterline 95] Goldstein, D., and Esterline, A., "Methods for Building Sharable Ontologies", Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing, Menlo Park CA, USA: AAAI Press, 1995.

[Gruber 1993] Gruber, T.R. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5:199-220.

[Gruninger and Menzel 2002] Gruninger, M. and Menzel, C. Process Specification Language: Principles and Applications, to appear in *AI Magazine*, 2002.

[Gruninger and Uschold 2002] Gruninger, M. and Uschold, M. Ontologies and semantic integration, to appear in *Software Agents for the Warfighter*. Information Technology Assessment Consortium.

[Gruninger and Fox 1998] Gruninger, M., and Fox, M.S., Enterprise Modelling, *AI Magazine*, 19:109-121 (Fall 1998), AAAI Press.

[Gruninger 1996] Gruninger, M., Designing Generic Ontologies, *Workshop on Ontological Engineerin*g, European Conference on Artificial Intelligence 1996, Budapest.

[Gruninger 1997] Gruninger, M., Ontologies for Enterprise Engineering*, Enterprise Engineering and Integration: Building International Consensus*, Springer-Verlag.

[Gruninger & Fox 94] Gruninger, M., and Fox, M.S., "The Role of Competency Questions in Enterprise Engineering*", Benchmarking - Theory and Practic*e. Chapman Press.

[Guarino *et al.* 94] Guarino, N., Carrara, M., and Giaretta, P. 1994. An Ontology of Meta-Level Categories. In E. Sandewall and P. Torasso (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94*). Morgan Kaufmann, San Mateo, CA: 270-280.

[Guarino & Welty 2000] Guarino, N. and Welty, C. Identity, unity, and individuality: Towards a formal toolkit for ontological analysis. *Proceedings of the Fourteenth European Conference on Artificial Intelligence*, Berlin.

[Hayes & Menzel 2001] Hayes, P. and Menzel, C. A semantics for the Knowledge Interchange Format, *Workshop on the IEEE Standard Upper Ontology*, IJCAI 2001, Seattle.

[Hendler & McGuinness 2001] Hendler, J. and McGuinness, D. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, January 2001.

[ISO 10303] *Industrial Systems and Automation – Product Data*, 1994.

[ISO 15531] *Industrial Systems and Automation – Industrial Manufacturing Management Data,* 1999.

[ISO/TC184/SC5/WG1] Annex A: GERAM. In ISO/DIS 15704: Industrial automation systems - Requirements for enterprise-reference architectures and methodologies, 1999

[ISO 18629] *Industrial Systems and Automation – Process Specification Language,* 2000.

[Kim & Fox 95] Kim, H. and Fox, M.S. An Ontology of Quality for Enterprise Modelling, Fourth Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises, (West Virginia University 1995).

[Kosanke & Nell 1997] Kosanke, K. and Nell, J. (eds*.) Enterprise Engineering and Integration: Building International Consensus*. Springer-Verlag.

[Lee *et al*. 1998] Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost, G. (1998) The PIF Process Interchange Format and Framework, *Knowledge Engineering Review*, 2:1-30.

[McGuinness & and Patel-Schneider 1998] McGuinness, D.L. and Patel-Schneider, P.F. Usability issues in description logic systems, *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, Wisconsin.

[McIlraith*et al*. 2001] McIlraith, S., Son, T.C., and Zeng, H. Semantic web service, *IEEE Intelligent Systems* 16:46-53.

[Menzel 1997] Menzel, M. Modeling method ontologies: A formal foundation for enterprise model integration, *Workshop on Ontological Engineering*, AAAI-97 Spring Symposium, Stanford.

[Menzel & Gruninger 2001] Menzel, C. and Gruninger M. A formal foundation for process modeling, *Formal Ontology in Information Systems 2001*, Ogunquit Maine.

[Neches *et al*. 1991] Neches, R., Fikes, R., Finin, T., Gruber, T., Patil, R., Senator, T., and Swartout, W. Enabling technology for knowledge sharing, AI Magazine, Vol. 12, No. 3, 36-56.

[Pease 2001] Pease, A. (ed*) Proceedings of the IEEE Standard Upper Ontology Workshop*, IJCAI-2001, Seattle.

[Pinto 1999] Pinto, H.S. Towards ontology reuse, *Workshop on Ontology Management*, AAAI-99, Orlando.

[Schenk & Wilson 1994] Schenk, D.A. and Wilson, P.R. *Information Modeling the EXPRESS Way*. Oxford University Press, 1994.

[Schlenoff *et al*. 1999] Schlenoff, C., Gruninger, M., Ciocoiu, M. The Essence of the Process Specification Language, *Transactions of the Society for Computer Simulation vol.16 no.4 (December 1999) pages 204-216.*

[Smith *et al.* 1998] I. Smith, P. Cohen, J. Bradshaw, M. Greaves and H. Holmback. "Designing Conversation Policies using Joint Intention Theory". Proceedings of the Third International Conference on Multi Agent Systems (ICMAS-98), 3 - 7 July, 1998, Paris, France, IEEE Press, pp. 269-276.

[Smith & Becker 1997] Smith, S. and Becker, M. An ontology for constructing scheduling systems, *Ontological Engineering*, AAAI-97 Spring Symposium Series, Stanford.

[Sowa 2000] Sowa, J.F*., Knowledge Representation: Logical, Philosophical and Computational Foundations.* Brooks/Cole.

[Stickel *et al*. 1994] Stickel, M., Waldinger, R., Lowry, M., Pressburger, T., and Underwood, I. Deductive composition of astronomical software from subroutine libraries*, Proceedings of the Twelfth International Conference on Automated Deduction*, Nancy, France, 341-355.

[Tham *et al*. 94] Tham, D., Fox, M.S., and Gruninger, M., "A Cost Ontology for Enterprise Modelling", Third Workshop on Enabling Technologies-Infrastructures for Collaborative Enterprises, (West Virginia University 1994).

[Uschold & Gruninger 1996] Uschold, M. and Gruninger, M. Ontologies: Principles, Methods, and Applications, *Knowledge Engineering Review*, 1:96-137.

[Uschold *et al.* 97] Uschold, M., King, M., Moralee, S., Zorgios, Y., "The Enterprise Ontology", Knowledge Engineering Review,

[Uschold *et al.* 1998] Mike Uschold, Mike Healy, Keith Williamson, Peter Clark, Steven Woods. Ontology Reuse and Application. In Proceedings of the International Conference on Formal Ontology and Information Systems - FOIS'98 (Frontiers in AI and Applications v46), pages 179-192, Ed: N.

Guarino, Amsterdam:IOS Press, 1998.

[Warmer & Kleppe 1999] Warmer, J. and Kleppe*, A. The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley.