ONTOLOGIES: Principles, Methods and Applications

Mike Uschold & Michael Gruninger

AIAI-TR-191

February 1996

To appear in Knowledge Engineering Review
Volume 11 Number 2, June 1996

Mike Uschold
Artificial Intelligence Applications Institute
(AIAI); The University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland

Tel: +44 (0)131 650-2732
Fax: +44 (0)131 650-6513
Email: m.uschold@ed.ac.uk

Michael Gruninger
Department of Industrial Engineering
University of Toronto
Toronto, Ontario
M5S 1A4
Canada

+1 416-978-6347
+1 416-971-2479
mudcat@ie.utoronto.ca

Abstract

This paper is intended to serve as a comprehensive introduction to the emerging field concerned with the design and use of ontologies. We observe that disparate backgrounds, languages, tools, and techniques are a major barrier to effective communication among people, organisations, and/or software systems. We show how the development and implementation of an explicit account of a shared understanding (i.e. an 'ontology') in a given subject area, can improve such communication, which in turn, can give rise to greater reuse and sharing, inter-operability, and more reliable software.

After motivating their need, we clarify just what ontologies are and what purposes they serve. We outline a methodology for developing and evaluating ontologies, first discussing informal techniques, concerning such issues as scoping, handling ambiguity, reaching agreement and producing definitions. We then consider the benefits of and describe, a more formal approach. We re-visit the scoping phase, and discuss the role of formal languages and techniques in the specification, implementation and evaluation of ontologies. Finally, we review the state of the art and practice in this emerging field, considering various case studies, software tools for ontology development, key research issues and future prospects.

Contents

1 Introduction

The overall goal of this paper is to give readers a practical understanding of the emerging field concerned with the nature and use of ontologies. This is an introduction to the field, rather than a comprehensive review of it. In particular, our aim is that readers will:

- Understand what is meant by the term 'ontology';
- Know the range of purposes that an ontology may serve and thus to be able to identify when to use an ontology for their own problems;
- Be familiar with a number of current applications of ontologies;
- Be familiar with the current state of the technology, in particular:
  - the main steps in building an ontology,
  - analytical techniques and software tools to support the process of building and using ontologies,
  - current limitations of such techniques;
- Have a better understanding of the potential for commercial exploitation of ontologies in the short, medium, and long term.

Outline

We begin by motivating the need for ontologies, in particular by describing a number of important problems that obstruct communication between or among people, organisations, and/or software systems. We illustrate that the development and implementation of an explicit account of a *shared understanding* (*i.e.*. an 'ontology') in a given subject area, can help solve these problems.

We briefly consider the nature of an ontology, and the range of uses that they have. We outline a skeletal methodology for the process of developing and using ontologies. In subsequent sections, we generalise and summarise our experiences in developing two significant ontologies in the domain of enterprise modelling. In doing so, we elaborate on some of the specific steps outlined in the skeletal methodology.

We consider first, some important *informal* techniques for ontology development. We proceed by considering how to identify what the important concepts and ideas are in a domain of interest, thus limiting the scope of the ontology. Next, we give a procedure and suggest guidelines for producing the actual definitions, and how to reach agreement.

Next, we consider advantages of and describe a more formal approach to the development of ontologies. We re-visit the scoping phase, and discuss the role of formal languages and techniques in the specification, implementation and evaluation of ontologies.

In § 7 we describe a variety of practical applications of ontologies. We look at various case studies, reviewing what is available by way of software tools and techniques for ontology development and implementation.

We conclude by reviewing some important research issues, summarising the state of the art and discussing future directions.

2 Why Ontologies, and What are They?

2.1 What are the Problems?

People, organisations, and software systems must communicate between and among themselves. However, due to different needs and background contexts, there can be widely varying viewpoints and assumptions regarding what is essentially the same subject matter. Each uses different jargon; each may have differing, overlapping and/or mis-matched concepts, structures and methods. The consequent lack of a *shared understanding* leads to

- *poor communication* within and between these people and their organisations.

In the context of building an IT system, this lack of a shared understanding leads to

- difficulties in identifying requirements and thus in the defining of a *specification* of the system.

Disparate modelling methods, paradigms, languages and software tools severely limit:

- *inter-operability*;
- the potential for *re-use and sharing.*

In turn this leads to

- much *wasted effort re-inventing the wheel.*

2.2 How can we Solve them?

The way to address these problems, is to reduce or eliminate conceptual and terminological confusion and come to a *shared understanding.* Such an understanding can function as a *unifying framework* for the different viewpoints and serve as the basis for:

Communication between *people* with different needs and viewpoints arising from their differing contexts;

Inter-Operability among *systems* achieved by translating between different modelling methods, paradigms, languages and software tools;

System Engineering Benefits: In particular,

Re-Usability: the shared understanding is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. This formal representation may be (or become so by automatic translation) a re-usable and/or shared component in a software system.

Reliability: A formal representation also makes possible the automation of consistency checking resulting in more reliable software.

Specification: the shared understanding can assist the process of identifying requirements and defining a specification for an IT system. This is especially true when the requirements involve different groups using different terminology in the same domain, or multiple domains.

2.3 Examples

2.3.1 Unifying Research Fields

Here we describe an interesting example whereby a shared understanding can be used to enhance communication between people.

SITUATION/PROBLEM: Researchers in the different but related fields of AI Planning, Decision Theory, and Distributed Systems Theory (from work in theoretical computer science) cannot readily make use of each other's results. This is because they have a different perspective on, and use different terms to describe, the same underlying ideas.

SOLUTION: Develop a unifying conceptual framework which enables research results in one field to be applied to the other fields.

HOW? Identify the common ideas in each of these fields and the terms used that correspond to them. Perform a careful technical analysis of what exactly these concepts are; identify any exact matches, and note other important relationships between them. This unifying conceptual framework is intended to function as an lingua-franca enabling translation between the different perspectives in the three subfields.

When a new research result is published, it may be possible to interpret the results in one field using terms from another. For example, a new algorithm to solve a problem in Distributed Systems Theory might be used as a new search algorithm in an AI planning system.

SO WHAT? By allowing the conceptual frameworks and underlying assumptions in each of the three fields to be compared and built upon, there is great potential for increasing the rate of progress in all three fields by avoiding re-discovering equivalent results.

This example was obtained from an invited talk by Michael Georgeff titled "Agents and Their Plans" given at IJCAI-95 in Montreal. Although much more remains to be done before the unification is fully accomplished, preliminary results are encouraging[1].

2.3.2 Semi-Conductor Fabrication

SITUATION/PROBLEM: Software bought in from the outside includes a WIP tracking system[2] and production line simulation package. The simulation package requires as input, a very large description of a model of the product flow in the factory, which incorporates various details of the WIP tracking mechanism. When new versions of the simulation package are released, or if a new supplier is chosen, the model must be converted to a new format. This conversion is both *time-consuming* and *error-prone*.

SOLUTION: Automate the process of converting the model when new external software is introduced. This both saves time and ensures model fidelity.

HOW? There are 3 intersecting domains of interest: WIP tracking, product flow simulation, and the semi-conductor fabrication process. Whichever particular WIP tracking system or simulation package is used, the underlying concepts are the same. The approach was to develop a *unifying framework* which identified, defined, and named all the important concepts in this intersection. The models are expressed in terms of this framework and stored in an ORACLE relational database.

An automatic translator converts the models from the ORACLE DB into the format required by the simulation software. If the simulation software changes, then the translator must be changed, manually. However, the changes are usually relatively minor, especially compared to the original task of manually converting the model.

The ORACLE DB is itself populated by a translator that extracts information from the WIP tracking system. Just as the DB entries are automatically translated into model components required as input to the simulator, WIP system tracking entries are also automatically translated into DB entries.

SO WHAT? This insulates the semiconductor fabrication company from changes in software provided externally, thus saving time and ensuring model fidelity. Development of the unifying framework assisted in the *specification* of the software for representing the appropriate concepts in the model and translating it into the appropriate format. The framework was the basis for implementing the sound software engineering practice of modularity, which in turn facilitated *inter-operability* of independently produced software.

[1] There is no paper that directly corresponds to the talk, however, this material is covered by a number of papers that may be found at 'http://www.aaii.oz.au/'.

[2] WIP is for Work In Progress; such trackers monitor location and status of products as they are being assembled. They are updated every time value is added during production.

2.3.3 Spacecraft Mission Operations

SITUATION/PROBLEM: Various knowledge-based systems were developed independently to assist in different aspects of spacecraft operations (*e.g.* in planning, anomaly detection, diagnosis). Each uses its own approach to structuring and representing the relevant concepts in a large knowledge base.

It is desirable to integrate these system, so that each can make use of the knowledge of the others. For example, mission planning results are inputs to the mission execution system; anomalies are input to the diagnosis system.

However, the differing approaches are a barrier. Furthermore, it is undesirable (and perhaps impractical) to impose a uniform approach on any of the individual systems.

SOLUTION: Use a federated agent-based approach to knowledge sharing. The overall system is called ATOS: Advanced Technology Operations System. [20, 21]

HOW? Identify the important underlying concepts, define them, assign terms to them and note their important inter-relationships. This is a *unifying framework*, which is the basis for achieving inter-system integration. It is the heart of the ATOS infra-structure.

Each separate module remains unchanged; agents act as brokers between the individual systems. The unifying framework acts as a standard to which each agent must comply *e.g.* such terms as 'resource', or 'schedule' have a carefully defined agreed meaning. Such compliance serves to guarantee for other agents that terms are being used in a particular manner, which in turn enables knowledge in one sub-system to be accessible to other sub-systems.

The framework acts as a lingua-franca. Each agent *translates* the encoding used by the independent module into the representation used in the unifying framework, and vice versa.

In ATOS, the framework is represented in a formal language, this facilitates (semi-)automatic development of the agent translators and ensuring compliance with the standard.

SO WHAT? This approach facilitates *knowledge sharing* and *inter-operability* between independently developed sub-systems.

2.4 What is an ontology?

'*Ontology*' is the term used to refer to the shared understanding of some domain of interest which may be used as a unifying framework to solve the above problems in the above-described manner.

An ontology necessarily entails or embodies some sort of world view with respect to a given domain. The world view is often conceived as a set of concepts (*e.g.* entities, attributes, processes), their definitions and their inter-relationships; this is is referred to as a *conceptualisation*.

Such a conceptualisation may be *implicit*, *e.g.* existing only in someone's head, or embodied in a piece of software. For example, an accounting package presumes some world view encompassing such concepts as invoice, and a department in an organisation. The word 'ontology' is sometimes used to refer to this implicit conceptualisation. However, the more standard usage and that which we will adopt is that the ontology is an *explicit* account or representation of [some part of] a conceptualisation.

What does an ontology look like? An [explicit] ontology may take a variety of forms, but necessarily it will include a vocabulary of terms and some specification of their meaning (*i.e.* definitions).

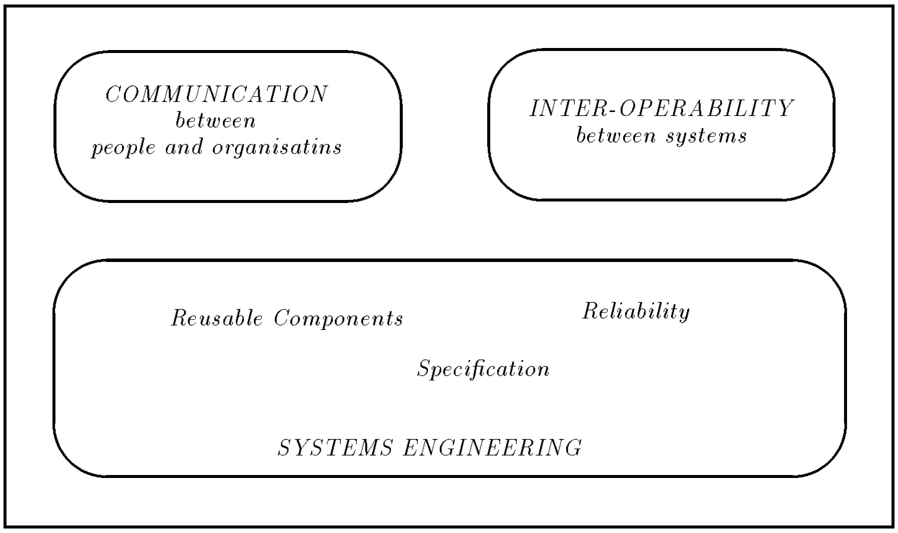
The degree of formality by which a vocabulary is created and meaning is specified varies considerably. Four somewhat arbitrary points along what might be thought of as a continuum are:

- highly informal: expressed loosely in natural language
- semi-informal: expressed in a restricted and structured form of natural language, greatly increasing clarity by reducing ambiguity,
  e.g. the text version of the 'Enterprise Ontology' ct[19] (see § 7.3.3).
- semi-formal: expressed in an artificial formally defined language,
  e.g. the Ontolingua version of the Enterprise Ontology;
- rigorously formal: meticulously defined terms with formal semantics, theorems and proofs of such properties as soundness and completeness.
  e.g. TOVE.

The following quote from the SRKB (Shared Re-usable Knowledge Bases) electronic mailing list nicely summarises what an ontology is and the various forms and contexts it arises in.

> "Ontologies are agreements about shared conceptualizations. Shared conceptualizations include conceptual frameworks for modeling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories. In the knowledge sharing context, ontologies are specified in the form of definitions of representational vocabulary. A very simple case would be a type hierarchy, specifying classes and their subsumption relationships. Relational database schemata also serve as ontologies by specifying the relations that can exist in some shared database and the integrity constraints that must hold for them."

Closing Remarks Our focus has been on identifying real problems and identifying real approaches to solving them. We aim to be non-controversial, merely reflecting how the term 'ontology' is being used in this community. A separate concern is the unfortunate fact that there is no agreed meaning of the term (see [15] for a competent analysis of this situation).



We identify three main categories of uses for ontologies. Within each, other distinctions may be important, such as the nature of the software, who the intended users are, and how general the domain is.

Figure 1: Uses for Ontologies

Part of the confusion is due to the fact that the central ideas and issues have been addressed in a number of contexts and fields, often using different terminology. For example, there is a strong similarity between a conceptual schema for a data base, and an ontology. Other areas concerned with these issues include knowledge representation and acquisition, ontologies for natural language understanding, domain modelling in software engineering, and enterprise integration.

3 Uses of Ontologies

In this section, we review and elaborate the motivations for ontologies that we discussed above. In doing so, we characterise the space of uses for ontologies.

The literature is currently rich with descriptions of ontologies and their intended purposes. At a high level, most seem to be intended for some manner of re-use. Some of these purposes are implicit in the various interpretations of the word 'ontology' that are commonly found in the literature, as noted in [15]; (*e.g.* a vocabulary for [9] vs a meta-level specification of, a logical theory [32, 40]). Other dimensions of variation include the nature of the software with which the ontology will be used, whether it is intended to be shared within a small group and reused within that context for a variety of applications, or whether it is intended to be re-used by a larger community. Some view their ontologies mainly as a means to structure a knowledge base; others conceive an ontology to be used as part of a knowledge base, *e.g.* by loading it in as a set of sentences which will be added to as appropriate; still others view their ontology as an application-specific inter-lingua (e.g. ATOS). [20, 21]

Another important motivation for ontologies is to integrate models of different domains into a coherent framework. This arises in business process reengineering (where we need an integrated model of the enterprise and its processes, its organisations, its goals, and its customers), in distributed multiagent architectures (where different agents need to communicate and solve problems), and in concurrent engineering and design.

With these intuitions, we sub-divide the space of uses for ontologies into the following three categories:

- Communication
- Inter-Operability
- Systems engineering: specification, reliability and reusability

3.1 Communication

Recall that ontologies reduce conceptual and terminological confusion by providing a unifying framework within an organisation. In this way, ontologies enable shared understanding and communication between people with different needs and viewpoints arising from their particular contexts. We will now consider in detail several aspects of the use of ontologies to facilitate communication among people within an organisation.

Normative Models Within any large-scale integrated software system, different people must have a shared understanding of the system and its objectives. By using an ontology, we can construct a normative model of the system. This creates a semantics for the system and an extendible model that can later be refined, and which allows semantic transformations between different contexts.

Networks of Relationships We can also use ontologies to create a network of relationships, keep track of what is linked, and explore and navigate through this network. Such a network is implicit within the system, but people often have different perspectives and perhaps use different assumptions. Thus there is a lack of shared understanding concerning the nature of the key relationships within the system. This is particularly important in applications which require the use of multiple ontologies from different domains. Ontologies serve to make all of these assumptions explicit by identifying the logical connections between elements across models of the system.

In general, we will also want the ontology to support the ability to reason about the impact of possible changes to the system. For example, using an ontology to support enterprise modelling allows us to capture a picture of the enterprise that can be reworked. We can then answer questions about the enterprise model, such as what-if scenarios related to changing different parts of the enterprise during reengineering.

Consistency and Lack of Ambiguity One of the most important roles an ontology plays in communication is that it provides unambiguous definitions for terms used in a software system. Any set of software tools should be able to maintain consistency among themselves and the ontologies, though they need not be uniform. There may be the problem that a user's ontology is different from the ontology supporting the tool. In this case, we must provide an environment that can represent the different meanings for terms used by different people ("meaning mapper"). This also involves identifying the relevant assumptions used by different people, tools, or ontologies and the ability to capture multiple synonyms and utilise them in translation to various audiences.

Integrating Different User Perspectives If we have a system with multiple communicating agents, this integration through shared understanding becomes vital. We face the challenge of integrating different perspectives while capturing key distinctions in a given perspective. For example, people in different positions in an organisation will have different perspectives on what the organisation does, what goals it achieves, and how these goals are achieved. There is also the problem of integrating global and local views of the system. By using an ontology to provide a normative model of the system, this integration can be achieved by assisting participants in communicating and coming to an agreement.

This also lays the groundwork for the development of standards within a community. By adopting a shared ontology, all participants use a standardised terminology for all objects and relations in their domains.

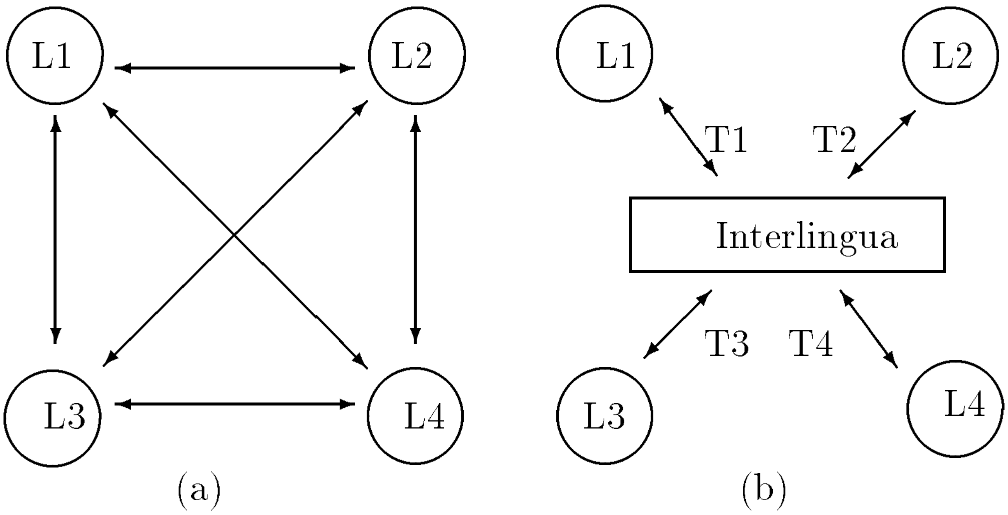
3.2 Inter-Operability

Many applications of ontologies address the issue of inter-operability, in which we have different users that need to exchange data or who are using different software tools. A major theme for the use of ontologies in domains such as enterprise modelling and multiagent architectures is the creation of an integrating environment for different software tools. Toolkits for spot solutions exist, but there is often no consistency among these tools.

3.2.1 Ontologies as Inter-Lingua

Any information technology environment for business process reengineering or multiagent systems should use integrated enterprise models spanning activities, resources, organisation, goals, products, and services. These integrated enterprise models serve as a common repository accessible by multiple tool sets.

This can also serve to integrate existing data repositories, either by standardising terminology among the different users of the repositories, or by providing the semantic foundations



To translate from language L_i to L_j and vice versa, a translator is required between L_i and the inter-lingua and another between the inter-lingua and L_j. Thus, given n languages, only $O(n)$ translators are required, not $O(n^2)$.

Figure 2: Ontology as Inter-Lingua

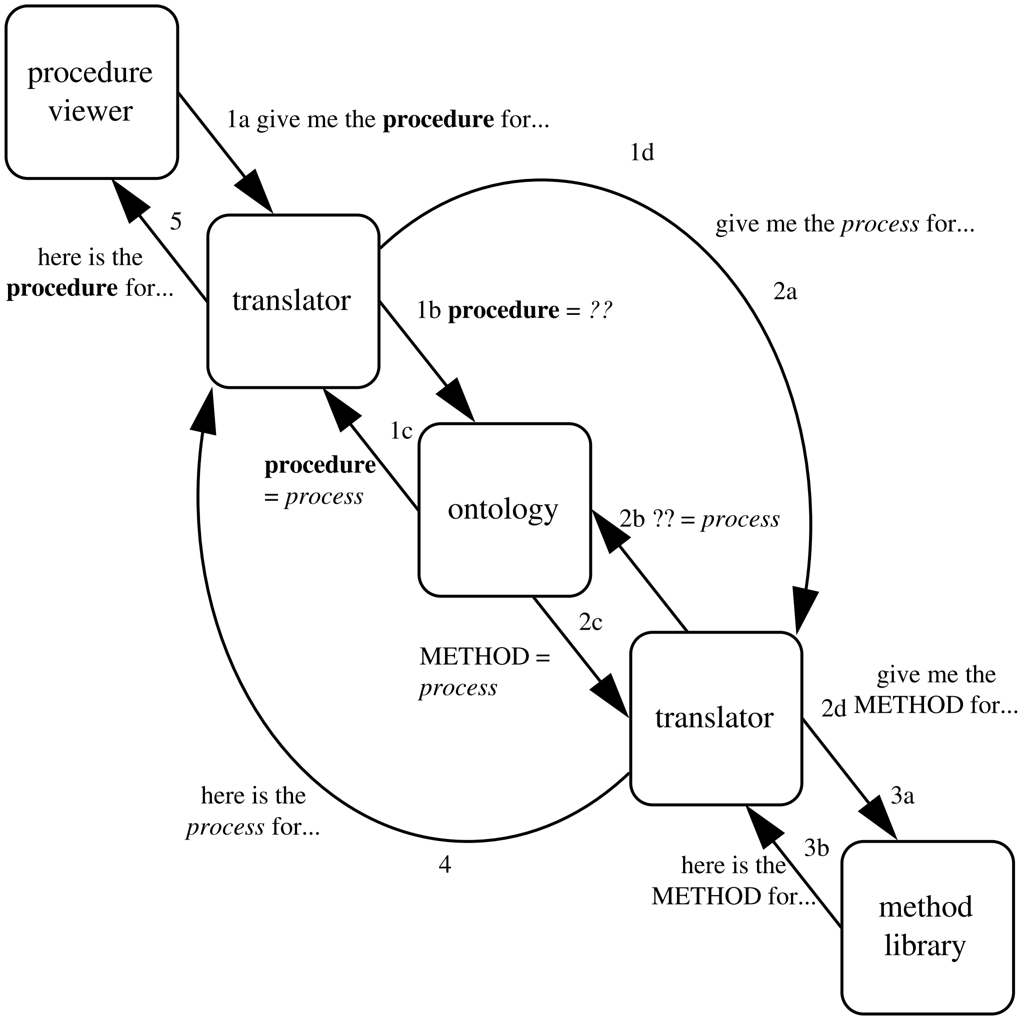
for translators among the different users.

To assist inter-operability, ontologies can be used to support translation between different languages and representations. One approach is to design unique translators for every two party exchange; however, this would require $O(n^2)$ translators for n different ontologies (see figure 2a). Using ontologies as inter-lingua to support translation, we can reduce the number of translators to $O(n)$ for n different ontologies, since it would only require translators from a native ontology into the interchange ontology (see figure 2b). This is the approach taken by the Process Interchange Format (PIF) Project.

3.2.2 Dimensions of Inter-Operability

In addition to tools and repositories, there are several distinctions that can be made. First, we need to consider the nature of the relationships among the users who are sharing tools and data. It is vital that the ontologies and tools used by different agents or organisations within the same enterprise be sharable and reusable across these multiple organisations.

Internal Inter-Operability With internal inter-operability, all systems requiring inter-operation are under the direct control of some organisational unit. Differences exist for historical reasons and legacy systems which will no longer change, need to be integrated.



This illustrates the use of an ontology as an inter-lingua to integrate different software tools. The term procedure, *used by one tool is translated into the term,* method *used by the other via the ontology, whose term for the same underlying concept is* process.

Figure 3: Ontology as Inter-Lingua: Example

External Inter-Operability With external inter-operability, we have an organisational unit that wishes to insulate itself from changes imposed on it from the outside (as in the semi-conductor example in § 2.3.2). Note that 'external' could mean another department in the same organisation.

Integrated Ontologies Among Domains The other distinction for inter-operability arises from the issue of the integration of ontologies from *different* domains in order to support some task. For example, an ontology to support workflow management systems will need to integrate ontologies for processes, resources, products, services, and organisation. The set of workflow tools would then use this set of integrated ontologies.

Integrating Ontologies Among Tools On the other hand, we may also need to integrate different ontologies in the *same* domain because of legacy systems. For example, different tools may use different process ontologies; to achieve inter-operability, we need to have a common ontology that both sets of tools can use. This is the most difficult challenge facing the use of ontologies, since it is usually not possible to impose the requirement of integration on the tools themselves; rather we need to construct ontologies for tools that are already being used.

3.3 Systems Engineering

The applications of ontologies that we have considered to this point have focussed on the role that ontologies play in the operation of software systems. In this section we consider applications of ontologies that support the design and development of the software systems themselves.

3.3.1 Specification

A shared understanding of the problem and the task at hand can assist in the specification of software systems. For example, the IBM Business System Development Method (BSDM) [17] develops and uses a ontology of the organisation as the basis for IT design and development in that organisation. The CommonKADS Conceptual Modelling Language (CML) is used to build domain and task ontologies to assist specification of knowledge based systems [31]. This idea is being further explored in the Kactus project (see § 7.3.4).

The role that ontologies play in specification varies with the degree of formality and automation within the system design methodology.

In an informal approach, ontologies facilitate the process of identifying the requirements of the system and understanding the relationships among the components of the system. This is particularly important for systems involving distributed teams of designers working in different domains.

In a formal approach, an ontology provides a declarative specification of a software system, which allows us to reason about what the system is designed for, rather than how the system supports this functionality.

3.3.2 Reliability

Informal ontologies can improve the reliability of software systems by serving as a basis for manual checking of the design against the specification. Using formal ontologies enables the use of [semi-]automated consistency checking of the software system with respect to the declarative specification.

In addition, formal ontologies can be used to make explicit the various assumptions made by different components of a software system, facilitating their integration. For example, in the the Integrated Development Support Environment (IDSE), semantic constraints and relationships between different tools must be maintained for successful tool integration. Axioms stating these constraints are interpreted and enforced semi-automatically, thus facilitating integration (see appendix B for further details). This is closely related to the use of declarative constraints to maintain semantic integrity in data bases [2].

Declaratively specified assumptions may explicitly restrict the applicability of a particular ontology to a problem domain [12]. By proving that the ontology is capable of supporting various reasoning problems, we can demonstrate the reliability of the software system within the domain.

3.3.3 Reusability

To be effective, ontologies must also support reusability, so that we can import and export modules among different software systems. The problem is that when software tools are applied to new domains, they may not perform as expected, since they relied on assumptions that were satisfied in the original applications but not in the new ones. By characterizing classes of domains and tasks within these domains, ontologies provide a framework for determining which aspects of an ontology are reusable between different domains and tasks.

Ontologies provide an "easy to re-use" library of class objects for modelling problems and domains. The ultimate goal of this approach is the construction of a library of ontologies which can be reused and adapted to different general classes of problems and environments. One such library is being constructed at the Knowledge Systems Laboratory using their online Ontology Server (see § 7.4).

To be useful, these ontologies must be customisable, both to the class of problems and the class of users, whether they be managers, consultants, or engineers.

Further, the ontologies in such a library must be extendible, allowing the incorporation of new classes of constraints and the specialisation of concepts and constraints for a particular problem.

One approach to extendibility is the notion of partially shared views [24] in the Process Interchange Format Project [25] (see § 7.1.1). There is a core PIF ontology which all translators operate with. In addition, there are different extensions of this core ontology which not all ontologies may share. In PIF, these extensions are captured by partially shared views, so that ontologies that have a partially shared view in common can translate without loss of expressiveness.

Similarly, in the KRSL Plan Ontology (see § 7.1.2), there is a set of modular specialised ontologies augment the general categories with sets of concepts and alternative theories of more detailed notions commonly used by planning systems, such as specific ontologies and theories of time points, temporal relations, and complex actions.

Closing remarks – Thus far, we have motivated the need for ontologies, clarified what they are and described a variety of circumstances in which they may be used. In the next few sections, we turn our attention to the process of building and evaluating ontologies. First we describe some of the important steps in building an ontology; these are then elaborated on in sections 5 and 6.

4 A Skeletal Methodology for Building Ontologies

Although there is much collective experience in developing and using ontologies, there is no field of ontological engineering comparable to knowledge engineering. In particular, there are no standard methodologies for building ontologies; nor is there much published in this area, even in the research literature.

In an attempt to begin filling this gap, we envisage a comprehensive methodology for developing ontologies to include the following:

- Identify Purpose and Scope;
- Building the Ontology;
  - ontology capture,
  - ontology coding,
  - integrating existing ontologies;
- Evaluation;
- Documentation;
- Guidelines for each phase.

Below, we briefly define each stage and indicate what if any work has been reported that could be used to develop a comprehensive methodology.

4.1 Purpose and Scope

It is important to be clear about why the ontology is being built and what its intended uses are. The previous section explores the space of possible uses; this can be a starting point in identifying the purpose of an ontology yet to be constructed. It will also be useful to identify and characterise the range of intended users of the ontology.

4.2 Building the Ontology

The identification of the purpose and scope of the ontology, at least in general terms, serves to provide a reasonably well-defined target for building the ontology. Three aspects to this are capture, coding, and integration of existing ontologies.

4.2.1 Capture

By ontology capture, we mean 1) identification of the key concepts and relationships in the domain of interest; 2) production of precise unambiguous text definitions for such concepts and relationships; 3) identification of terms to refer to such concepts and relationships; and finally, agreeing on all of the above.

Perhaps, the most directly relevant work reported is in [33], where Skuce argues for an intermediate representation of a conceptualisation which is more formal than loosely structured natural language, but less formal than a formal language. He proposes a specific format for such an intermediate representation, which is to include assumptions, justifications as well as precisely worded definitions.

In § 5 we describe the method successfully used for ontology capture in the development of the Enterprise Ontology in the Enterprise Project [19].

4.2.2 Coding

By coding, we mean explicit representation of the conceptualisation captured in the previous stage in some formal language. This will involve

- committing to the basic terms that will be used to specify the ontology (*e.g.* class, entity, relation); this is often called a 'meta-ontology' because it is in essence, the [underlying] ontology of representational terms that will be used to express the main ontology;
- choosing a representation language (which is capable of supporting the meta-ontology);
- writing the code.

With regards to choosing a language, possibly the most extensive work done in this area is the Plinius Project [36, 39] (see appendix A). They have experimented with a large variety of languages for representing their ontology in the materials science domain. These experiences could serve as a starting point for developing guidelines in choosing representation languages for ontologies.

Coding and capture are sometimes merged into a single step. Indeed, some of the design decisions of the KSL Ontology Editor[3] presume that ontology builders may be developing the conceptualisation on the fly[3]. This may be appropriate in some cases, however our experience suggests that many benefits derive from separating the two.

Insofar as an ontology is a kind of a knowledge base, there is a wealth of useful methodological guidance that is potentially applicable. A comprehensive methodology would make very clear what applies for building ontologies as opposed to knowledge bases in general. It will also clarify under what circumstances, if any, capture and coding stages may be merged. These are important research issues at this time.

4.2.3 Integrating Existing Ontologies

During either or both of the capture and coding processes, there is the question of how and whether to use [all or part of] ontologies that already exist. In general this is a very difficult problem. Some important progress in this area is described in [3] and [33]. The former is implemented in the KSL Ontology Server. Skuce's main point is that in order to agree on ontologies that can be shared among multiple user communities, much work must be done to achieve agreement. One way forward is to make explicit all assumptions underlying the ontology.

Overall, provision of guidance and tools in this area may be one of the biggest challenges in developing a comprehensive methodology for building ontologies. It is easy enough to identify synonyms, and to extend an ontology where no concepts readily exist. However, when there are obviously similar concepts defined in existing ontologies, it is rarely clear how and whether such concepts can be adapted and reused.

4.3 Evaluation

Gómez-Pérez [8] provides a good definition of evaluation in the context of knowledge sharing technology:

> "to make a technical judgement of the ontologies, their associated software environment, and documentation with respect to a frame of reference ...The frame of reference may be requirements specifications, competency questions, and/or the real world."

[3] Email communication with James Rice.

Some detailed work has been done on the evaluation of ontologies which could contribute to a comprehensive methodology for building ontologies [6, 7, 12]. The approach taken in some of this work, is to look first at what has been done in the field of KBS, and to adapt it for ontologies.

4.4 Documentation

It may be desirable to have established guidelines for documenting ontologies, possibly differing according to type and purpose of the ontology.

As pointed out by Skuce [33], one of the main barriers to effective knowledge sharing, is the inadequate documentation of existing knowledge bases and ontologies. To address these problems all important assumptions should be documented, both about the main concepts defined in the ontology, as well as the primitives used to express the definitions in the ontology (*i.e.* the meta-ontology).

The facilities provided by Ontolingua, and supported by the KSL Ontology Editor facilitate both formal and informal documentation of such assumptions. Though such facilities may be conceptually straightforward, they can have significant benefit.

4.5 Initial Guidelines for Designing Ontologies

A comprehensive methodology for building ontologies should also include a set of techniques, methods, principles for each of the above four stages, as well as indicating what relationships exist between the stages (e.g. recommended order, interleaving, inputs/output).

The first attempt to consolidate experience gained in developing ontologies is describe in [10]. This is summarised below as a set of design criteria – the emphasis is on sharing and reuse.

In subsequent sections, we further elaborate on these points, reporting on the authors' experiences in developing ontologies for enterprise modelling.

Clarity An ontology should effectively communicate the intended distinctions to humans who design agents. This means that ambiguity should be minimised, distinctions should be motivated, and examples should be given to help the reader understand definitions that lack necessary and sufficient conditions. When a definition can be specified in formal axioms, it should be. In all cases, definitions should be documented with natural language and examples to help clarify the intent.

Coherence An ontology should be internally consistent. At the least, the defining axioms should be logically consistent. Coherence should also apply to the parts of the definitions that are not axiomatic, such as the natural language documentation and examples.

Extensibility An ontology should be designed to anticipate the uses of the shared vocabulary. It should offer a conceptual foundation for a range of anticipated tasks, and the representation should be crafted so that one can extend and specialise the ontology monotonically. One should be able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of existing definitions.

The next two criteria help achieve extensibility.

Minimal ontological commitment An ontology should require the minimal ontological commitment sufficient to support the intended knowledge sharing activities. An ontology serves a different purpose to a knowledge base, and therefore a different notion of representational adequacy or completeness applies. A shared ontology need only describe a vocabulary for talking about a domain, whereas a knowledge base may include the knowledge needed to solve a problem or answer arbitrary queries about a domain. An ontology should make as few claims as possible about the world being modelled, allowing the parties committed to the ontology freedom to specialise and instantiate the ontology as needed.

While making too may ontological commitments can limit extensibility, making too few can result in the ontology being consistent with incorrect or unintended worlds (*i.e.* models) [13, 14]. For this reason, it is beneficial to make ontological commitments with respect to aspects *intrinsic* to a domain; the guideline above applies to contingent aspects of a domain.

Minimal encoding bias The conceptualisation should be specified at the knowledge level without depending on a particular symbol-level encoding. The encoding bias of an axiomatisation, that is, representation choices that are made purely for the convenience of notation or implementation, should be minimised. The goal is to enable knowledge sharing across agents that may be implemented in different representation systems and styles of representation.

This concludes the overview of an ontology development methodology and brief consideration of some initial guidelines. In the next section, we elaborate on the Ontology Capture phase.

5 Ontology Capture

Recall that ontology capture consists of identifying and defining the important concepts and terms. In this section, we outline a procedure for ontology capture, and describe in some detail, the actual process we went through in creating the Enterprise Ontology [19].

We do not present this as a set of normative guidelines, supposing that it is better than all other approaches. Rather, it is one approach which worked well in our particular circumstances. The emphasis here will be on *informal* techniques, where the output is a 'semi-informal' ontology consisting of very carefully defined terms expressed in a restricted natural language (*e.g.* in the form of a glossary). In § 6 we describe a more formal approach which is better suited to a different set of circumstances.

We consider the following four phases in turn: scoping, producing definitions, review, and development of a meta-ontology.

5.1 Scoping

Brainstorming — Have a brain-storming session to produce all potentially relevant terms and phrases; at this stage the terms alone represent the concepts, thus concealing significant ambiguities and differences of opinion.

Brainstorming worked well for us; however if collectively, those involved possess insufficient domain expertise, another corpus of knowledge may need to be consulted to ensure adequate coverage.

Grouping — Structure the terms loosely into work areas corresponding to naturally arising sub-groups. In our case, groups arose such that terms were more related to other terms *within* the same group than they were to terms in other groups. Specifically, for each term:

- provisionally categorise it for inclusion or exclusion, or note it as a borderline case. This was determined mainly by reference to a previously agreed requirements document;
- keep notes to record such decisions for future reference;
- group similar terms and potential synonyms together for further consideration.

Finally, identify semantic cross-references between the areas; *i.e.* concepts that are likely to refer to or be referred to by concepts in other areas. This information can be used to help identify which work area to tackle first to minimise likelihood of re-work (see below).

This concludes our consideration of informal methods for scoping; in § 6 we consider scoping in a more formal context.

5.2 Produce Definitions

During scoping, most of the important concepts and many terms will have been identified. The main work of building an ontology is producing definitions. Next, we consider when it may be important to do certain things before others; after this, we present some detailed guidelines and suggestions for reaching agreement, handling ambiguity and producing the final wording of definitions.

5.2.1 Deciding What To Do Next

Determining Meta-Ontology — Initially, do not commit to any particular meta-ontology. Doing so may constrain thinking and potentially lead to inadequate or incomplete definitions. Also, if it ends up being wrong, many definitions may have to be re-done.

Instead, let the careful consideration of the concepts and their inter-relationships determine the requirements for the meta-ontology. Keep in mind various possibilities, and use words and phrases in a consistent manner where appropriate (e.g. role, entity, relationship, type, instance).

Note that this guideline applies only to the informal capture phase. The meta-ontology must be determined before coding begins; also, it can be constrained by the representation language chosen.

Work Areas — Address each work area in turn. Start with work areas that have the most semantic overlap with other work areas. These are the most important to get right in the first place, because mistakes lead to more re-work. If there is little overlap between work areas, work on them in any order.

Terms — Proceed in a *middle-out* fashion rather than top-down or bottom up. That is, define the most fundamental terms in each work area before moving on to more abstract and more specific terms within a work area. In our experience, this makes it easier to relate terms in different areas more precisely. It also is likely to reduce potential for re-work.

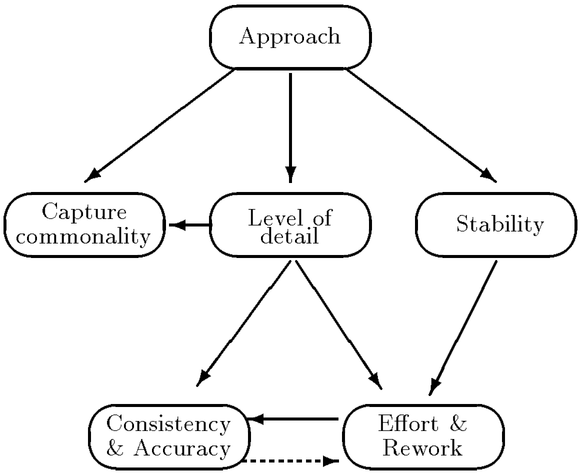
This is quite an important point, which is discussed in greater detail in [37]. The idea of what is fundamental, or basic, is a psychological phenomenon discussed at length in [23]. For example, 'dog' is basic, 'mammal' is a generalisation, and 'cocker spaniel' is a specialisation. While differences arise for individuals of widely varying expertise in an area, (*e.g.* a dog breeder may regard a particular species as basic), broadly, what is basic is the same for most people.

In the Enterprise Ontology [19], the basic concepts included 'sale', 'activity' and 'organisational-unit', among others. The idea of a market is a more abstract concept defined in terms of actual and potential sales. A specific kind of sale might be for a specific good or service in a particular area.

Benefits of a Middle-Out Approach

The choice of whether to go top-down, middle-out or bottom-up has a number of effects (see figure 4). A bottom-up approach results in a very high *level of detail*. This, in turn 1) increases overall *effort*, 2) makes it difficult to spot *commonality* between related concepts and 3) increases risk of *inconsistencies* which leads in turn to 4) *re-work* and yet more effort.

A top-down approach results in better control of the level of detail, however starting at the



Various issues are affected by the choice of whether to go bottom-up, top-down, or middle out. The latter makes it easier to spot commonality, results in stable models, and keeps the level of detail in control. This reduces inaccuracies which in turn leads to less re-work and overall effort.

Figure 4: Why Middle Out?

top can result in choosing and imposing arbitrary high-level categories. Because these are not naturally arising, there is a risk of less *stability* in the model which in turn leads to re-work and greater effort. The emphasis on dividing up rather than putting together also results, for a different reason, in missing the commonality inherent in the complex web of inter-connected concepts.

A middle-out approach, by contrast, strikes a balance in terms of the level detail. Detail arises only as necessary, by specialising the basic concepts, so some effort is avoided. By starting with the most important concepts first, and defining higher level concepts in terms of these, the higher level categories naturally arise and thus are more likely to be stable. This, in turn, leads to less re-work and less overall effort.

Stability and spotting commonality and are extremely important, as is illustrated in the following real examples. Using a bottom-up approach, a major UK aerospace company took over two years to produce a soon to be out of date model.

Many organisations use top-down analysis, which typically leads to a failure to recognise that the same entity can be both a buyer and a seller, with respect to to a given organisation. This results in a common occurrence whereby company A complains to company B that B owes A money, to which A responds that company B owes it even *more* money.

The root of the problem is the failure to recognise that the primary concepts are the legal entities and the relationships between them (*i.e.* sale agreement). Buyer and seller are secondary concepts; they are *roles* defined in terms of the sale agreement. The IT systems based on viewing roles as primary typically can not recognise that multiple roles may correspond to the same underlying legal entity. Consequently, banks lend money to companies already in debt to them (a different department!), and most people get multiple statements from the same financial institution, one for each account, rather than a single statement summarising all accounts. The idea of a role as a secondary notion is also discussed in [35].

The middle-out approach has been used successfully for many years as part of the BSDM, developed by IBM[17]; the problems noted above are largely avoided. A major part of this method entails the development of a shared understanding of the most important things in an organisation; this is used as a unifying framework for specifying and developing IT in the organisation. The 'shared understanding' is referred to as a business map, or a business model, but it is the same as what we are calling an ontology. See [22] for a detailed comparison between the ontology and the business modelling communities.

A major benefit of BSDM is stability; currently IBM are developing a significant business based on production of generic BSDM models which can serve as the basis for a number of companies in the same industry (*e.g.* insurance). Models which were not stable would be of limited use.

5.2.2 Reaching Agreement

There was considerable variation in the degree of effort required to agree on definitions and terms for underlying concepts. For some terms, consensus on the definition of a single concept was fairly easy. In other cases several terms seemed to correspond with one concept definition. In particular, there were several cases where commonly used terms had significantly different informal usage, but no useful different definitions could be agreed. This was recorded in notes against the definition. Finally, some highly ambiguous terms are identified as corresponding with several closely related, but different concepts. In this situation, the term itself *gets in the way* of a shared understanding.

Handling Ambiguous Terms — In the above special case where a term has many possibly meanings, we proceeded as follows:

1. Suspend use of the term; it is too ambiguous.
2. Clarify the *ideas* by carefully defining each concept using as few technical terms as possible, or only those whose meaning is agreed – consult the dictionary, thesauri, and/or other technical glossaries.
3. It can be helpful to give these definitions meaningless labels such as x1, x2, x3 etc. so they can be conveniently referred to in a neutral way.
4. Determine which, if any of the concepts are important enough to be in the ontology [usually one].
5. Choose a term for the concept, ideally avoiding the original ambiguous term (*e.g.* 'thing' rather than entity or object).

For example, consider the concept of *doing something*, for which there are a plethora of terms: activity, process, procedure to name a few. The following are all distinct, but closely related:

1. A category of something to do (*e.g.* physical activity)
2. A specific kind of thing to do (*e.g.* go from A to B)
3. A more specific kind of thing to do (*e.g.* go from York to London)
4. A general specification or plan of how to do something, a recipe, a set of instructions; for example:
   - go to train station in York
   - ride train to London
   - go to local destination in London
5. Something actually done (Ellen went form York to London on 1jan95)

In the Enterprise Ontology, two of the above concepts (4 and 5) were named and defined; they are called *Activity Specification* and *Activity* respectively.

Guidelines — In all cases the following guidelines were followed:

- Produce a natural language text definition, being as precise a possible;
- Ensure consistency with terms already in use; in particular:
  - make ample use of dictionaries, thesauri and other technical glossaries,
  - *avoid introducing new terms* where possible;
- Indicate the relationship with other commonly used terms that are similar to the one being defined (*e.g.* synonyms or variants referring to the same underlying notion, but perhaps from different perspectives);
- Avoid circularity in defining terms; this makes for increased clarity in general, but is essential if they will be later formalised;
- The definition of each term is intended to be necessary and sufficient as far as this is possible in natural language. Provide clarification or additional information essential to understanding the definition as separate notes following the definition;
- Give examples where appropriate.

Wording — Although the text version of the ontology served as the specification for producing code, there was a requirement that it be accessible to non-technical readers. To achieve an appropriate balance between technical precision and clarity, we :

1. kept the text definitions relatively informal;
2. equivalent, but more technically precise definitions cast using the primitives in the meta-ontology are used in documentation directly accompanying the code.

5.3 Review

Critically review definitions, revising as appropriate; where important decisions were made overturning previous decisions, keep track of the changes as a set of historical notes.

5.4 Meta-Ontology

Devise a meta-ontology, using the natural language definitions as an an implicit requirements specification. The main terms defined in the the Enterprise Meta-Ontology were Entity, Relationship, Role, State of Affairs and Actor. These served as the basis for the formal coding stage.

6 A Formal Approach to Ontology Design and Evaluation

Recall that the degree of formality by which the vocabulary of an ontology is specified varies from informal definitions expressed in natural language to definitions stated in a formal language such as first-order logic with a rigorously defined syntax and semantics. Similarly, recall that the uses of ontologies ranged from informal requirements such as a glossary for shared understanding among users to more formal requirements such as inter-operability among software tools.

Until now, we have mainly considered fairly informal methods for developing ontologies. At this point, we motivate the need for more formal methods, and then describe one approach to the design and evaluation of ontologies using a more formal framework.

6.1 The Role of Formal Languages

There are two important roles that formal languages play in the axiomatisation of an ontology – specification and implementation. In this section we will investigate the use of formal languages to specify and implement ontologies. We will conclude with some thoughts on the degree of formality that is required for a given ontology and domain of application.

6.1.1 Declarative Specification

A declarative specification of an ontology provides a characterisation that is independent of how the ontology is implemented. It allows us to reason about what the ontology is designed for, rather than how the ontology supports this reasoning.

A declarative specification has the following benefits:

No Extra-Ontological Distinctions Key distinctions are made *within* the language, so that all conclusions can be drawn from the ontology alone. Without this property, we would need to represent various assumptions procedurally outside of the language of the ontology. However, since one of the purposes of the ontology is to provide a framework for shared understanding, if these assumptions are not represented within our ontology, we risk a disagreement in how different agents interpret these extra-ontological assumptions.

No Hidden Assumptions All assumptions are made explicit. This is also addressing the challenge of shared understanding – what is an obvious assumption for one person is not obvious to another. As long as these assumptions remain implicit, the potential for disagreement is present.

This also plays a role in applying existing ontologies to new domains. Many software tools have been constructed in the context of a certain range of applications; when these tools are applied to new domains, they may not perform as expected, since they relied on assumptions that were satisfied in the original applications but not in the new ones.

Moreover, this provides an explicit characterisation of the relationships among different constraints.

Design Options There may be several ways of representing any given problem, and we often need to search through these different possibilities. For example, in business process reengineering, we want to find a new enterprise model which improves some aspect of the enterprise's performance. The problem is that we need to precisely define the set of possible alternatives.

A declarative specification of an ontology provides a precise and rigorous characterisation of this design search space. If the specification is consistent with the axioms of the ontology, then it is a possible alternative model.

Ontological Commitments An ontology is a specification used for making ontological commitments. Practically, an ontological commitment is an agreement to use a vocabulary (*i.e.* ask queries and make assertions) in a way that is consistent with respect to the theory that specifies the ontology. We build agents that commit to ontologies and we design ontologies so we can share knowledge with and among these agents.

With a declarative specification, we can explicitly reason about different ontological commitments. For example, we can compare two different proposals for an ontology with respect to the classes of objects that they require and the properties and relations among these objects that they postulate.

Modifiability If we change part of the ontology, we need to determine what else must be changed. With a declarative specification, we have a precise characterisation of the relationships among different sets of constraints used to represent a problem. Without such a specification, these relationships may not be explicitly represented but instead be implicit in some partially shared understanding (which not everyone may actually share).

Re-Usability By characterizing classes of domains and tasks within these domains, ontologies provide a framework for determining which aspects of an ontology are reusable between different domains and tasks.

Adequacy Criteria A declarative specification allows us to define rigorous criteria for adequacy. We will see this later in this section in the methodology for the design and evaluation of ontologies.

6.1.2 Implementing Ontologies

In this section we introduce the notion of a formal language for implementing an ontology; in particular, we will look at the Knowledge Interchange Format. We also introduce terms that we will need to discuss the formal methodology for designing and evaluating ontologies.

KIF (Knowledge Interchange Format) is a language that has been developed by the Inter-Lingua Working Group, under the DARPA Knowledge Sharing Initiative to facilitate knowledge sharing. Its features include:

- a formally defined declarative semantics,
- provision for the expression of arbitrary sentences in first-order logic. This gives it the expressive power to represent knowledge required for a typical application knowledge base.

In this section, we will present some of the terminology for KIF [5] which will be used in these notes. In § 7.4.1, we will revisit KIF and its role in implementing ontologies.

A *universe of discourse* is the set of all objects presumed to exist in the world.

There are four special types of expressions in KIF – terms, sentences, rules, and definitions. *Terms* are used to denote objects in the world being described; this includes *variables*, which are used in quantifying over individual objects, and *constants*, that are used to denote individual objects. *Sentences* are used to express facts about the world. *Rules* are used to express legal steps of inference. *Definitions* are used to define constants. We will informally use the word *axiom* to refer to sentences or definitions.

For every set of n objects, an n-ary *function* associates a unique object. For every set of n objects, an n-ary *relation* associates a truth value; the set of objects that evaluate to true specifies the objects that satisfy the relation. Function symbols in the language are used to denote functions and *predicate* symbols in the language are used to denote relations.

We will use the terms *axiomatisation* or *theory* to refer to the set of axioms that we use to represent the meaning of the terms in an ontology.

6.1.3 The Role of Automation

The formality required of the language for the ontology is to a large extent dependent on the degree of automation in the various tasks which the ontology is supporting.

If an ontology is a framework for communication among people, then the representation of the ontology can be informal, as long as it is precise and captures everyone's intuitions. However, if the ontology is to be used by software tools or intelligent agents, then the semantics of the ontology must be made much more precise.

For example, consider the role that ontologies play in supporting software tools for business process reengineering. At one end of the continuum, there are tools which are simply visualisations of the enterprise that facilitate communication and provide insight into the enterprise and its problems. By providing a mental model of the enterprise, the ontology supports opportunity identification as participants gain an understanding of how the enterprise succeeds or fails.

As we move along the continuum, we encounter BPR tools that provide analysis of a given enterprise model through evaluation, identification, and monitoring of different properties of the enterprise. In these different forms of analysis, we are considering alternative enterprise models, which includes alternative plans or schedules for activities, alternative organisations, or alternative sets of policies for people in the enterprise. We are also considering alternative explanations for different properties of the enterprise, and alternative predictions for possibly hypothetical behaviour of the enterprise.

Given this characterisation, the analysis tasks performed by the tools may simply compare the alternatives in a given set produced by the user of the tool. This type of analysis is performed by current simulation tools. To provide guidance for the user, the tools may themselves generate the set of alternatives which the user then evaluates. In the most automated form of analysis, the tools perform automated design by generating models, explanations, or predictions with particular properties.

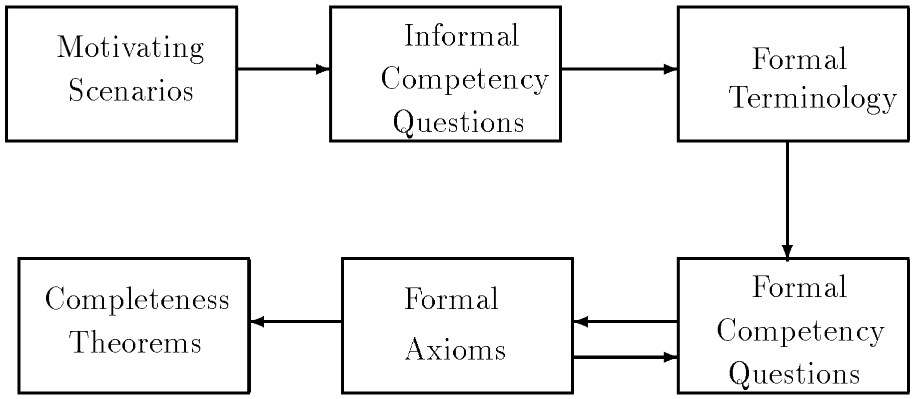


Figure 5: Procedure for a formal approach to ontology design and evaluation.

6.2 Overview of a Formal Methodology

For any given ontology, the goal is to agree upon a shared terminology and set of constraints on this terminology. We must agree on the purpose and ultimate use of our ontologies. We must therefore provide a mechanism guiding the design of ontologies, as well as providing a framework for evaluating the adequacy of these ontologies. Such a framework allows a more precise evaluation of different proposals for an ontology, by demonstrating the competency of each proposal with respect to the set of questions that arise from the applications. These justify the existence and properties of the objects within the ontology.

This section gives an overview of the methodology used in the Enterprise Integration Laboratory for the design and evaluation of integrated ontologies. Figure 5 outlines this methodology. It consists of the following steps:

1. Capture of motivating scenarios.
2. Formulation of informal competency questions.
3. Specification of the terminology of the ontology within a formal language such as first-order logic.
4. Formulation of formal competency questions using the terminology of the ontology.
5. Specification of axioms and definitions for the terms in the ontology within the formal language.
6. Justification of the axioms and definitions by proving characterisation theorems.

We will now consider each of these steps in more detail, using first-order logic as the formal language.

6.3 Motivating Scenarios

The development of ontologies is motivated by scenarios that arise in the applications. In particular, such scenarios may be presented by industrial partners as problems which they encounter in their enterprises. The motivating scenarios are story problems or examples which are not adequately addressed by existing ontologies. A motivating scenario also provides a set of intuitively possible solutions to the scenario problems. These solutions provide an informal intended semantics for the objects and relations that will later be included in the ontology.

Any proposal for a new ontology or extension to an ontology should describe one or more motivating scenarios, and the set of intended solutions to the problems presented in the scenarios. This provides a rationale for the objects in an ontology, particularly in cases when there are different objects in different proposals for the same ontology. By providing a scenario, we can understand the motivation for the prior ontology in terms of its applications.

6.4 Informal Competency Questions

Given the motivating scenario, a set of queries will arise which place demands on an underlying ontology. We can consider these queries to be expressiveness requirements that are in the form of questions. An ontology must be able to represent these questions using its terminology, and be able to characterise the answers to these questions using the axioms and definitions. These are the informal competency questions, since they are not yet expressed in the formal language of the ontology.

By specifying the relationship between the informal competency questions and the motivating scenario, we give an informal justification for the new or extended ontology in terms of these questions. This also provides an initial evaluation of the new or extended ontology; the evaluation must determine whether the proposed extension is required or whether the competency questions can already be answered by existing ontologies.

It may happen that people have prior informal ontologies for some application. In this case, for every object, attribute, relation, and axiom in the prior ontology or proposed extension to the ontology, there should first be an informal competency question which requires the objects or constraints defined with the object.

Ideally, the competency questions should be defined in a stratified manner, with higher level questions requiring the solution of lower level questions. It is not a well-designed ontology if all competency questions have the form of simple lookup queries; there should be questions that use the solutions to such simple queries. Consider Figure 6, which illustrates the structure of a competency question. Given a set of assumptions, constraints, and the set of sentences that are given in the statement of the question, there is some sentence that forms the query. For any competency question we want to specify the rationale for the question (which states how the answer to the question is used to answer more complex questions) and/or specify the decomposition of the question (in which we pose additional simpler questions which we must answer in order to answer the given question).

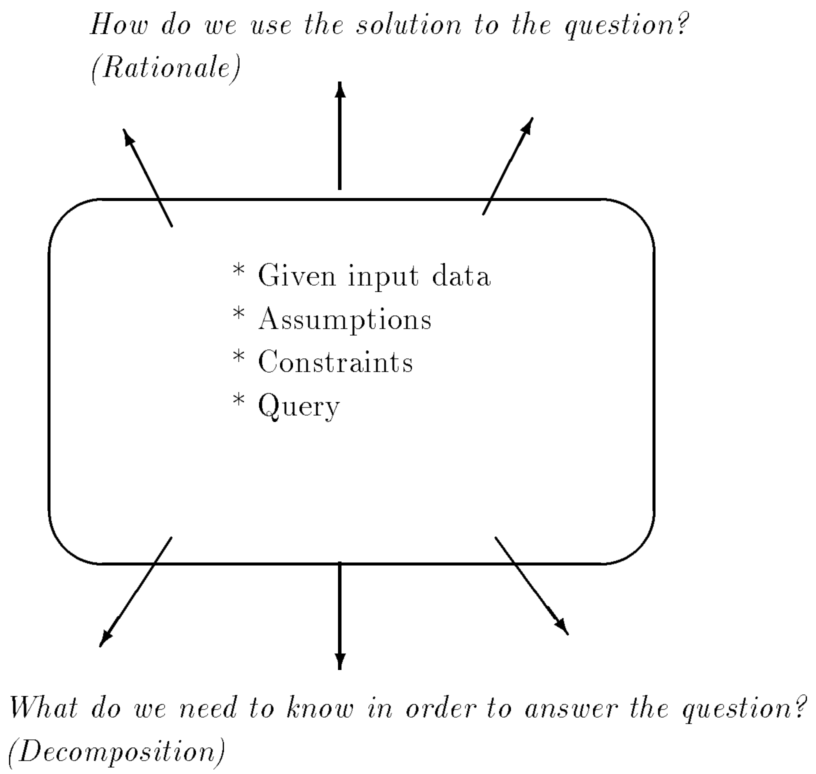


Figure 6: Stratification of competency questions.

The competency questions specify the requirements for an ontology and as such are the mechanism for characterising the ontology design search space. The questions serve as constraints on what the ontology can be, rather than determining a particular design with its corresponding ontological commitments. There is no single ontology associated with a set of competency questions. Instead, the competency questions are used to evaluate the ontological commitments that have been made to see whether the ontology meets the requirements.

6.5 Terminology

6.5.1 Informal Terminology

Given the informal competency questions, we can extract the set of terms used in expressing the question; these will form the basis for the specification of the terminology in a formal language.

The informal methodology for ontology capture described in the preceding section is particularly useful at this stage. In addition to identifying the set of terms, we must also produce informal definitions of the terms and address the problem of handling ambiguous terms. The informal dictionaries and glossaries defined using this methodology provide the intended semantics of the terminology and lay the foundations for the specification of axioms in the formal language.

6.5.2 Specification of Formal Terminology

Once informal competency questions have been posed for the proposed new or extended ontology, the terminology of the ontology is specified using a logical formalism such as KIF.

A formal ontology is a formal description of objects, properties of objects, and relations among objects. This provides the language that will be used to express the definitions and constraints in the axioms. This language must provide the necessary terminology to restate the informal competency questions. If we are designing a new ontology, then for every informal competency question, there must be objects, attributes, or relations in the proposed ontology or proposed extension to an ontology, which are intuitively required to answer the question.

The first step in specifying the terminology of the ontology is to identify the objects in the domain of discourse. These will be represented by constants and variables in the language. Attributes of objects and relations among objects are defined using predicates.

6.6 Formal Competency Questions

Once the competency questions have been posed informally and the terminology of the ontology has been defined, the competency questions are defined formally as an entailment or consistency problem with respect to the axioms in the ontology. Thus, they will have one of the following forms:

- Given the set of axioms in the ontology, and a set of instances of objects and relations in the ontology, can we infer some first-order sentence Q that uses only predicates in the language of the ontology?
- Can we determine whether some first-order sentence Q that uses only predicates in the language of the ontology is consistent with the set of axioms in the ontology and a set of instances of objects and relations in the ontology?

More formally, these can be stated as the following forms, where $T_{ontology}$ is the set of axioms in the proposed ontology, T_{ground} is a set of ground literals (instances), and Q is a first-order sentence using only predicates in the language of $T_{ontology}$.

- Determine $T_{ontology} \cup T_{ground} \models Q$
- Determine whether $T_{ontology} \cup T_{ground} \not\models \neg Q$

The axioms in the ontology provide the core axioms applicable to all objects and relations within the ontology as well as the definition of classes of objects; the set of instances of objects and relations in the formal competency question provides the constraints specific to a particular problem. For example, in a process ontology, the axioms of the ontology contain the definitions of complex actions and constraints on the occurrence of actions; the set of instances would contain the particular plan, schedule, or scenario of external events.

Every proposal for a new or extended ontology should be accompanied by a set of formal competency questions. It is also important to understand that all terms in the statement of the formal competency questions must be included in the terminology of the ontology. To have a formal declarative specification of an ontology, any sentences entailed using an ontology must be entailed by the axioms *alone*. It is only in this way that we can evaluate the ontology and claim that it is adequate, since this forces all intuitions to be made explicit.

Another important issue in the use of ontologies is the notion of a library of ontologies which can be adapted to different classes of problems. The challenge in this case is to determine which ontologies are the most appropriate for a given problem. Using the above methodology, ontologies may be distinguished by their corresponding competency questions; that is, one ontology may be able to represent a different set of competency questions than another ontology. In this case, the relationship between the ontologies can be formally represented by the questions.

6.7 Specification of Formal Axioms

The axioms in the ontology specify the definitions of terms in the ontology and constraints on their interpretation; they are defined as first-order sentences using the predicates of the ontology. It is important to understand the significance of using axioms to define the terms and constraints for objects in the ontology. Simply proposing a set of objects alone, or proposing a set of ground terms in first-order logic, does not constitute an ontology. Axioms must be provided to define the semantics, or meaning, of these terms.

It is also important to realise that this is not the implementation of the ontology; it is the specification of the ontology.

The process of defining axioms is perhaps the most difficult aspect of defining ontologies. However, this process is guided by the formal competency questions. As with the informal competency questions, the axioms in the ontology must be minimally sufficient to express the competency questions and to characterise their solutions; without the axioms we cannot express the question or its solution, and with the axioms we can express the question and its solutions. Further, any solution to a competency question must be entailed by or be consistent with the axioms in the ontology alone. If the proposed axioms are insufficient to represent the formal competency questions and characterise the solutions to the questions, then additional objects or axioms must be added to the ontology until it is sufficient. This development of axioms for the ontology with respect to the competency questions is therefore an iterative process.

The formal competency questions rigorously specify the requirements for the axioms in the ontology, and are the formal mechanism for characterising the ontology design search space. There may be many different ways to axiomatise an ontology. The formal competency questions do not determine these axioms, rather, the questions are used to evaluate the completeness of the sets of axioms in any particular axiomatisation.

This allows us to compare the expressiveness of different sets of axioms using the competency questions. If there is a competency question that one set of axioms can represent and another cannot, then the first set is more expressive with respect to that question. If two different axiomatisations can represent a competency question and characterise its solutions, then they are equivalent with respect to the question, and any comparison must use other criteria.

In some applications, there may be a common core ontology that is shared, while different groups use extensions specific to their applications. If this is the case, it is necessary to explicitly characterise the relationships between the core and the different extensions. In fact, the advantage of specifying ontologies in formal languages is that we are able to represent and reason about the ontological commitments for different applications.

6.8 Completeness Theorems

Once the competency questions have been formally stated, we must define the conditions under which the solutions to the questions are complete. This forms the basis for completeness theorems for the ontology. These theorems have one of the following forms, where $T_{ontology}$ is the set of axioms in the ontology, T_{ground} is a set of ground literals (instances), Q is a first-order sentence specifying the query in the competency question, and Φ is a set of first-order sentences defining the set of conditions under which the solutions to the problem are complete:

- $T_{ontology} \cup T_{ground} \models \Phi$ if and only if $T_{ontology} \cup T_{ground} \models Q$.
- $T_{ontology} \cup T_{ground} \models \Phi$ if and only if $T_{ontology} \cup T_{ground} \cup Q$ is consistent.
- $T_{ontology} \cup T_{ground} \cup \Phi \models Q$ or $T_{ontology} \cup T_{ground} \cup \Phi \models \neg Q$
- All models of $T_{ontology} \cup T_{ground}$ agree on the extension of some predicate P.

Completeness theorems can also provide a means of determining the extendibility of an ontology, by making explicit the role that each axiom plays in proving the theorem. Any extension to the ontology must be able to preserve the completeness theorems.

7 Ontologies in Practice

In this section, we will see how the ideas which we have presented are being used in practice. We will begin by giving an overview of several projects which are concerned with the construction of ontologies to support inter-operability. We will then broaden our scope and consider several endeavours within the industrial and academic communities concerned with the role of ontologies as standards. Next we consider several projects which have implemented ontologies dealing with various domains. We will conclude with a look at the KSL Ontology Server, a tool that is currently available for the design and development of ontologies.

This section does not provide an exhaustive review of existing ontologies; rather, it serves as an introduction to work in the field.

7.1 Ontologies for Inter-Operability

As we saw earlier, one of the purposes of ontologies is to serve as a unifying framework, both for shared understanding among users and for inter-operability of software tools. In this section, we will focus on two projects that address this problem of inter-operability.

7.1.1 Process Interchange Format

To assist inter-operability, ontologies can be used as inter-lingua in conjunction with translators (figure 2). The goal of the Process Interchange Format project [25] is to support the exchange of business process models among different process representations. Tools inter-operate by translating between their native format and PIF.

The project pursues the above goals by developing PIF (an inter-lingua to unify heterogeneous process representations) along with local translators between PIF and local process representations. It also provides a mechanism for extending PIF to accommodate different expressive needs in a modular way [24]. In this sense, there is a core PIF ontology with which all translators operate. In addition, there are different extensions of this core ontology which not all ontologies may share; for example, different sets of ontologies may have different ontologies for time, complex actions, or constraints. In PIF, these extensions are captured by partially shared views, so that ontologies that have a partially shared view in common can translate without loss of expressiveness.

The PIF project aims to support translations such that process descriptions can be automatically translated back and forth between PIF and other process representations with as little loss of meaning as possible. If translation cannot be done fully automatically, the human efforts needed to assist the translation should be minimised. If a translator cannot translate part of a PIF process description to its target format, it should translate as much of the description as possible (and not, for example, simply issue an error message and give up). In addition, it should represent any untranslatable parts so that the translator can add them back to the process description when it is translated back into PIF.

7.1.2 KRSL Plan Ontology

The first phase of the ARPA/Rome Laboratory Planning Initiative (ARPI) included development of the Knowledge Representation Specification Language (KRSL) for representing plans and planning information [26]. It was intended to provide a sharable ontology of planning information as an interchange medium for ARPI systems and as a means for specifying shared domain information.

The main aim of the ontology is to provide a shared vocabulary of concepts, relations, and conditions common to planning activities for use by disparate and communicating systems, as opposed to a particular language and syntax as exhibited by KRSL. To achieve this, the shared ontology has two major aspects – an abstract ontology setting out major categories (such as space, time, agents, actions, reasoning, and plans), and a set of modular specialised ontologies which augment the general categories with sets of concepts and alternative theories of more detailed notions commonly used by planning systems, such as specific ontologies and theories of time points, temporal relations, and complex actions. The specialised ontologies are also used to provide definitions of concepts in the case where several alternative sets of concepts are widely used to describe the same subject in the abstract ontology.

The relationship between these two sets of ontologies is similar to the relationship between core ontologies and partially shared views in PIF. In particular, the abstract ontology seeks to capture those general categories about which there is little disagreement, while the specialised ontologies provide means for expressing alternative views of the same subject matter as well as concepts not expressible in the abstract ontology.

7.2 The Role of Standards

Ontologies can also play a role in standardising representations among tools. A number of projects are currently being undertaken to provide some kind of standard in different domains of application:

- Workflow Management Coalition (WfMC)
- STEP and EXPRESS
- CORBA
- KIF and Conceptual Graphs

The standards play the same roles for inter-operability and shared understanding that we have already discussed.

7.2.1 Workflow Management Coalition

With the Workflow Management Coalition, a standard terminology is evolving which can serve as a common framework for different workflow management system vendors. This is manifest in one of the early outputs of the project, a glossary[29]. This document contains technical definitions for terms to be used in the WfMC specifications and discussions. The definitions themselves help in establishing a consistency in the usage of such terms. For each term the following is provided: a definition, a discussion of usage, and a set of possible synonyms. This glossary serves as a 'semi-informal' ontology for shared understanding analogous to the natural language version of the Enterprise Ontology.

The ultimate objective of the Workflow Management Coalition is to enable inter-operability between different workflow systems. This would require an interchange format similar to PIF.

7.2.2 STEP

STEP (Standard for the Exchange of Product Model Data) is an inter-lingua for defining and specifying products [18]. The primary motivation for STEP is to achieve inter-operability and to enable product data to be exchanged among different computer systems and environments associated with the complete product lifecycle. This includes design, manufacture, utilisation, maintenance, and disposal. This use may involve many computer systems, including some that may be located in different organisations. In order to support such uses, organisations must be able to represent their product information in a common computer-interpretable format that is required to remain complete and consistent when exchanged among different computer systems.

The overall objective of STEP is to provide a mechanism that is capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product data bases and archiving. The ultimate goal is an integrated product information database that is accessible and useful to all the resources necessary to support a product over its lifecycle.

STEP uses the formal specification language, EXPRESS, to specify the product information to be represented.

7.2.3 CORBA

CORBA (the Common Object Request Broker Architecture) is an emerging standard for retrieving objects and invoking operations on objects across a network [30]. It is a collaborative project developed by and endorsed by (to varying degrees) the members of the Object Management Group (OMG).

CORBA provides mechanisms by which objects transparently make requests and receive responses. The ORB provides inter-operability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems. It has a language Interface Definition Language (IDL) which specifies objects and operations to an ORB and allows operations to be invoked on those objects by remote/distributed applications. IDL is also supported by the KSL Ontology Server (see § 7.4), which provides a translator between IDL and Ontolingua.

There is also a definition of an Object Model, which defines what an object is in the CORBA space. The object implementation provides the semantics of the objects; in this sense, we can consider the object model to be a step towards an ontology.

The CORBA project also incorporates informal notions of ontologies. As part of the project, the Business Object Management group has developed a glossary of terms to be used in the object model. Although this glossary is not in itself an ontology, it does provide an informal framework for shared understanding.

7.2.4 KIF and Conceptual Graphs

KIF, a knowledge interchange format [5] (see § 7.4) and conceptual graphs [34] are both languages which can be used to represent ontologies. They were developed independently but both are based on first-order predicate logic, and thus are in essence, syntactic variants. However, there are differences in the details. Currently there is an effort underway to standardise these two languages; this will allow formal translation from one to the other and facilitate inter-operation of tools based on these languages [1].

7.3 Implemented Integrated Ontologies

In this section, we will discuss several endeavours within the industrial and academic communities that address the problem of implementing and integrating a set of different ontologies.

7.3.1 CYC

CYC [27] is a project of the Microelectronics and Computer Technology Corporation (MCC) in Austin, Texas that provides a foundation for common sense reasoning by developing ontologies for a wide variety of domain-specific applications.

All of the knowledge in CYC is represented declaratively in the form of assertions in a variant of first-order logic called CYCL. The CYC knowledge base itself contains simple assertions, inference rules, and control rules for inference; an inference engine can be used to derive new assertions using this knowledge base.

The ontologies underlying CYC are organised into sets of modules known as microtheories. Each microtheory captures the knowledge and reasoning required for some particular domain, such as space, time, causality, or agents. Multiple microtheories may exist for a given domain, reflecting the different perspectives and assumptions made by people modelling that domain. In this sense, CYC is not a monolithic integrated ontology; rather, it is a network of microtheories for a set of domains whose union covers the different ontological commitments that can be made within those domains.

7.3.2 TOVE

The goal of the TOVE (TOronto Virtual Enterprise) [11] project is to create an enterprise ontology that has the following characteristics: 1) provides a shared terminology for the enterprise that every application can jointly understand and use, 2) defines the meaning (semantics) of each term in a precise and as unambiguous manner as possible using First Order Logic, 3) implements the semantics in a set of Prolog axioms that enable TOVE to automatically deduce the answer to many 'common sense' questions about the enterprise, and 4) defines a symbology for depicting a term or the concept constructed thereof in a graphical context.

The TOVE ontologies constitute an integrated enterprise model, providing support for more powerful reasoning in problems that require the interaction of the following ontologies:

- Activities, states, and time
- Organisation
- Resources
- Products

- Services
- Manufacturing
- Cost
- Quality

This framework provides a characterisation of classes of enterprises by sets of assumptions over their processes, goals, and organisation constraints.

7.3.3 Enterprise

The overall objective of the Enterprise Project[4] [19] is to improve and where necessary replace existing modelling methods with a framework for integrating methods and tools which are appropriate to enterprise modelling and the management of change. This framework is based on an ontology for enterprise modelling.

A goal of the Enterprise Project is to provide a computer-based toolset which will help capture aspects of a business and analyse these to identify and compare options for the meeting the business requirements. The toolset will provide task management support to users by helping them perform enterprise modelling activities and guiding them through the toolset facilities. These facilities will enable 1) capture and description of an enterprise, 2) specifications of business problems/requirements (consistent with the ontology), 3) identification and evaluation of solution options and alternative design and implementation paths at strategic, tactical and operational levels, and 4) representations for the definition of relevant metrics and advanced simulation support.

The Enterprise Ontology — The Enterprise Ontology is conceptually divided into several major sections. These are listed below, along with a few of the most important concepts for each.

Meta-Ontology: Entity, Relationship, Role, Actor, State of Affairs;

Activities and Processes: Activity, Resource, Plan, Capability;

Organisation: Organisational Unit, Legal Entity, Manage, Ownership;

Strategy: Purpose, Strategy, Help Achieve, Assumption;

Marketing: Sale, Product, Vendor, Customer, Market.

Figure 3 illustrates how the ontology is intended to facilitate inter-operation among tools.

[4] The Enterprise Project is led by AIAI at The University of Edinburgh and the partners are IBM UK, Lloyd's Register, Logica and Unilever. The project is supported by the Department of Trade and Industry.

7.3.4 KACTUS

KACTUS [32, 40] is a European ESPRIT project aiming at the development of a methodology for the reuse of knowledge about technical systems during their life-cycle , so that we can use the same knowledge base for design, diagnosis, operation, maintenance, redesign, and instruction.

KACTUS supports an integrated approach embracing computer integrated manufacturing and engineering methods, and knowledge engineering methods by creating an ontological and computational basis for reuse of product knowledge across different applications within technical domains. It achieves this by creating domain ontologies and reusing them for different applications.

In addition, KACTUS attempts to integrate its ontologies with existing standards, such as STEP, by using the ontologies where available to capture domain data.

The main formalism in KACTUS is CML (Conceptual Modelling Language). This language was originally developed as part of the KADS and CommonKADS projects. CML can be used to model knowledge. CML is different from most other ontology formalisms in that it makes an explicit distinction between domain knowledge, inference knowledge, task knowledge and problem-solving knowledge. CML uses a notation that is mostly informal, that is knowledge modelled in CML cannot be executed by a program.

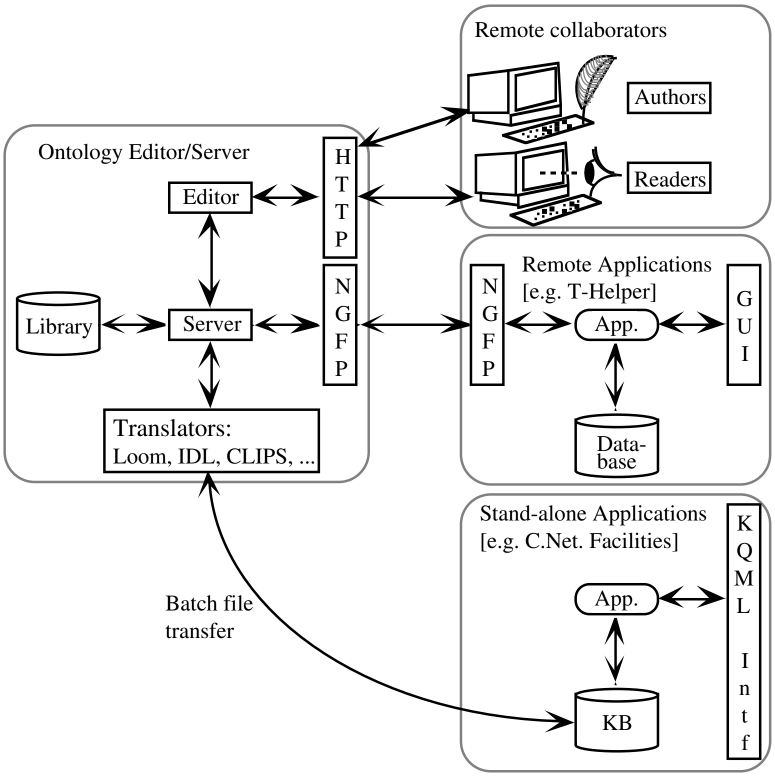
KACTUS also provides a toolkit that is an interactive environment for browsing, editing and managing (libraries of) ontologies. The KACTUS toolkit supports the theoretical and application oriented work packages by providing an environment in which one can experiment with theoretical issues (e.g. organisation of libraries of ontologies, mappings between ontologies, translating between different ontology formalisms) and also perform practical work (e.g. browse, edit and query ontologies in various formalisms).

In addition to CML, the KACTUS toolkit also provides support for EXPRESS and Ontolingua.

7.3.5 Plinius

The goal of the Plinius project [38] is semi-automatic knowledge extraction from natural language texts, in particular, literature on mechanical properties of ceramic materials. Since the texts cover a wide range of subjects, a set of integrated ontologies is required to cover concepts such as materials and their properties, processes to make these materials, and flaws of materials such as cracks and pores.

The ontologies in the Plinius project are a combination of formal and informal approaches. A lexicon is used to map natural language tokens onto formal expressions in the knowledge representation language. The ontology specifies a language in which the semantic part of the lexicon is expressed. See appendix A for further details of this project.



The Ontology Server supports three main types of usage. Authors and readers edit and browse ontologies using HTTP. An API allows remote applications to query and update ontologies. Stand-alone applications use ontologies after they are translated into the appropriate target languages

Figure 7: Ontology Server Architecture

7.4 Computer Support Tools: KSL Ontology Server

So far, we have mainly described conceptual approaches, proposals for standards, and research projects albeit in applied domains. Not discussed yet, is the need for computer support for the design, implementation and use of ontologies. In this section, we describe a suite of tools for this called the Ontology Server (or Ontology Editor) [3] provided by the Knowledge Systems Laboratory at Stanford University. The main functions it provides include creating, editing, evaluating, publishing, maintaining and browsing ontologies. Of particular importance is the ability to support collaborative work, which is greatly facilitated by its being accessible over the world-wide web. See figure 7 for an overview of the system architecture[5].

[5] This figure is reproduced from [3] with permission.

7.4.1 Background

The background and basis for the Ontology Server is Ontolingua [9], which [ambiguously] referred both to the *language* for representing ontologies, and to the *implementation*. The goals of the Ontolingua project were to overcome difficulties in knowledge sharing due to lack of consistency between knowledge bases with respect to vocabulary, semantics, and underlying assumptions. Ontolingua was conceived and built as a system for defining portable ontologies. Services it provided included parsing, cross-referencing and analysing ontologies. Portability was achieved by translating ontologies into various target languages which could be directly inserted into external software.

This was the first comprehensive attempt to demonstrate the ontology as inter-lingua idea which facilitates reuse, sharing, and improves potential for inter-operation. In a wider context, this work is part of a knowledge sharing initiative [4] which has been active for a number of years. An important early result from this effort was the development of KIF, of which Ontolingua, the representation language, is a mild syntactic variant.

Knowledge Interchange Format KIF [5] is designed to be an inter-lingua which, ideally, any knowledge base can be translated to/from. Crucially, KIF specifications are meant to be *sharable*. KIF is designed to be state of the art, *i.e.* able to represent most/all of the important concepts and distinctions available in today's advanced knowledge representation languages.

It is based on predicate calculus, but is extended to cater for advanced capabilities such as defining terms; representing meta-knowledge; specifying sets and encoding commonly used facilities for non-monotonic reasoning. KIF includes "model-theoretic semantics for the language and an axiomatisation of the primitive object types such as sets, lists, relations, and functions" [9].

The idea is that new target languages will be catered for by development of new translators. Importantly, development of such translators is meant to be possible with no prior knowledge of other languages that the knowledge base may be translated into or from. Ontolingua used KIF to explore this idea.

A NOTE ON TERMINOLOGY – From this point on, we will adopt the current usage for the term 'Ontolingua', which refers to the *representation language* used by the current Ontology Server, and *not* the system. The term 'Ontology Editor' is also sometimes used loosely to refer to the whole system, though strictly, the editor is just part of the system.

7.4.2 Overview

Ontologies are specified using KIF syntax and semantics, augmented by natural language descriptions. The Server translates these ontologies into the representation language of choice. Currently, translators exist for nearly a dozen languages.

Ontolingua itself, is not an implemented knowledge representation language. Initially there was no support of automated reasoning, although as the Ontology Editor develops, some simple and useful inferences are being incorporated. The intention is that Ontolingua specifications represent knowledge at the epistimological level in a clear unambiguous manner. Where practical, key assumptions (e.g. constraints; relationships) are formalised as KIF expressions, otherwise, they are expressed in natural language text.

The Ontology Server differs from and/or extends the original Ontolingua system in a number of important ways (see [3] for details):

- It is a remote compute server available on the world-wide web (http://www-ksl.stanford.edu/)
- It provides an extensible library of sharable reusable ontologies with suitable protections for proprietary, group, and private work.
- There is an extensive browsing capability which allows convenient viewing of ontologies. Currently, the format is most suitable for ontologies constructed using an object-oriented style.
- It has extended the original representation language to support decomposition of ontologies into modules and assembling new ontologies from existing modules from the library. This includes a mechanism for handling name conflicts.
- It provides explicit support for collaborative work. This includes the concept of a session to which multiple parties can be attached; parties are automatically informed of each others' activities.
- It has an application programmer's interface (API) which allows remote applications to query and modify ontologies stored on the Server over the Internet.
- As well as translating into multiple output languages, it also allows multiple input languages (*e.g.* CORBA's IDL). The fixed internal representation is semantically equivalent to a set of axioms represented in KIF. Ontolingua is one of the input languages.

7.4.3 Specifying Ontologies

An Ontolingua ontology consists of a set of definitions. There are four basic kinds of things to be defined: classes; relations; functions and instances. In addition, there are axioms relating the above terms. An Ontolingua definition consists of:

- argument list
- documentation text
- set of labeled KIF statements

An example definition from the Documents ontology (available from the Server) is given below:

```
(Define-Class Author
      (?X)
      "An author is an agent who writes things.  An
       author must have a name, which is its real name
       as an agent.  The name as author may or may not
       be the agent's name, but usually is."
      :Def
      (And (Agent ?X) (Has-One ?X Name))
      :Issues
      ((:See-Also "Has-Name augmented in this ontology.")))
```

'Name' can be thought of as a slot for the class Author. 'Has-One' is a second-order relation used to specify the cardinality of the slot.

The ':Def' portion of the definition gives necessary conditions on class membership. The English translation of the two conjuncts in the above definition are: (1) all authors are agents; (2) every author has exactly one name.

In addition to ':Def', there are other keywords for labelling statements in Ontolingua definitions. Below is a summary of some of the main keywords and their meaning:

`:Def`	necessary conditions (may have variables)
`:Iff-Def`	necessary and sufficient conditions
`:Lambda-Body`	KIF term to compute value of function
`:Axiom-Def`	necessary conditions (no variables)

Full documentation is available on line.

7.4.4 Translation

We will not discuss the details of translation, but some of the difficulties and tradeoffs are worth mentioning. It was designed to meet the following competing requirements:

1. the definition language is to be expressive, declarative and system independent;
2. it should support translation into less expressive languages;
3. It should be easy to add translations into additional target languages.

Achieving all three is impossible. Translation into less expressive languages means that translation will necessarily be incomplete.

The Ontology Server is biased towards an object-oriented representation style. The editor's input language steers ontology builders towards the object-oriented subset of KIF that uses the Frame Ontology, a set of idioms in First-Order Logic which are easily translated into object-oriented languages. For example, *(dog ?x)* $\rightarrow$ *(mammal ?x)* is an idiom for the subclass relationship (*i.e.* every dog is a mammal). The translator is equipped with special purpose code to recognise such idioms which is much less work than processing arbitrary expressions in first-order logic.

It is possible for users to write arbitrary axioms in KIF, but it is awkward for them to do so. As a result, most users most of the time write definitions that are readily translated into the object-oriented languages. The principle is: never prevent users from saying what they want to say, but encourage them to say things in a way that it is easy to work with.

Closing Remarks The Ontolingua language, and the overall approach to the ontology development and use supported by the Ontology Server appear to be emerging as de facto standards. They were found to be a major benefit during the formal encoding of the Enterprise Ontology, and seem to be the basis for much research in the field. One author of this paper found the developers to be extremely helpful and responsive, showing unusual commitment to his being a satisfied user. The Ontology Server is undergoing active development, and thus will continue to improve.

8 Conclusions and Future Directions

In this paper we motivated the need for a shared understanding to overcome barriers in communication between people, organisations and software systems. Such a shared understanding (*i.e.* ontology) can function as a unifying framework giving rise to a variety of benefits. We classified the various roles of ontologies as follows:

Communication between and among *people* and *organisations*, *e.g.* to unify different research fields (see page 3);

Inter-Operability: among *systems*, *e.g.* using the ontology as an inter-lingua to unify different languages and software tools (see figures 2 and 3);

System Engineering Benefits: Ontologies also assist in the process of building and maintaining software systems, both knowledge-based and otherwise. In particular,

- *Re-Usability:* the ontology, when represented in a formal language can be (or become so by automatic translation) a re-usable and/or shared component in a software system; *e.g.* the Ontology Server in batch mode (figure 7);
- *Reliability:* A formal representation facilitates automatic consistency checking;
- *Specification:* the ontology can assist the process of identifying requirements and defining a specification for an IT system, *e.g.* a BSDM 'map' [17].

We have also presented several methodologies and tools that can support the design and evaluation of new ontologies. We first considered an informal approach to developing ontologies that included the following steps:

- Identify purpose and scope;
- Ontology capture, including the production of precise unambiguous definitions for the terms of the ontology and agreement on these definitions;
- Ontology coding, including the specification of the meta-ontology and identification of the representation language for the ontology;
- Integrating existing ontologies;
- Evaluation;
- Documentation.

We then considered a more rigorous approach to the development of ontologies and discussed the role of formal languages in the specification, implementation and evaluation of ontologies. In this approach, the scope of an ontology is defined by a set of competency questions, which are different reasoning problems that the ontology is expected to support. The definitions and constraints of the ontology are formally evaluated with respect to these competency questions.

Throughout, we suggested guidelines based on our experience for various stages of development. These methodologies will become increasingly important as ontologies are developed for new domains and new classes of problems, and much further work remains to elaborate and improve them. For example, it is unclear under which circumstances different approaches are most appropriate

Frontiers of Ontology Research

There are many ways to exploit ontologies, but much research must be done to take full advantage. To date, there is no comprehensive review of the emerging field of ontologies from a research perspective. See [16] for a step in that direction.

We conclude this paper with a brief discussion of several important issues and opportunities for ontology research. These are:

- Development of ontologies as inter-lingua to support; interoperability among tools in some domain;
- Development of tools to support ontology design and evaluation;
- Development of libraries of ontologies;
- Development and integration of new ontologies;
- Methodologies for the design and evaluation of ontologies.

Ontologies for Inter-Operability As we saw with the various projects such as PIF, the Workflow Management Coalition, STEP, and the KRSL Plan Ontology, the development of ontologies that can serve as inter-lingua among a set of tools is becoming more prominent. Such ontologies, in conjunction with translators, facilitate integration among and between different sets of domain tools (*e.g.* see figure 3).

Given the large number of legacy systems, it will be difficult to design integrated ontologies that all of the systems can use. Rather, well-defined translators among these systems will be crucial, and this will require ontologies to support the translation. In this way, individual systems can maintain their private ontologies.

The development of new ontologies that can serve as inter-lingua among different software systems also has a major impact on reusability. Solutions found in one domain can be reused by other applications by translating the problems and solutions using the inter-lingua ontologies.

The challenges in this endeavour of developing new ontologies are in the identification of the different domains for the tools and ontologies, since the benefit of ontologies is greatest when they are generic and can be applied to a wide range of domains. For example, what are the differences between the PIF ontology, the KRSL plan ontology, and the ontologies in the Workflow Management Coalition? Perhaps there is some ontology that is generic enough that it can serve as an inter-lingua among these ontologies.

The enterprise domain is one where ontologies may provide great benefits, *e.g.* as as inter-lingua among business process reengineering tools. However, there are a number of independently developed enterprise ontologies whose differences are unclear, similar to the above situation.

In addition, there will be the challenge of integrating ontologies from different domains. For example, if we use a common process ontology such as PIF, how do we integrate this with a possible common organisation ontology or a common product ontology such as STEP/EXPRESS? Both of these ontologies would be used as inter-lingua for their particular domains, but there will some interaction among them. A product ontology is related to the process ontology through the notions of process plans; an organisation ontology is related to the process ontology through notions such as commitment and responsibility for performing different processes.

Tools for Ontology Design Given the challenge of integrating ontologies, we need to develop ontology design tools that can assist in the integration of ontologies. The development of tools to support ontology design and evaluation is particularly important for the case of distributed teams. The challenge is even greater when these teams are developing ontologies in different domains. In this case we need software tools that can manage consistency among the ontologies as they are being designed.

Steps in this direction are already being taken. We already discussed the Ontology Server [3]. In § 6 we described how competency questions can assist ontology evaluation. Gómez-Pérez is building tools to assist in evaluation of ontologies expressed in Ontolingua [6, 7].

Ontology Libraries As we saw earlier in this paper, one of the primary applications of ontologies is to facilitate reusability. The ultimate goal of this approach is the construction of libraries of ontologies which can be reused, customised, and adapted to different general classes of problems and environments. Users would construct the ontology appropriate for their problems by importing various modules of ontologies from the library. An example of this can be seen in the partially shared views of ontologies in the PIF project [24] and Process Handbook [28] projects.

An application of this approach will be to construct archives or repositories of ontologies that have been used in various domains. These domain-specific ontologies could then be structured into more generic classes so that they could be applied to a wider range of general problems. Such repositories of ontologies could be used to facilitate benchmarking among different companies. For example, a ship-building enterprise could reuse the ontologies designed for house construction enterprises. The challenges facing this application of ontologies are concerned with ownership of the ontology libraries; in particular, will companies want to share ontologies that could potentially be used by their competitors.

New Ontologies We need to develop more expressive ontologies for activities/processes, resources, products, services, and organisation. This includes support for reasoning about the constraints that agents must satisfy, such as goals, commitments, and policies.

The development of ontologies for new domains and new classes of problems is an important step in widening the application of ontologies. This includes ontologies for enterprise modelling, materials science and engineering, petrochemical and plastics industries, natural language lexicons, and medicine.

Integrating Ontologies An important conceptual challenge is how to choose between and/or merge ontologies when several exist in a given domain. Each may have subtly different assumptions, though significant overlap may exist. They may have been created from different perspectives, and therefore not easily re-used. We need to develop ontology design tools that can assist in the integration of ontologies as they are being designed.

In particular, there is the problem of developing new ontologies while maximising reuse of existing ontologies. Although one of the primary uses of ontologies is the support of knowledge sharing, surprisingly little work has been done on the actual integration of existing ontologies; a major exception being the KSL Ontology Server. Even for ontologies within the same domain, such as enterprise modelling and natural language glossaries, competing ontologies have been designed independently with little or no reuse of existing ontologies. At the very least, more work needs to be done in explicating the relationships among the various ontologies that have been developed within similar domains.

This situation is beginning to improve, if only slightly. For example, although the PIF, WfMC, and KRSL efforts had independent origins, there are now some formal links. The extent to which these ontologies can or should be merged is is unclear, however efforts are underway to ensure compatibility and uniformity where possible.

Methodologies Current methodologies for constructing ontologies have focussed on generic ontologies such as activities and time. We need new methodologies for designing domain-specific ontologies; such methodologies need to balance the need for expressing detailed domain knowledge with the objective of designing generic ontologies that maximize the benefit of reuse.

Acknowledgements

We are grateful to Martin King, IBM UK, for the analysis and examples of the benefits of the middle-out approach to building ontologies. Adam Farquhar provided information about and checked our descriptions of the KSL Ontology Server. Michael Georgeff did the same for our description of his invited talk at IJCAI95. John Fraser assisted in describing the Enterprise Project. Asunción Gómez-Pérez provided information about her work on evaluating ontologies. Nicola Guarino provided helpful comments on an early draft. Austin Tate drew our attention to the existence of the Workflow Management Coalition and links between this, PIF and the KRSL planning ontology. The use of BSDM industry models was brought to our attention by Martin Gladwell, IBM UK. Joshua Duhl drew our attention to various practical applications of ontologies. Florence Fillion and Chris Menzel provided details of the IDSE project. Paul van der Vet provided the summary of the Plinius project.

References

[1] ANSI. Conceptual graphs, a presentation language for knowledge in conceptual models; working draft of proposed american national standard. Technical Report X3T2/95-019r1, ANSI, 1995. Contact person: J.F. Sowa.

[2] S. Embury and P. Gray. Compiling a declarative high-level language for semantic integrity constraints. Technical Report AUCS/TR9506, University of Aberdeen, 1995.

[3] A. Farquhar, R. Fikes, W. Pratt, and J. Rice. Collaborative ontology construction for information integration. Technical Report KSL-95-63, Stanford University Knowledge Systems Laboratory, 1995.

[4] R. Fikes, M. Cutkosky, T. Gruber, and J. van Baalen. Knowledge sharing technology project overview. Technical Report KSL-91-71, Stanford University, Knowledge Systems Laboratory, 1991.

[5] M.R. Genesereth and R.E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[6] A. Gómez-Pérez. Some ideas and examples to evaluate ontologies. In *Proceedings of the Eleventh Conference on Artificial Intelligence Applications*. IEEE Computer Society Press, 1995.

[7] A. Gómez-Pérez. Guidelines to verify completeness and consistency in ontologies. 1996. to appear at the Third World Congress on Expert Systems.

[8] A. Gómez-Pérez, N. Juristo, and J. Pazos. Evaluation and assessment of knowledge sharing technology. In N.J. Mars, editor, *Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995*, pages 289–296. IOS Press, Amsterdam, 1995.

[9] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[10] T. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928, 1995.

[11] M. Gruninger and M.S. Fox. The logic of enterprise modelling. In J. Brown and D. O'Sullivan, editors, *Reengineering the Enterprise*, pages 83–98. Chapman and Hall, 1995.

[12] M. Gruninger and M.S. Fox. Methodology for the design and evaluation of ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing*. International Joint Conference on Artificial Intelligence, 1995.

[13] N. Guarino, M Carrara, and P. Giaretta. Formalizing ontological committment. In J. Doyle, E. Sandewall, and P. Torasso, editors, *National Conference on Artificial Intelligence Conference (AAAI-94)*. Morgan Kaufman, Seattle, 1994.

[14] N. Guarino, M Carrata, and P. Giaretta. An ontology of meta-level categories. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, pages 270–280. Morgan Kaufman, San Mateo, CA, 1994.

[15] N. Guarino and P Giaretta. Ontologies and knowledge bases- towards a terminalogical clarification. In N.J. Mars, editor, *Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995*, pages 25–32. IOS Press, Amsterdam, 1995.

[16] N. Guarino and R. Poli. The role of formal ontology in the information technology. *International Journal of Human-Computer Studies*, 43(5/6):623–624, 1995. This is an editorial introduction to a special issue on this topic.

[17] IBM. Introduction to business system development method. Technical Report GE19-5387-01, International Business Machines Corporation, 1990.

[18] Initial release of international standard (is) 10303. Technical Report IS 10303, International Standards Organization, 1994.

[19] Fraser. J., A. Tate, and M. Uschold. The enterprise toolset - an open enterprise architecture. In *The Impact of Ontologies on Reuse, Interoperability and Distributed Processing*, pages 42–50. Unicom Seminars, London, 1995. Further information about the Enterprise Project and Ontology is available on the World Wide Web from: *http://www.aiai.ed.ac.uk/~entprise/enterprise/*.

[20] Fuchs. J. and J. Wheadon. Prospective applications of ontologies for future space missions. In *The Impact of Ontologies on Reuse, Interoperability and Distributed Processing*, pages 83–96. Unicom Seminars, London, 1995.

[21] M. Jones, J. Wheadon, D. Whitgift, M. Niezatte, R. Timmermans, I. Rodriguez, and R. Romero. An agent based approach to spacecraft mission operations. In N.J. Mars, editor, *Towards Very Large Knowledges Bases - Knowledge Building and Knowledge Sharing 1995*, pages 259–269. IOS Press, Amsterdam, 1995.

[22] M. King. Knowledge reuse in business domains: Experience with ibm bsdm. In *The Impact of Ontologies on Reuse, Interoperability and Distributed Processing*, pages 97–107. Unicom Seminars, London, 1995.

[23] G. Lakoff. *Women, Fire and Dangerous Things*. University of Chicago Press, 1987.

[24] J. Lee and T. Malone. Partially shared views: A scheme for communicating among groups that use different type hierarchies. *ACM Transactions on Information Systems*, 8(1):1–26, 1990.

[25] J. Lee, G. Yost, and PIF Working Group. The pif process interchange format and framework. Technical Report 180, MIT Center for Coordination Science, 1995.

[26] N. Lehrer. Knowledge representation specification language. Technical report, DARPA/Rome Laboratory Planning and Scheduling Initiative, 1993. Reference Manual.

[27] D. Lenat and R.V. Guha. *Building Large Knowledge-based Systems: Representation and Inference in the CYC Project*. Addison Wesley, 1990.

[28] T. Malone, K. Crowston, J. Lee, and D. Pentland. Tools for inventing organizations: Toward a handbook of organizational processes. Technical Report 141, MIT Center for Coordination Science, 1993.

[29] Workflow Management Coalition Members. Glossary - a workflow management coalition specification. Technical report, The Workflow Management Coalition, 1994.

[30] T.J. Mowbray and R. Zahavi. *The ESSENTIAL COBRA: System Integration Using Distributed Objects*. John Wiley and Object Management Group, 1995.

[31] A. T. Schreiber, B. J. Wielinga, J. M. Akkermans, W. Van de Velde, and A. Anjewierden. Cml: The commonkads conceptual modelling language. In L. Steels, A. T. Schreiber, and W. Van de Velde, editors, *A Future for Knowledge Acquisition: Proceedings of the 8th European Knowledge Acquisition Workshop EKAW 94*, pages 1–25. Springer-Verlag, 1994. Volume 867 of *Lecture Notes in Artificial Intelligence*.

[32] G. Schreiber, B. Wielinga, and W. Jansweijer. The kactus view on the 'o' word. In *Workshop on Basic Ontological Issues in Knowledge Sharing*. International Joint Conference on Artificial Intelligence, 1995.

[33] D. Skuce. Conventions for reaching agreement on shared ontologies. In *Proceedings of the 9th Knowledge Acquisition for Knowledge Based Systems Workshop*, 1995.

[34] J. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison Wesley, Reading, MA, 1984.

[35] J. Sowa. Top-level ontological categories. *International Journal of Human-Computer Studies*, 43(5/6):669–686, 1995.

[36] P.H. Speel, P.E. van Raalte, P.E. van der Vet, and N.J. Mars. Scalability of the performance of knowledge representation systems. In *Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995*, pages 173–183. IOS Press, Amsterdam, 1995.

[37] M. Uschold and M. King. Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing.* International Joint Conference on Artificial Intelligence, 1995. Also available as AIAI-TR-183 from AIAI, The University of Edinburgh.

[38] P. van der Vet and N. Mars. Structured system of concepts for storing, retrieving, and manipulating chemical information. *Journal of Chemical Information and Computer Sciences*, 33:564–568, 1993.

[39] P.E. van der Vet and N.J. Speel, P.H.and Mars. Ontologies for very large knowledge bases in materials science: a case study. In N.J. Mars, editor, *Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995*, pages 73–83. IOS Press, Amsterdam, 1995.

[40] R. Wielinga, G. Schreiber, W. Jansweijer, A. Anjewierden, and F. van Hamelen. Framework and formalism for expressing ontologies. Technical report, University of Amsterdam, 1994. Esprit Project 8145 Deliverable DO1b1, available from http://www.swi.psy.uva.nl/projects/Kactus/Reports.html.

A The Plinius Project and its Ontology

by Paul E. van der Vet and Nicolaas J.I. Mars

© 1995 Knowledge-Based Systems Group,
Dept. of Computer Science
University of Twente, P.O.
Box 217, 7500 AE Enschede,
the Netherlands;
phone +31 53 489 36 90,
fax +31 53 489 29 27;
email: vet,mars@cs.utwente.nl.

A.1 Setting and scope

The Plinius project aims at semi-automatic knowledge extraction from short natural-language texts. The source texts for Plinius are the title and abstract fields of bibliographic document descriptions. They are taken from the on-line version of *Engineered Materials Abstracts* (EMA) produced by Materials Information.[6] A subset of descriptions of primary literature on mechanical properties of ceramic materials has been selected from an entire EMA volume.

The focus is on cheaper methods for knowledge acquisition. Texts are a major source of knowledge. Today, almost every text is produced in machine-readable form before printing. Fully manual acquisition of knowledge from texts is too expensive while fully automatic acquisition is an illusion. So we want to arrive at a point in between these extremes. The result can be characterised as human-aided knowledge acquisition by machines or machine-aided knowledge acquisition by humans, depending on the division of work-load. We are primarily interested in obtaining a precise assessment of the investments needed to acquire knowledge this way, including extra investments needed when the process is scaled up and/or ported to other domains. This assessment will help decide which division of work-load is economically optimal.

The source texts for Plinius are processed unedited, *i.e,*, in the form in which they are found on the EMA tapes. They have not been selected for easy processing. The process utilises linguistic and domain knowledge resources to obtain representations of text contents in a knowledge representation language. The representation are stored incrementally (per source text) in the interim knowledge base. A further process serves to integrate these knowledge representations into an integrated whole. It is a virtual process: it is a procedure one would carry out for particular subjects and with a particular view on integration.

[6] Materials Information is a registered trademark of The Institute of Materials, London, and ASM International, Metals Park OH.

A.2 Ontology development in Plinius

Among the more important design decisions underlying the approach chosen in Plinius is that of using an ontology as one of the core elements of the system. The texts cover a large range of subjects. In order to capture most of the contents, we need concepts for: materials and their properties, processes to make them, processes operating on samples (such as wear and creep), and flaws such as cracks and pores. Obviously, ontology construction is a major effort in Plinius. In our ontologies, the definitions of concepts are formal wherever possible and useful, and informal otherwise. Often, some of the more interesting relations between concepts (part-whole and subconcept-superconcept) are built into the concept definitions.

We have developed some parts of the ontology in the traditional, top-down fashion of recursive differentiation. This kind of approach produces taxonomic hierarchies. Other parts have been built in a bottom-up fashion, which represents a break with tradition.

We have also investigated the representation of ontologies in formal knowledge representation languages. In the course of a Ph.D. project of Piet-Hein Speel, ontologies at various stages of development have been formalised and often also implemented in description logics (such as CLASSIC, BACK, LOOM), furthermore in Ontolingua/KIF, Prolog, Conceptual Graphs [34] and L_{LILOG} language developed for the LILOG project.

The ontology fulfills three purposes directly relevant to the Plinius process:

1. By demanding that all domain knowledge in the system is expressed in ontology concepts, the various resources of the system co-operate smoothly.
2. The ontology specifies a language in which the semantic part of the lexicon is to be expressed. The transition from natural language to knowledge representation language is partly supported by a lexicon, which maps natural-language tokens onto formal expressions in the knowledge representation language. The ontology specifies the non-logical constants that may be used.
3. The ontology implicitly specifies the desired output of the language-dependent process. Any message in the source text that cannot be expressed in ontology concepts cannot occur in the output.

As a consequence of these purposes, the ontology is also helpful in deciding about further use of the interim knowledge base and of integrated knowledge derived from the interim knowledge base by means of one of the integration programs.

A.3 Further information

Some of our publications are listed below. Contact the authors for more information.

Paul E. van der Vet, Hidde de Jong, Nicolaas J.I. Mars, Piet-Hein Speel and Wilco G.

ter Stal, *Plinius intermediate report*, Memoranda Informatica 94-35, University of Twente, Enschede, the Netherlands, 1994.

Paul E. van der Vet and Nicolaas J.I. Mars, "Ontologies for very large knowledge bases in materials science: a case study", in: *Towards very large knowledge bases*, Nicolaas J.I. Mars (ed.), Amsterdam: IOS Press, 1995, pp. 73–83.

Piet-Hein Speel, *Selecting knowledge representation systems*, Ph.D. thesis, University of Twente, Enschede, the Netherlands. Available through WWW at URL `http://wwwis.cs.utwente.nl:8080/kbs/pubpage.html`.

B Using Ontologies to Enable Enterprise Model Integration

Florence Fillion (ffillion@kbsi.com) and Christopher Menzel (cmenzel@kbsi.com)
Knowledge Based Systems, Inc,
1500 University Drive East,
College Station, TX 77840, USA.
Tel (409) 260-5274

B.1 Introduction

This brief paper describes the role of ontologies in a commercial computational environment that supports the integration of enterprise models. The environment is called the Integrated Development Support Environment (IDSE) and was developed by Knowledge Based Systems, Inc. (KBSI) as part of the Information Integration for Concurrent Engineering (IICE) project funded by Armstrong Laboratory at Wright Patterson Air Force Base, USA.

The paper is organized as a set of questions and answers intended to guide the reader in understanding the role and appropriateness of ontologies in providing solutions to the problem of enterprise model integration.

B.2 What Is Enterprise Model Integration?

Enterprise modeling is concerned with the development of models of various aspects of an enterprise. Different types of modes have evolved to support different modeling activities: data models to describe the data managed by the enterprise, process models to describe the processes that take place within the enterprise, activity models to describe the business activities performed within and by the enterprise, and so on. These various enterprise models, if integrated, can provide better understanding and managing of an enterprise's complexity, enable simulation of alternative solutions to problems faced by the enterprise, and support the control and monitoring of an enterprise's operations. Informally speaking, models are integrated to the extent that they provide a coherent picture of the enterprise and support efficient management of inter-operations across various portions of the enterprise. Briefly stated, integrated models should have the following characteristics:

1. *Consistency:* Information carried by the models should be consistent. This ensures that the enterprise is modeled accurately. If consistency is not maintained, then the models cannot provide a coherent picture of the enterprise to support efficient enterprise management.
2. *Lack of ambiguity:* Terms used in the models must be disambiguated. This requirement is essential to ensure an understanding of the models and enable effective consistency checking across models.

3. *Logical connectedness:* Logical connections between elements across models must be identified and maintained. These logical connections will enable the implications of changes in one model to be propagated to other relevant models. This feature is critical in providing support for change assessment and the simulation of alternatives.

B.3 Why Is Enterprise Model Integration Hard?

There are three central impediments to enterprise model integration.

1. *Number of modeling methods*: Enterprise modelers use a wide variety of modeling methods tailored to different tasks and different types of information arising from different aspects of the enterprise (eg, static information that might be stored in an employee database, dynamic information involved in a planning model or a description of a manufacturing process, a complex array of information found in a detailed product design, etc.). This fact, in turn, entails two related challenges.

   First, because different modeling methods target different types of information, models developed under different methods cannot be integrated unless the underlying information types have been integrated in some fashion. For instance, the type of information captured in a process model differs considerably from the type of information expressed in a data model. Process models focus upon dynamic entities-processes, activities, events, and the like-that bear certain temporal relations to one another, while data models tend to focus on more static entities-classes, relations, and attributes. However, static and dynamic entities are of course, not unrelated. Most notably, the processes in a typical process model are usually represented as containing objects of the sort that might be represented in a typical data model. Thus, to be clear on the connections between a process-oriented information type and a data-oriented information type, the logical relations between processes and object classes need to be made precise.

   Second, even when the connections between different information types are clarified, there remains the problem that the different enterprise models are expressed in different modeling languages. A complex representation like a data model or a business process model carries the information it does in virtue of some established, systematic connection between the components of the representation and the world. It is this connection that determines the semantic content of the representation. Hence, the information carried by a model cannot be shared and accurately interpreted by, a (computer or human) agent in the enterprise unless this agent is attuned to the semantic rules of the representation of the corresponding modeling method.

2. *Number of modeling tools:* In today's computerized age, modelers use software tools to develop their models. These are typically stand alone tools developed independently of one another. Integrating enterprise models means first being able to integrate the tools that were used to developed these models. Tools can be considered integrated in an environment if there exist mechanisms that enable logical connections between models to be computationally maintained in the environment.

The main impediment to enterprise modeling tool integration is that it is not possible, for all practical purposes, to impose strict requirements on the tools themselves. A useful model integration environment should allow any enterprise modeling tool to be easily integrated in the environment. Hence, it is not possible, for example, to require from the tools that they use a given dynamic data linkage capability such as the Dynamic Data Exchange Protocol. If such a requirement was made on the tools, then only those tools that can be modified or built to satisfy those requirements could be integrated.

3. *Number of differing contexts:* To integrate two or more models is, among other things, to identify relevant logical connections between elements of those models. Each enterprise model is created and maintained by some set of agents in a given context. However, the agent across different context often live in very different worlds; each brings to his or her own context (and hence to the corresponding enterprise model) a unique body of background knowledge, with its own distinctive logic, cast in its own distinctive vocabulary. Such knowledge constitutes a semantic backdrop without which it is impossible in general to interpret the data generated in that context, and hence the bearing of the models created and maintained in that context on models in other contexts. Thus the information carried by the data in a model-whether explicitly in virtue of the basic semantics of the data or implicitly in virtue of cross-contextual constraints-cannot be shared with other contexts that lack that knowledge.

B.4 Why Ontologies?

The descriptions of the main impediments to enterprise model integration lead to the following observations.

- An initial step toward solving the first impediment to model integration is to capture a description of the types of information managed by the various existing modeling methods and their inter-relationships, and to describe and disambiguate the languages that are used to capture those types of information. The specification of the ontology of a method provides both formal and informal definitions of the method's basic semantic categories and the logical connections between those categories. Such ontologies can then be used to provide manual and/or automated support to the understanding of models captured using those methods.
- An important approach to the second impediment to enterprise model integration-the problem of having numerous, independent enterprise modeling tools-is to provide a neutral computational medium in which the information contained in the various models and their inter-relationships can be represented and maintained. For this solution to work, however, there must be a way to describe the way the information stored in the modeling tools can be represented in the neutral medium. Here, again, what is needed is a somewhat formal description of the terms and elements used by a tool's internal representation system. Such specification (in essence, a specification of the ontology of the modeling tool) can be used to create the necessary computational structures in the neutral medium to store model information captured by the tool.

- Finally, the third impediment-the problem of differing contexts-can be addressed by defining precisely the terms used in the various contexts, that is, by capturing the ontology of the domains relevant to the contexts in which the models are created. These ontologies can then be used manually or automatically to provide support for understanding the models and for drawing logical connections between elements in the models.

B.5 How Can Ontologies Enable The Integration Of Enterprise Modeling Tools?

The specification of a tool's ontology provides a formal, structured description of (i) the information types managed by the tool, (ii) how those types are structured and how data are stored within the tool, and (iii) the constraints that must be maintained on the types and their structure. In an integrated environment in which tools communicate and share information through a neutral medium (eg, some global repository of information), a tool's ontology specification can therefore be used to automatically integrate the tool into the environment.

How does tool integration work in the IDSE?

To understand how automatic integration of modeling tools is possible, it is necessary to describe in more detail how tools interact in the IDSE.

1. The IDSE contains a neutral medium (or global repository) called the Evolving System Description (ESD). The ESD is mainly composed of a knowledge base and a truth maintenance system that is used to enforce constraints on the information stored in the knowledge base.
2. Models are integrated through the ESD. This is done as follows: modeling tools send information contained in a model to the ESD using a dedicated communication language; model information is then stored in the ESD's knowledge base where it can be integrated with other information sent by other tools and where constraints on the elements of the model can be enforced. The communication language used in to exchange information between the ESD) and the modeling tools is an extension of the Knowledge Interchange Format (KIF) [URL http://www-ksl-stanford.edu/knowledge-sharing/papers/index.html#kif]
3. Model information sent by a tool to the ESD can be stored in the knowledge base only if the object and data structures necessary to store those types of information exist in the knowledge base. In the IDSE, these structures are created automatically by the ESD from the ontology specifications of the tools.

How do ontologies enable the automatic integration of tools?

In the IDSE, tool integration is achieved as follows:

1. The ontology of the tool is captured (by the tool's developer or a tool expert) using an ontology capture tool. In the IDSE, a prototype tool called the Ontology Capture and Browsing Tool (OCBT) was used. The OCBT implements the IDEF5 Ontology Capture Method [URL http://www.brooks.nf.mil/HSC/AL/HR/HRG/DOCS/docs.htm] and is serving as the basis for the development of a commercial tool.
2. The resulting ontology specification is sent (electronically) to the ESD using the IDSE communication language.
3. The ontology specification is interpreted by a component of the ESD and the data and object structures necessary to store the information types described in the ontology are computationally created in the knowledge base (there is no need for human intervention during this process). The constraints and axioms contained in the ontology are also computationally interpreted and stored in the ESD's truth maintenance system.
4. The knowledge base is now ready to store any model information sent by the tool using those newly created object structures, and is ready to enforce the rules and constraints defined on these object structures.

What are the advantages of using ontology specifications to integrate enterprise tools within an environment?

There are four primary advantages to using ontology specifications to integrate enterprise modeling tools:

- Ontologies enable the tool integration process to be automated. This is because ontology specifications provide a description of a system (in this case a tool) that is both formal and precise enough to be computer processable and interpretable.
- The automation of that process, in turn, greatly facilitates the integration of tools in the environment and eliminates the need to reconfigure the ESD every time a tool is added to the environment.
- Ontology specifications eliminate the need to hard-code the object structures and their associated constraints in the ESD. Hence, the process of upgrading a tool that is part of the environment is greatly facilitated and enhanced. All that is needed is for the ontology of the tool to be modified accordingly and for the changes to be sent to the ESD interpreter. There is no need to modify the code and recompile any of the components of the environment.
- Finally, ontology specifications provide a documented design rationale, or justification, of how the information sent by a tool to the ESD is stored in the knowledge base. If the process was done manually, it is likely that the rationale for choosing a given object and data structures would go undocumented.

B.6 How Can Ontologies Enable The Integration Of Enterprise Models?

Ontologies also provide support for model integration by enabling the correct interpretation of models. Model integration, of course, presupposes model interpretation. In order to draw logical connections between, test the consistency of and draw inferences from a set of enterprise models, those models must be reasonably well understood. There are two ways in which models can be misinterpreted and those two ways mirror the first and third impediments to model integration. A model captured using a given method can be misinterpreted if (i) the information types managed by the method, their inter-relationships, and their relationships to the world are not well understood by the agent interpreting the model, or (ii) the agent interpreting the model is unfamiliar with the terminology used in the domain in which the model was developed. Both problems can be addressed by the use of ontologies-method ontologies to solve the first problem and domain ontologies to solve the second.

How are method ontologies used to interpret models?

A method ontology is essentially a characterization of the information type of a method, its primitive semantic categories, their properties, and their logical connections. By browsing a method ontology, then, an agent can better understand a model. To illustrate, consider the data modeling method IDEF1X. The basic semantic categories of IDEF1X are 'entity', 'attribute', and 'link'. These, and a number of auxiliary notions needed to capture the logical connections between these categories, are formally characterized in the IDEF1X ontology. Now suppose that a model contains the entities 'Employee' and 'Department' and the one-to-many link 'employs' between the entity 'Department' and the entity 'Employee'. To correctly interpret this model and integrate it with other models, an agent needs to know the implication of a link being of the type one-to-many (in this case that an employee can only be employed by one department but that a department can employ one or more employees). This information is contained both formally (through axioms) and informally (in description strings in the ontology of the method. Hence, by browsing the IDEF1X ontology, an agent can better understand the implications of the information contained in the model on both the enterprise being described and other models of the enterprise.

Because a method ontology contains both informal and formal descriptions of the semantic categories of the method, it is also used by the ESD's truth maintenance system to enforce rules and constraints defined in the method. Following the example above, if an agent was to relate an instance of the entity 'Employee' with two or more instances of the entity 'Department' through the relation 'employs', the truth maintenance system would signal the violation of an axiom and ask the agent to resolve the conflict.

How are domain ontologies used to interpret models?

In the IDSE, a domain ontology is essentially a sophisticated data dictionary that provides formal and informal definitions of the terms used in the domain and the concept they denote, and that describes precisely the relationships between those concepts. Like a method ontology, then, a domain ontology can be browsed by an agent to determine the meaning of terms used in a model. This browsing activity supports three activities that are central to model integration. By using domain ontologies to interpret models, an agent can (i) disambiguate the terms used in the models, ie, the agent can identify situations in which the same term is used with different meanings in different contexts; (ii) recognize when different terms used in different contexts denote the same concept; and (iii) identify dependencies and logical connections between elements in the same model and across models. These three activities are critical for ensuring the consistency of models and their integration.

Domain ontologies are also used by the ESD's truth maintenance system to enforce constraints on domain elements and to propagate changes. To illustrate, consider this simplistic example of a furniture ontology that contains (i) the concepts of table, where some of the characteristics of a table as described in the ontology are its height, the length of its legs, and the thickness of its top; and (ii) an axiom that states that the height of a table is equal to the thickness of its top plus the length of it's legs. Given any value for two of the three attributes for a given table (say its height and the length of its legs), the ESD's truth maintenance system would use the axiom in the ontology to deduce the value of the third attribute (the thickness of its top). If values were given for the three attributes and the axiom was violated, the ESD's truth maintenance system would detect an inconsistency and ask the user to resolve it.

B.7 What Are The Advantages Of Using Ontologies For IDSE Technology?

Ontologies provide significant advantages in the development and use of IDSE technology. The most prominent advantage of using ontologies is automation. Because of their formal nature, ontology specifications are well suited to be interpreted and used directly by computer programs. This advantage was crucial in the IDSE in that it enables the automation of the tool integration process and, hence, makes the IDSE an expendable, easy to update environment. Another important advantage of using ontologies is that they provide an application-independent specification of a domain, method, or modeling tool. These specifications represent, in essence, a reusable and documented recording of an enterprise's corporate knowledge. Because they are application independent, these specifications are not biased in their description of concepts in the domain they describe and the relationships between these concepts. Thus, they can be reused in a number of projects and situations. Finally, because of their formal and informal aspects, ontologies can be used both as mechanisms to automate various processes (such as tool integration, change management, or constraint propagation) and as documentation support for model interpretation. This dual use of ontologies makes a system such as the IDSE very attractive in that the logical connections and dependencies between model elements are implemented as they are described in the ontology.

What Are The Potential Commercial Applications Of This Technology?

The IDSE prototype has demonstrated the feasibility and the advantages of using ontology technology to enable model integration. There are several important commercial applications that will use this technology. First, KBSI is planning to release, in 1996, a commercial tool that enables the capture, browsing, and sharing of ontological information. This tool is based on the OCBT prototype developed during the IDSE and and will be part of an integrated modeling workbench (also developed at KBSI) expected to be released in 1996. The ontology capture tool will provide support for design rationale capture, corporate knowledge capture, and enterprise model integration.

Many concepts and techniques developed during the building of the ESD are being incorporated in the integrated modeling workbench, in particular, the techniques necessary to computationally maintain logical connections across models.

The modeling method ontologies developed during the IDSE have been a key in the development of the integrated modeling workbench. They have provided a crucial support in studying the logical connections between the semantic categories of the methods, thus enabling the design of translation rules between the methods. These rules are being incorporated in the integrated modeling workbench.

Finally, the use of ontologies as mechanisms to enable task automation is a very promising technology. Although KBSI has no short term plan to use this technology in commercial tools, it has been using and plans to continue using this technology in research projects and in the development of prototypes.